1 Жизненный цикл приложения базы данных

Базы данных, как и все разновидности программного обеспечения, требуют постоянного изменения в соответствии с той частицей реального мира, которую они отображают. Разработка БД – только первый шаг, необходимо представлять весь жизненный цикл, чтобы сразу не наделать ошибок.

Жизненный цикл БД имеет следующие периоды.

- *Планирование разработки БД* – определение наиболее эффективного способа реализации этапов жизненного цикла.
- *Определение требований к системе* – диапазона действия и границ приложения БД, состава пользователей и областей применения.
- *Сбор и анализ требований пользователей* – сведений о данных, их структуре, объеме, изменчивости; сведений о функциях пользователя, которые будут выполняться с помощью БД.
- *Проектирование БД* – создание проекта собственно БД, включая фазы концептуального, логического и физического проектирования.
- *Выбор целевой системы управления базами данных (СУБД)* – сравнительный анализ подходящих для реализации проекта БД СУБД и выбор из них наиболее оптимальной.
- *Разработка приложений* – конструирование пользовательского интерфейса и прикладных программ для ведения БД, контроля целостности данных, выполнения требований пользователя.
- *Реализация БД* – создание рабочей версии БД и прикладных программ в среде целевой СУБД.
- *Конвертирование и загрузка данных* – заполнение БД данными с нуля или преобразование и загрузка данных из унаследованных систем, на смену которым пришла наша БД.
- *Тестирование* – пробный запуск БД с целью обнаружить ошибки в ее работе, а также проверить способность БД выполнить все требования пользователя.
- *Эксплуатация и сопровождение БД* – полностью разработанное и реализованное приложение БД наблюдается в реальной работе, дополняется новыми требованиями пользователя, модернизируется с целью улучшения его характеристик (быстродействие, удобство работы и т.д.).

В жизненном цикле приложения баз данных одним из наиболее важных этапов является проектирование БД, от результатов которого зависит эффективность дальнейшего использования базы данных в автоматизации деятельности предметной области. Главная задача, которая решается в процессе проектирования, это организация данных: их интегрирование, структурирование и определение взаимосвязей. Способ организации данных определяется логической моделью, которая отражает основные сущности ПО и их взаимосвязи. Различные формы представления структур данных и связей между ними породили различные логические модели данных. К ним относятся иерархическая, сетевая, реляционная, объектно-реляционная и объектная модели данных. Наибольшую популярность, начиная с 80-х годов и до сегодняшнего дня, имеет реляционная модель данных в силу ее простоты и математической обоснованности. Как следствие, большинство современных СУБД поддерживают эту модель. Поэтому

настоящее пособие посвящено проектированию и разработке реляционных баз данных.

2 Этапы проектирования баз данных

При проектировании БД организацию данных принято рассматривать на трех уровнях: концептуальном, логическом и физическом (см рис. 1).

Рис.1. Этапы проектирования базы данных

Этим уровням соответствуют концептуальная, логическая и физическая модели предметной области. Весь процесс проектирования может быть разбит на три этапа.

3 Модели данных

Модель данных определяет правила, в соответствии с которыми структурируются данные.

Концептуальная модель описывает предметную область на содержательном уровне. На первом этапе при ее разработке осуществляется анализ предметной области, решаемых задач, запросов пользователей и документов, отражающих события и процессы, протекающие в ПО. Результатом этого анализа являются списки объектов предметной области, перечни их свойств или атрибутов, определение связей между объектами и описание структуры ПО в виде диаграмм. Для каждого из атрибутов указываются ограничения на их возможные значения, определяемые свойствами предметной области. Такие ограничения называются ограничениями целостности данных. *Концептуальная модель* объединяет в единое «обобщенное представление» концептуальные требования отдельных пользователей и служит средством общения между ними, поэтому разрабатывается без учета особенностей физического представления данных.

Логическая модель описывает объекты и связи предметной области на формальном уровне. Ее разработка ведется на втором этапе и основывается на концептуальной модели, полученной на первом этапе. В процессе разработки осуществляется выбор типа модели данных, а затем определение ее элементов. Для реляционной модели данных такими элементами являются отношения и связи между ними степени 1 : 1 (один к одному) или 1 : М (один ко многим).

Физическая модель данных определяет способ реализации данных непосредственно в среде целевой СУБД, а также их размещения на машинном носителе, учитывает распределение данных, методы доступа и способы индексирования. В современных прикладных программных средствах самый нижний уровень организации данных на магнитных носителях обеспечивается СУБД автоматически без вмешательства пользователя. Пользователь, как правило, оперирует в прикладных программах и универсальных программных средствах представлениями о логической организации данных. Поэтому основная задача проектирования - это создание концептуальной и логической моделей предметной области и базы данных и реализация их в СУБД.

4 Основы проектирования реляционных баз данных (РБД)

Пример 1. База данных о поставке деталей может быть описана следующими отношениями:

<имя отношения>(<список атрибутов>)

Деталь(<***номер детали***>,<*название детали*>,<*цвет*>,<*вес*>, <*город*>).

Поставщик(<***код поставщика***>,<*фамилия*>,<*город*>, <*статус*>).

Поставка(<***код поставщика***>,<***номер детали***>,<*количество*>).

Жирным шрифтом выделены первичные ключи, а подчеркиванием отмечены внешние ключи.

Ограничения на значения атрибутов могут быть заданы различными способами в рамках возможностей конкретной СУБД, используемой для

реализации БД. Наиболее распространенными из них являются, например, тип данных (домен), условие на его значение, уникальный индекс, внешний ключ, первичный ключ и другие.

Атрибут	**Тип**	**Ограничение**
Код поставщика	Символьный (3)	
Фамилия	Символьный (6)	
Город	Символьный (10)	
Статус	Байт	От 10 до 100
Номер детали	Целый	
Название детали	Символьный (6)	
Цвет	Символьный (6)	
Вес	Вещественный	Больше нуля
Город	Символьный (10)	
Количество	Целый	От 100 до 10000

Другая форма представления отношения – табличная. Каждому отношению соответствует таблица с таким же именем. Атрибуту в таблице соответствует столбец с именем атрибута, а каждому кортежу отношения - строка таблицы. Строка таблицы часто называется записью, а значение атрибута – полем записи.
Пример 2. Описание отношений БД «Поставка деталей» с помощью таблиц.

Деталь

Номер детали	Название детали	Цвет	Вес	Город
101	Болт	Черный	3	Москва
102	Муфта	Синий	9	Новгород
103	Шайба	Черный	2	Рязань

Поставщик

Код поставщика	Фамилия	Город	Статус
П1	Иванов	Ярцево	20
П2	Алексин	Курск	10
П5	Рязанов	Рязань	40

Поставка

Код поставщика	**Номер детали**	Количество
П1	102	400
П2	101	600
П1	103	1000

Здесь в отношении *Деталь* атрибутами являются *номер детали*, *название детали*, *цвет, вес*, *город.* Значениями этих атрибутов в первой строке таблицы будут соответственно <101, болт, черный, 3, Москва>. Каждая строка таблицы описывает конкретный экземпляр сущности «*Деталь*». Доменами атрибутов будут: для *номера детали* D_1 – множество целых чисел, для *названия детали* D_2 – множество символьных строк, названий предметов, для *цвета* D_3 – множество

символьных строк, названий цветов, для *веса* D_4 – множество целых чисел. В отношении три кортежа, каждый состоит из четырех упорядоченных элементов. Для отношения «Поставка» первичным ключом является составной ключ *{код поставщика, номер детали}*.

Таким образом, в ходе разработки реляционной базы данных должен быть определен состав логически взаимосвязанных реляционных таблиц и определен состав атрибутов каждого отношения с указанием ограничений на их допустимые значения. Состав атрибутов должен отвечать требованиям нормализации. Нормализация отношений может быть обеспечена на этапе логического проектирования БД.

Существует несколько нормальных форм реляционной модели данных, которые вводят ограничения и позволяют минимизировать дублирование данных, обеспечить целостность и надежность ввода данных. Для практического применения реляционной БД обязательной считается первая нормальная форма (1НФ). При этом каждый атрибут отношения должен быть простым, что соответствует принципу «в каждом поле только одно значение». Все последующие нормальные формы (2НФ – 5НФ) носят рекомендательный характер, однако всегда следует в проекте БД добиваться самой высокой нормальной формы и только в случае крайней необходимости идти на ее нарушение с целью добиться других преимуществ БД.

Практика проектирования БД в соответствии с технологией «сущностей-связей» показывает, что при правильном ее употреблении, отношения БД, полученные в результате проектирования, находятся в самой высокой НФ. Однако методология проектирования БД предусматривает использование теории НФ для окончательной оценки качества проекта.

5 Целостность реляционных данных

Логические ограничения, которые накладываются на данные, называются *ограничениями целостности*. Они формулируются в соответствии со свойствами предметной области в форме предикатов, которые для одних множеств данных имеют значение истина, для других – ложь. Ограничения используются в моделях данных для поддержания целостности данных при функционировании системы, т.е. СУБД должна обеспечивать непротиворечивость данных заданным ограничениям при переводе базы данных из одного состояния в другое. Использование ограничений связано также с адекватностью отражения предметной области с помощью данных, хранимых в БД. Существует два основных вида ограничений целостности данных: внутренние и внешние ограничения.

Внутренние – это ограничения, заложенные в выбранной модели данных и определяемые допустимыми структурами данных, связями между ними, допустимыми значениями наборов данных.

Внешние ограничения - задаются предметной областью и описываются ее бизнес-правилами, которым должны удовлетворять данные, чтобы иметь право находиться в БД.

Внутренние ограничения целостности реляционных баз данных делятся на два вида: ограничение по существованию и ограничение по связям.

1. *Ограничение по существованию* требует, чтобы потенциальный ключ отношения был всегда определен, т.е. он не может быть NULL-определителем (проще сказать, потенциальный ключ отношения не может оставаться пустым).

2. *Ограничение по связи (по ссылке)* требует, чтобы внешний ключ отношения был либо полностью определен, либо полностью не определен (был NULL-определителем), кроме того, каждое значение внешнего ключа должно совпадать со значением соответствующего ему потенциального ключа отношения-владельца связи.

Явные ограничения целостности данных определяются теми правилами, которые существуют в организации, например:

- возраст работника не должен превышать 40 лет;
- количество студентов в группе должно находиться в диапазоне от 7 до 25 человек;
- заработная плата работника должна изменяться только в сторону увеличения;
- количество продаваемого товара не должно превышать его количества на складе;
- в расписании занятий не должно быть двух строк с одинаковыми значениями атрибутов: Номер аудитории, День недели, Время начала и Фамилия преподавателя.

6 ER-технология проектирования БД

6.1 Общие понятия ER-технологии

Разработка концептуальной модели данных ПО может осуществляться различными методами. Наиболее эффективным и простым для понимания является метод «сущность-связь», или ER-метод (Entity - Relationship). Суть метода состоит в определении основных понятий предметной области: сущностей, связей и их характеристик, атрибутов, доменов, ключей. Основным достоинством ER-метода является возможность построения диаграмм ER-типов, отображающих в графической форме основные объекты предметной области, связи между ними, и характеристики этих связей.

Наличие диаграммы ER типов позволяет получить два преимущества.

- Во-первых, дает возможность в беседах с представителями предметной области (пользователями) обсудить результаты проектирования и выявить все ошибки и неоднозначности в проекте.
- Во-вторых, позволяет отобразить логическую модель данных предметной области на физическую модель данных реальной СУБД по четким правилам. То есть получить нормализованные базовые таблицы.

Остановимся более подробно на основных понятиях ER-технологии.

- *Сущность* – это объект, информация о котором должна быть представлена в БД (обычно соответствует существительному). Множество сущностей одного и того же типа образуют *Тип сущности*. Например, для типа сущности *Предприятие* сущностью является *ОАО РНПЗ или Агентство*

недвижимости «Лея». Иногда тип сущности называют просто сущность, а каждое значение сущности – экземпляр сущности.

- *Связь* – это отношение между сущностями. Множество связей одного и того же типа образует *Тип связи*. *Экземпляр связи* – это конкретная связь между конкретными экземплярами сущностей. Предложение «Студент Пухов А.Н. учится в группе 3010» представляет собой экземпляр связи «*Студент учится в Группе*».
- *Степень связи* – это количество типов сущностей, участвующих в типе связи. Степень связи может быть:
  - *унарной* – связь между сущностями одного типа (рекурсивная связь), например, работник X является супругом работника Y, или деталь 1 входит в состав детали 2 (см. рис. 2);

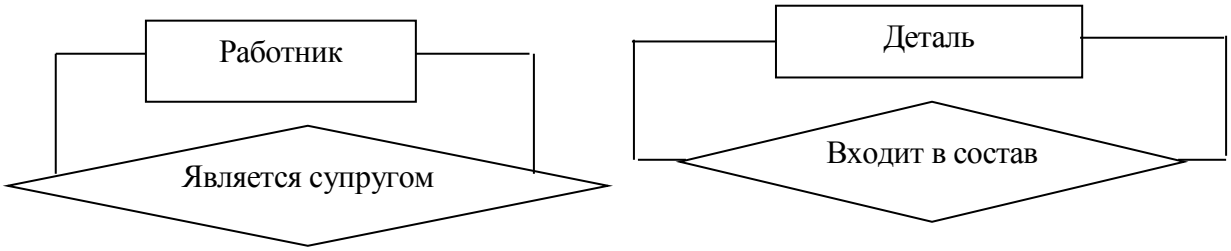


Рис. 2. Унарные связи между сущностями одного типа

  - *бинарной* – связь между двумя разными типами сущностей. Например, Сотрудник работает в Отделе, или Студент сдает экзамен по Дисциплине, или Товар поступает на Склад (см. рис. 3);

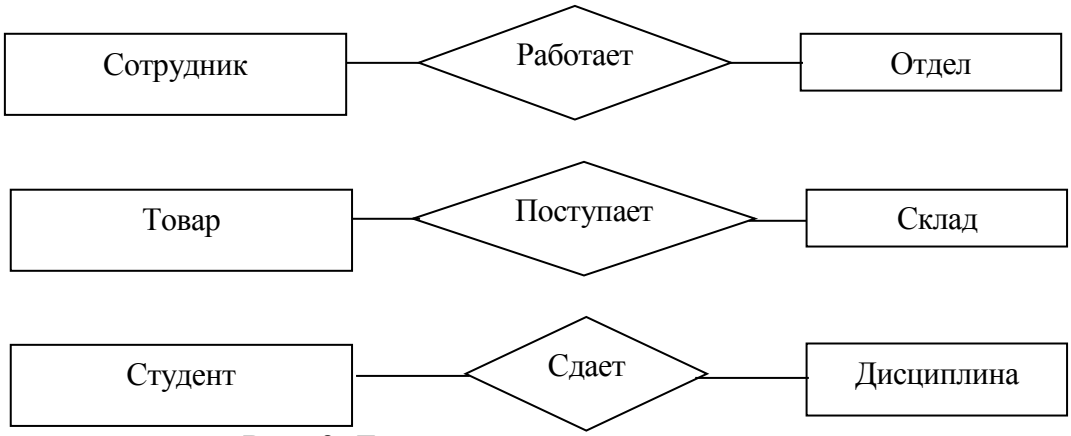


Рис. 3. Бинарные степени связи между сущностями

  - *тернарной* – связь между тремя разными типами сущностей. Например, Кафедра ведет Дисциплину на Специальности или Поставщик поставляет Товар в Магазин (см. рис. 4а, 4б)

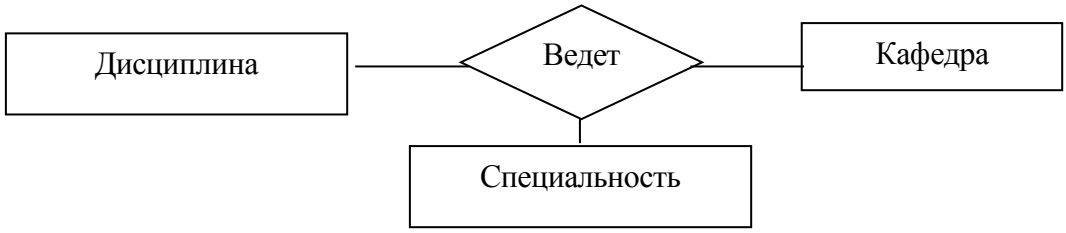


Рис 4а. Тернарные степени связи между сущностями разного типа

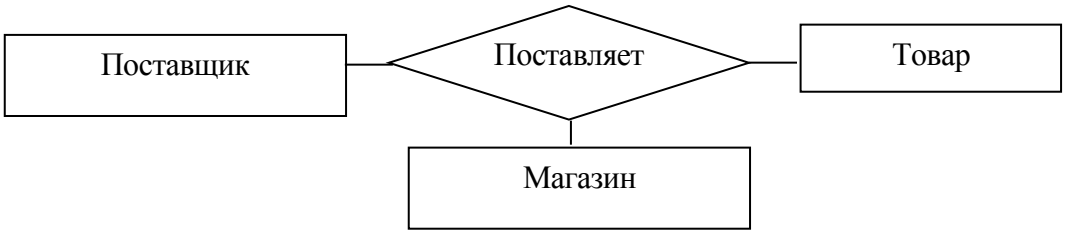


Рис. 4б. Тернарные степени связи между сущностями разного типа

N–арной – связь между N типами сущностей, например Преподаватель проводит Занятие в Группе по Дисциплине в Аудитории – это 5-арная связь Расписание (см. рис. 5).

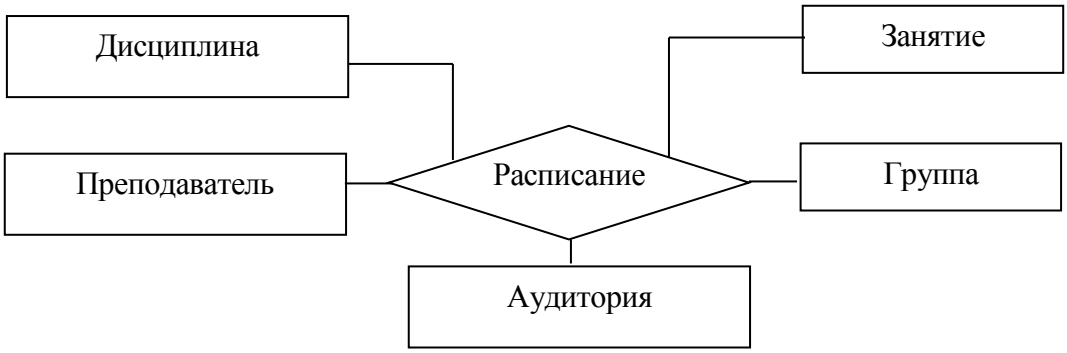


Рис. 5. Связь между сущностями степени 5

- *Степень участия в связи* – это величина, показывающая, со сколькими сущностями другого типа связана сущность данного типа. Эта характеристика типа связи определяется со стороны каждого типа сущности – участницы связи. Степень участия в связи называется также кардинальным числом (КЧ) связи. В общем случае КЧ может иметь два значения: минимальное КЧ и максимальное КЧ. На практике чаще всего степень участия в связи принимает значения 1:1 (один к одному), 1:M (один ко многим) и N:M (многие ко многим). Например, связь Директор школы *Руководит* Школой имеет степень участия 1:1, связь Договор *Заключает* Фирма – M:1, а Студент *Сдает* Дисциплину – M:N (см. рис. 6).

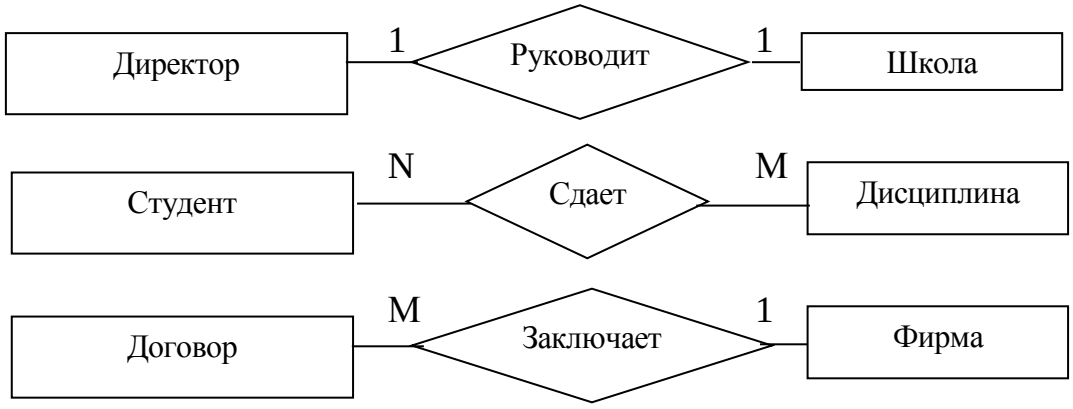


Рис. 6. Различные степени участия в связи.

- *Полнота участия в связи* – это величина, определяющая обязательно ли участие в связи каждой сущности-участницы связи. Эта величина может принимать одно из двух значений: полное участие (обязательное) или частичное участие (необязательное). Полнота участия может также определяться по

минимальному КЧ: если оно равно нулю, то участие необязательное (частичное), в противном случае – обязательное (полное). Например, участие каждой из сущностей Студент и Группа в связи *Учится* обязательное (не могут быть студенты без группы, а группы без студентов), а участие каждой из сущностей Студент и Дисциплина в связи *Сдает* необязательное, так как могут быть как студенты, которые еще ничего не сдавали, так и дисциплины, которые еще никто не сдавал.

- *Атрибут* – это свойство сущности или связи. Например, тип сущности Личность имеет атрибуты *Фамилия, Имя, Отчество, Номер паспорта, Дата рождения, Пол*; тип связи Ученик Получает оценку по Предмету имеет атрибуты: *Оценка, Дата*. Атрибуты могут быть простыми, составными, множественными, вычисляемыми (производными). Например:

- атрибуты: *Пол*, *Номер телефона*, *Фамилия, Оценка, Заработная плата* – являются простыми;
- атрибуты: *ФИО* (*Фамилия, Имя, Отчество*), *Дата рождения* (*День, Месяц, Год*), *Адрес* (*Индекс*, *Город, Улица, Дом, Квартира*) – являются составными;
- атрибуты: *День отправления* (*Пн, Вт, Пт, Сб*), *Время отправления* (*8:00, 16:30, 21:05*), *Категории водителя* (*A, B,*), *Телефоны* (*3-12, 3-33, 3-32*) – являются множественными;
- атрибуты: *Возраст, Средний балл, Суммарная заработная плата* – являются вычисляемыми, т.е. значения. Этих атрибутов могут быть вычислены по значениям других, а именно:

  Возраст = Текущая дата - Дата рождения;
  Средний балл = (*Оценка1 + Оценка2 +⋯+ Оценкаk*) / *k*,
  Суммарная зарплата = Зарплата за январь + Зарплата за февраль +⋯+ Зарплата за декабрь.

Реляционная модель данных поддерживает только простые атрибуты. Вопрос о том, включать или не включать в базу данных вычисляемые атрибуты зависит от того, насколько быстро приложение базы данных сможет их вычислить по запросу пользователя.

Атрибут или набор атрибутов, используемый для однозначной идентификации экземпляра сущности, называется *ключом сущности*. У одной и той же сущности могут быть от одного до нескольких ключей. Например:

сущность *Студент* имеет ключ – *Номер зачетной книжки*;

сущность *Владелец автомобиля* имеет ключи – *Регистрационный номер автомобиля, Номер водительского удостоверения;*

сущность *Химический элемент* имеет ключи – *Атомный вес, Порядковый номер, Обозначение*.

7 Пример выполнения проекта БД по ER-технологии

Задание: сконструировать множество отношений (таблиц) реляционной базы данных для предметной области «Бюро добрых услуг».

7.1 Анализ данных и выявление требований пользователя

Бюро добрых услуг, далее просто Бюро, занимается ремонтом оборудования

в котельных города и области. Эта организация имеет определённый перечень услуг, которые она может оказывать. Все услуги этого перечня разделены на разделы. Каждый работник предприятия может выполнять все услуги одного и только одного раздела. На все услуги в перечне установлены определённые цены и имеются единицы измерения количества услуг. Для оказания услуг у Бюро имеется в собственности различное оборудование.

Бюро имеет своих постоянных клиентов-заказчиков услуг, предоставляемых организацией. В основном Бюро периодически выполняет заказы этих клиентов. Каждый заказ включает определённое количество услуг, причём одна и та же услуга может выполняться при исполнении заказа несколько раз. Также иногда в заказе требуется покупка определённых материалов, поэтому смета выполнения заказа должна включать не только стоимость услуг с учётом количества выполнения каждой, но и распечатку использованных покупных материалов.

Таким образом, можно определить предварительные сущности предметной области.

- *Клиент* – список организаций-клиентов Бюро как тех, от которых в настоящее время есть заказ, так и потенциальных клиентов. Каждый клиент характеризуется такими атрибутами, как название организации, адрес, телефон, перечень оборудования, имеющегося у нее, особые замечания, счет в банке, статус (приоритет) обслуживания.
- *Сотрудник* – список работников Бюро как тех, которые заняты непосредственно исполнением заказов, так и руководящих работников и обслуживающего персонала, не занятых в исполнении заказов. Каждый сотрудник характеризуется такими атрибутами как: № паспорта, фамилия, имя, отчество, адрес, телефон, список работ, которые он может выполнять, квалификация.
- *Услуга* – список услуг, выполняемых сотрудниками Бюро при исполнении заказов. Каждая услуга характеризуется такими атрибутами как: код услуги, раздел группы услуг, наименование услуги, единица измерения, стоимость, перечень оборудования, требующегося для выполнения услуги.
- *Материал* – список материалов, используемых при выполнении заказов. Каждый материал характеризуется такими атрибутами как: код материала, наименование материала, единица измерения, стоимость, минимальный запас на складе Бюро.
- *Оборудование* – перечень оборудования, которое требуется для выполнения услуг. Каждое оборудование характеризуется такими атрибутами как: инвентарный номер, название оборудования, параметры, перечень услуг, для которых оно может применяться, факт занятости оборудования в текущий момент времени.
- *Заказ* – документ, отражающий информацию о желании клиента получить необходимые услуги по ремонту своего котельного оборудования. Каждый заказ характеризуется такими атрибутами как: номер заказа, заказчик (клиент), перечень услуг, оборудование и материалы, требующиеся для исполнения заказа, сотрудники-

исполнители заказа, дата заказа, дата выполнения заказа, сумма оплаты.

В беседах с представителями Бюро и путем анализа их деятельности были выявлены требования пользователя к будущей БД. Эти требования, в соответствии с заданием на проектирование, мы разделим на три группы.

1. Перечень запросов-документов:
   - смета стоимости заказа с распечаткой перечня и количества оказанных услуг;
   - смета покупных материалов, использованных при выполнении заказа;
   - печать визитки фирмы;
   - печать наклеек на конверты с адресами для отправления клиентам.
2. Перечень оперативных справок:
   - стоимость конкретного заказа;
   - общая стоимость заказов каждого клиента;
   - сумма заказов за конкретный период.
3. Перечень процессов преобразования и обработки данных:
   - подсчёт количества заказов каждого клиент;
   - определение стоимости материалов конкретного заказа;
   - определение стоимости услуг конкретного заказа
   -

7.2 Разработка концептуальной модели предметной области

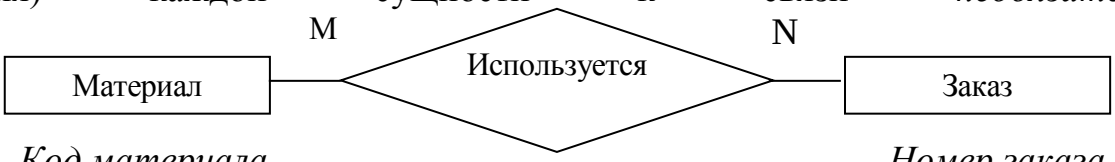
В результате анализа требований пользователя и структур данных, описывающих деятельность Бюро, следует определить как сущности следующие объекты.

- *Клиент* (*Название организации* – потенциальный ключ, *Адрес, Телефон* – потенциальный ключ, *Характеристика клиента* – примечание).
- *Сотрудник* (*№ паспорта* – потенциальный ключ, *Фамилия*, *Имя*, *Отчество*, *Адрес*, *Телефон* – потенциальный ключ, *Профессия* – должность).
- *Раздел* (*Номер раздела* – потенциальный ключ, *Название раздела*, *Характеристика* раздела – примечания).
- *Услуга* (*Код услуги* – потенциальный ключ, *Название услуги* – потенциальный ключ, *Единица измерения*, *Стоимость услуги*).
- *Материал* (*Код материала* - потенциальный ключ, *Наименование материала* – потенциальный ключ, *Стоимость материала*).
- *Оборудование* (*Инвентарный номер* – потенциальный ключ, *Название оборудования* – потенциальный ключ, *Характеристики оборудования*, *Используется в настоящее время*).
- *Заказ (Номер заказа* – потенциальный ключ, *Заказчик*, *Дата* заказа, *Дата исполнения*, *Стоимость заказа*, *Факт оплаты*).

В процессе функционирования Бюро между сущностями взаимодействуют друг с другом. В концептуальной модели взаимодействие между сущностями выражается с помощью связей, основными из которых являются следующие.

- *Используется* (*Материал* в *Заказе*) – показывает, сколько и каких материалов требуется для выполнения заказа (см. рис. 15). Бинарная связь – *многие ко многим*, так как один и тот же материал может использоваться в

различных заказах, а для выполнения одного и того же заказа может потребоваться много различных материалов. Класс принадлежности (полнота участия) каждой сущности к связи *необязательный*.



Код материала *Номер заказа*

Рис. 15. Бинарная связь о материалах, используемых в заказе.

- *Применяется* (*Оборудование* в *Заказе*) - показывает, какое оборудование необходимо для выполнения заказа.

Инвентарный номер *Номер заказа*

Бинарная связь – *многие ко многим*, так как одно и тоже оборудование может применяться в различных заказах, а в одном заказе может потребоваться различное оборудование.

- *Делает* (*Клиент* в *Заказе*) - показывает, какой клиент сделал заказ.

Бинарная связь – *один ко многим*, так как один и тот же клиент может сделать много заказов, но каждый заказ принадлежит только одному клиенту.

Название организации *Номер заказа*

При этом класс принадлежности к связи сущности З*аказ* является *обязательным*, т.к. у любого заказа должен быть заказчик, а сущности *Клиент – н обязательным*, т.к. в БД могут быть клиенты, у которых в настоящее время нет ни одного заказа.

- *Обеспечивает* – связывает сущности *Раздел* и *Сотрудник*.

Определяет, услуги какого раздела может выполнять тот или иной сотрудник.

Номер паспорта *Номер раздела*

Бинарная связь – *многие к одному*. Это означает, что каждый сотрудник имеет право оказывать услуги из одного и только одного раздела, а каждый раздел обеспечивается многими сотрудниками. Класс принадлежности для сущности *Сотрудник - н обязательный*, т.к. в Бюро могут быть сотрудники, не задействованные непосредственно в обеспечении того или иного раздела, а для

сущности *Раздел – обязательный*, т.к. каждый раздел обязательно обеспечивается хотя бы одним сотрудником.

- *Находится* – связывает сущности *Услуга* и *Раздел*. Определяет, какому разделу принадлежит услуга. Бинарная связь – *один ко многим*.

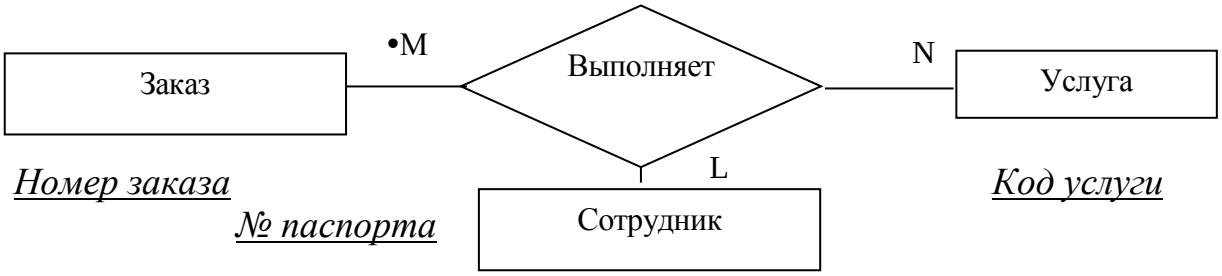
Это означает, что каждая услуга входит только в один раздел, а в каждый раздел может включаться много услуг. Класс принадлежности обеих сущностей *обязательный*, т.к. нет услуг, не входящих в какой-либо раздел, и нет разделов, не содержащих хотя бы одной услуги.

- *Требуется* – связывает сущности *Оборудование* и *Услуга*. Определяет оборудование, необходимое для выполнения услуги.

Бинарная связь *многие ко многим*. Класс принадлежности для сущности *Услуга н обязательный*, а для сущности *Оборудование – обязательный*, т.к. для выполнения одних услуг оборудование вообще не требуется, а для других его нужно много. В то же время каждое оборудование обязательно применяется в какой-либо услуге и, возможно, не в одной, а в нескольких.

- *Выполняет* – связывает сущности *Заказ, Услуга* и *Сотрудник*.

Отражает информацию о том, какие услуги в том или ином заказе выполнил тот или иной сотрудник.



Трехсторонняя (тернарная) связь *многие ко многим ко многим* позволяет хранить в БД информацию о том, что сотрудник *X* выполнял услугу *Y* в заказе *Z*. Такие характеристики связи обосновываются тем, что каждый сотрудник может выполнять много различных услуг во многих заказах, в каждом заказе могут быть задействованы разные сотрудники по разным услугам, и каждая услуга может выполняться в различных заказах разными сотрудниками.

Первичный вариант концептуальной модели данных предметной области представляем в виде диаграммы «сущностей-связей» (ER-диаграммы) (см. рис.16). При этом минимальная информация, изображаемая на диаграмме, должна содержать следующие пять элементов данных:

- *сущности*, изображаемые в виде прямоугольников произвольного размера;

- *ключи сущностей*, изображаемые в виде подчеркнутых надписей рядом с прямоугольником с любой стороны;
- *связи*, изображаемые в виде ромбов произвольного размера и соединенных прямыми линиями с соответствующими сущностями, эти линии могут быть произвольного наклона и, по возможности, без пересечений;
- *степень участия в связи* для каждой сущности, изображаемая в виде числа или буквы около соответствующего прямоугольника-сущности и линии связи;
- *полнота участия в связи* для каждой сущности, изображаемая в виде точки перед символом-степенью участия, при этом наличие точки означает полное участие в связи данной сущности, т.е. значение *обязательный*.

Каких-либо строгих правил для изображения диаграммы ER-типов не существует. В различных коммерческих системах (CASE средствах) проектирования БД используется и другая нотация для изображения ER-диаграмм, наиболее известными из них являются нотация «куриных лапок» [5] и нотация стандарта IDEF1X [9]. Однако, для обсуждения результатов проектирования с будущими пользователями БД удобнее всего представить диаграмму ER-типов в нотации, предложенной впервые П. Ченом, т.е. в виде прямоугольников и ромбов.

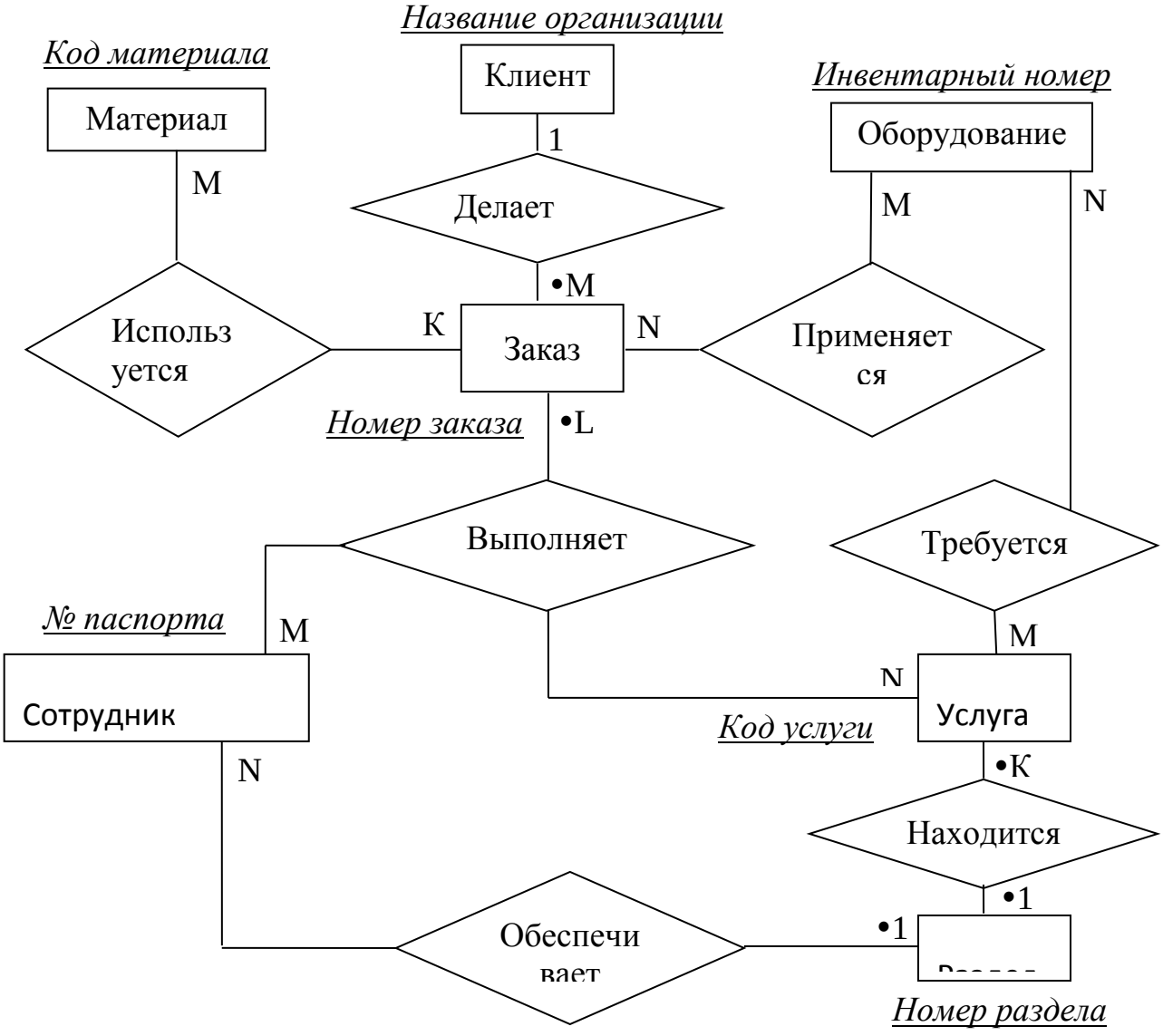


Рис.16. Первичный вариант диаграммы ER-типов предметной области.

7.3 Разработка логической модели базы данных

После того, как концептуальная модель обсуждена и одобрена пользователем БД, можно перейти к этапу логического проектирования БД. При этом исходными данными для проектирования являются диаграмма ER-типов концептуальной модели предметной области и логическая модель данных, используемая для реализации будущей БД и поддерживаемая какой-либо коммерческой СУБД (см. рис. 1).

В качестве логической модели данных выбираем реляционную модель Кодда, как наиболее распространенную в большинстве современных СУБД. Теперь следует преобразовать концептуальную модель (см рис. 16) в соответствии с требованиями реляционной модели данных.

В первую очередь необходимо устранить все связи степени более двух, в нашем примере это связь *Выполняет* степени 3. Преобразуем связь *Выполняет* в слабую сущность с названием *Статья* (заказа) и 3 дополнительных слабых связи этой сущности с тремя исходными сильными сущностями *Заказ*, *Сотрудник* и *Услуга*, назвав их соответственно: *Относится* (статья к заказу), *Исполняется* (статья сотрудником), *Включает* (статья услугу) (см. рис. 17). На диаграмме ER-типов слабые сущности и связи изображаются двойными линиями. Все слабые связи всегда имеют степень участия *один ко многим*, причем значение *много* находиться всегда на стороне слабой сущности.

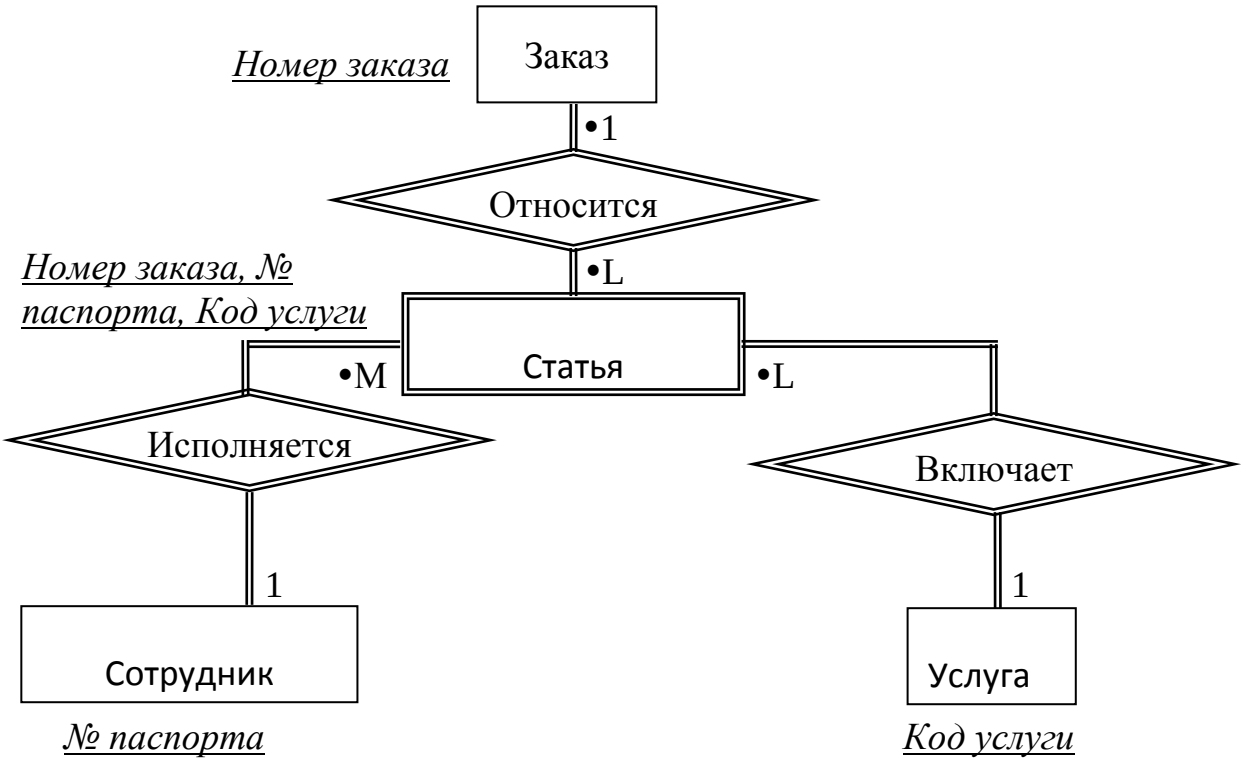


Рис. 17. Преобразование тернарной связи в слабую сущность и 3 слабые бинарные связи

Ключ слабой сущности всегда является составным и включает, как минимум, ключи соответствующих сильных сущностей. В нашем примере ключ сущности *Статья* состоит из трех атрибутов-ключей сущностей *Заказ, Сотрудник* и *Услуга* – {*Номер заказа, № паспорта, Код услуги*}. На практике этот

список может быть меньше или больше, в зависимости от степени участия в исходной связи сильных сущностей и конкретной интерпретации производной слабой сущности.

Характеристики полноты участия слабой сущности во вновь образованных слабых связях всегда имеет значение – *полное* (*обязательный*).

Далее следует преобразовать каждую бинарную связь со степенью участия *многие ко многим* в слабую сущность и две слабые связи, аналогично предыдущему случаю. В частности, для нашего примера связь *Требуется* преобразуется к виду, представленному на рисунке 18.

В результате преобразования появляется слабая сущность *Оборудование_Услуга* с составным ключом, состоящим из ключей сильных сущностей {*Инвентарный номер, Код услуги}* и две слабые связи со степенью участия *один ко многим*, причем значение *один* имеет всегда сильная сущность. Интерпретация слабой сущности *Оборудование_Услуга* состоит в том, что каждый экземпляр этой сущности представляет собой как минимум пару атрибутов, указывающих на соответствующий экземпляр сущности *Оборудование*, который используется для выполнения соответствующего экземпляра сущности *Услуга*.

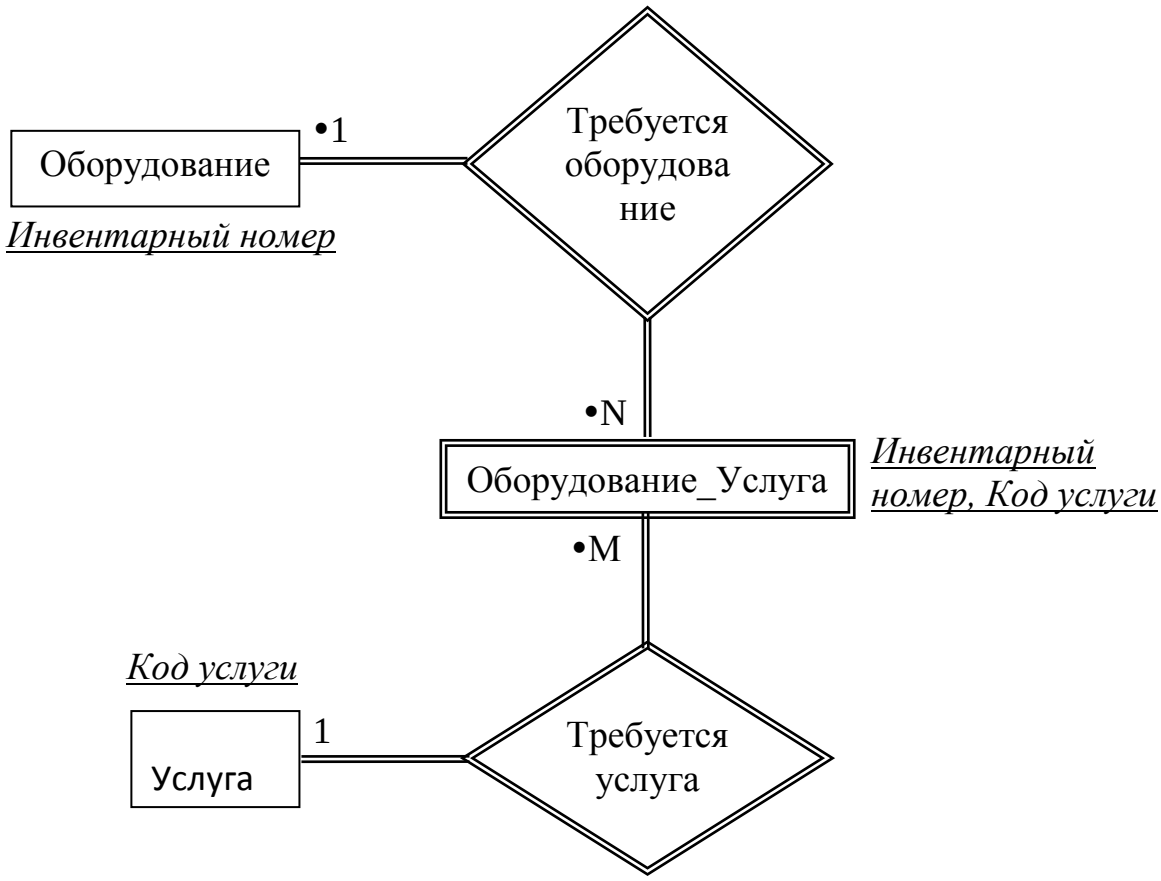


Рис. 18. Преобразование связи *многие ко многим* к реляционному виду

Отсюда следует, что класс принадлежности слабой сущности к связям ее с сильными сущностями должен быть всегда *обязательным*. Класс принадлежности сильных сущностей во вновь образованных слабых связях остается таким же, каким он был в концептуальной модели.

Замечание: тот же результат будет получен при реализации в реляционной модели данных связи *многие ко многим*, если воспользоваться правилом

преобразования диаграммы ER-типов в схему реляционной БД, минуя формально этап логического проектирования (см. правило 6, с. 27 настоящего пособия).

Таким образом, формально этап логического проектирования БД может в проекте отсутствовать, но при этом следует иметь в виду, что при выполнении преобразования диаграммы ER-типов в схему БД с помощью выше изложенных правил 1-8, мы, фактически и выполняем логическое проектирование БД с использованием реляционной логической модели данных. И в любом случае получаем логическую модель, в которой все объекты представлены как сущности (сильные или слабые) и связи *один ко многим* и *один к одному* (сильные или слабые).

Еще одним ограничением реляционной модели данных является требование нормализации всех сущностей-отношений. Это требование часто называют первой нормальной формой (1НФ). Оно заключается в том, чтобы все атрибуты в БД были атомарными с точки зрения СУБД. Атомарность означает неделимость каждого значения данных на многие значения, например не допускается иметь в БД атрибутов, представляющих собой списки значений (множественные значения), как-то: номера телефонов, оценки, дни работы и т.д. Не допускается иметь в БД и атрибуты составные, например адрес, включающий город, улицу, дом, квартиру при условии, что в процессе использования БД потребуется обращение к какой-либо части этого адреса.

Во всех этих случаях требуется на этапе логического проектирования обязательно провести нормализацию каждого отношения, т.е. привести его к 1НФ.

На следующем шаге проектирования необходимо оценить полученные отношения с точки зрения нормальных форм более высокого порядка. Причем, из пяти нормальных форм, известных в настоящее время 2НФ, 3НФ. НФБК, 4НФ и 5НФ, на практике, как правило, бывает вполне достаточно обеспечить третью нормальную форму или НФБК (усиленную 3НФ). В крайнем случае, можно вообще не заниматься вопросами приведения БД к той или иной НФ, но при этом надо помнить о том, что надежность проекта будет тем ниже, чем сложнее БД и чем большие требования предъявляются к целостности данных.

Чтобы провести проверку на уровень нормализации БД, необходимо:

- разместить все атрибуты во всех полученных сущностях-отношениях (слабых и сильных);
- выявить в каждом отношении потенциальные ключи;
- выявить все функциональные зависимости (ФЗ) между атрибутами в каждом отношении;
- сопоставить потенциальные ключи с детерминантами ФЗ (их левыми частями);
- если какой-либо детерминант отношения не совпадает ни с одним потенциальным ключом этого же отношения, то данное отношение не находится в той или иной НФ и его требуется нормализовать в соответствии с теорией нормальных форм [1].

В результате приведения БД в ту или иную НФ, как правило, появляются дополнительные отношения, которые следует добавить в логическую модель.

Таким образом, получается окончательный вариант логической модели БД, ER диаграмма которой представлена на рисунке 19.

На диаграмме не показаны ключи слабых сущностей:
Оборудование_Услуга – {*Код услуги, Инвентарный номер*},
Оборудование_Заказ – {*Инвентарный номер, Номер заказа*},
Материал_Заказ – {*Код материала, Номер заказа*}.

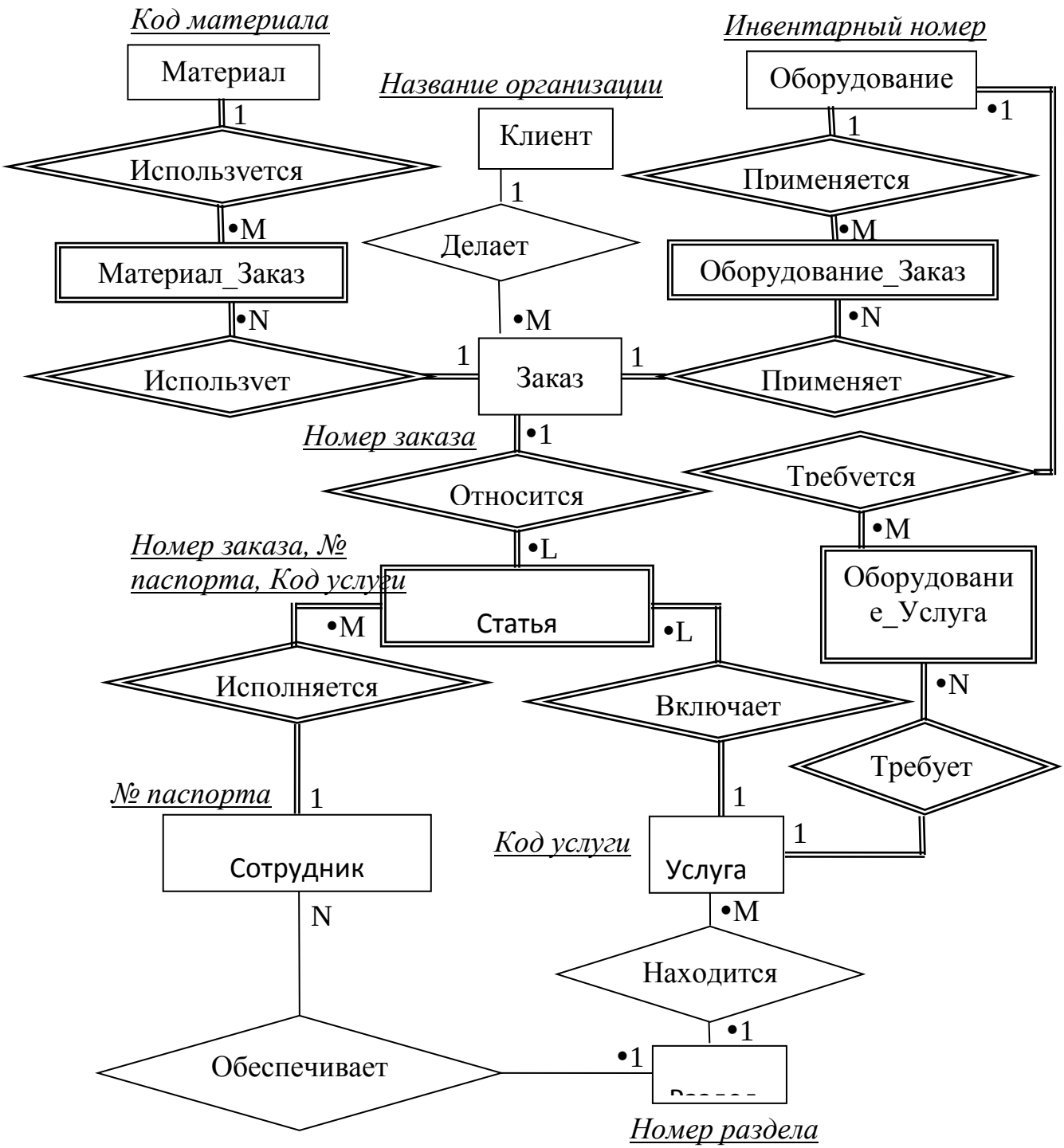


Рис. 19. Диаграмма ER-типов логической модели БД Бюро.

7.4 Разработка физической модели базы данных

На третьем этапе проекта, опираясь на логическую модель БД, полученную на предыдущем этапе, требуется, выполнить следующие основные шаги:

1) выбрать целевую СУБД;

2) спроектировать основные структуры логической модели БД, в среде целевой СУБД;
3) разработать схему связей между отношениями БД;
4) реализовать ограничения целостности данных средствами целевой СУБД;
5) реализовать требования пользователя (запросы), в соответствии с заданием на проектирование (см. п. 3 задания, приложение 1).

7.4.1 Выбор целевой СУБД

Выбор СУБД для реализации БД зависит от многих факторов:

- функциональных возможностей;
- модели данных;
- задания на проектирование;
- технической политики заказчика;
- популярности среди разработчиков.

Для разработки нашего проекта выберем систему управления базами данных Microsoft Access 2013, к достоинствам которой можно отнести:

- реляционную модель данных;
- объектно-ориентированную событийно-управляемую среду разработки приложений Visual Basic for Application (VBA);
- языки конструирования запросов SQL и QBE;
- преемственность с предыдущими и последующими версиями;
- возможность удаленного доступа к данным;
- возможность импорта/экспорта данных в другие СУБД и приложения.

В сфере информационных технологий обновление знаний происходит достаточно интенсивно. Вместе с тем, имеется тенденция преемственности концептуальных основ баз данных. К ним в первую очередь относятся реляционный подход к моделированию данных предметной области и объектно-ориентированный подход при разработке пользовательских приложений.

Следует отметить интенсивное внедрение интеграционных подходов в корпоративных базах данных, основанных на Web-технологии Internet. Простой интерфейс браузеров становится универсальным средством для работы с документами различных приложений, в том числе с базами данных Microsoft Access. СУБД Access обеспечивает публикацию баз данных в формате, доступном в сетях Internet и intranet. В Microsoft Access 2013 эти средства получили дальнейшее развитие. Теперь они позволяют конструировать в интерактивном режиме Web-страницы, предназначенные для работы с базами данных.

Особое значение имеет возможность создания экранных форм, совпадающих по структуре с реальными документами (источниками данных), для использования их при загрузке и корректировке справочных и оперативных данных, а также при просмотре информации. Чтобы эти формы позволяли выполнять задачи пользователя, требуется создать приложение БД в виде совокупности процедур, запросов и макросов.

7.4.2 Разработка схемы БД

На втором шаге физического проектирования БД требуется разработать и описать схемы для всех сущностей, полученных в логической модели данных. Схема сущности должна содержать полное описание атрибутов-характеристик этой сущности и их свойств на языке целевой СУБД, с учетом связей между сущностями и ограничений целостности, которые накладывает предметная область на сущности и их атрибуты.

Результаты разработки схемы БД лучше всего представить в виде таблицы (Табл. 1, 2). В примере приведены только основные свойства атрибутов и простейшие условия на их значение. В реальном проекте их, как правило бывает гораздо больше, в том числе маска ввода, сообщение об ошибке, индексированное поле.

Таблица 1

Схема БД Бюро для сильных сущностей

Имя отношения	Имя атрибута	Тип	Обяз	Формат	Условие	ПтК	ВК
Раздел	№Раздела	Ч	Да	Байт	1-10	+	
	Название	Т	Да	Длина – 150		+	
	Описание	М	Нет				
Услуга	КодУслуги	Т	Да	Длина – 6		+	
	№Раздела	Ч	Да	Дл. целое			+
	Название	Т	Да	Длина – 100		+	
	ЕдИзм	Т	Да	Длина – 10			
	Цена	Д	Да		>0		
Оборудование	Инвент№	Ч	Да	Дл. целое	>0	+	
	Название	Т	Да	Длина – 40		+	
	Характеристики	М	Нет				
	Использование	Л	Да	Да/Нет			
Клиент	КодКлиента	Ч	Да	Целое		+	
	НазваниеОрганизации	Т	Да	Длина– 30		+	
	Адрес	Т	Нет	Длина– 100			
	НомерТелефона	Т	Нет	Длина– 16			
	Заметки	М	Нет				
Сотрудн	НомерПаспорта	Т	Да	Длина– 14		+	

ик	Фамилия	Т	Да	Длина– 50			
	Адрес	Т	Нет	Длина– 255			
	ДомашнийТелефон	Т	Нет	Длина– 15		+	
	№Раздела	Ч	Да	Байт	1-10		+
Материал	КодМатериала	Ч	Да	Байт	>0	+	
	Наименование	Т	Да	Длина– 50		+	
	Стоимость	Дн	Да		>0		
Заказ	НомерЗаказа	Ч	Да	Целое	>0	+	
	КодКлиента	Ч	Да	Целое	>0		+
	ДатаЗаказа	Д	Да	Кр.фор.д.	<=Now		
	ДатаИсполнения	Д	Нет	Кр.фор.д.	>Now		

Таблица 2

Схема БД Бюро для слабых сущностей (связей)

Имя отношения	Имя атрибута	Тип	Обяз	Формат	Условие	ПтК	ВК
Материал_Заказ	КодЗаказа	Ч	Да	Байт	>0	+	+
	КодМатериала	Ч	Да	Байт	>0		+
	Количество	Ч	Нет	Целое	=>1		
Статья	КодЗаказа	Ч	Да	Байт	>0		+
	КодУслуги	Т	Да	Длина– 6			+
	ДатаИсполнения	Д	Нет	Целое	=>1		
	Качество	Ч	Нет	Байт	1-5		
	Номер_паспорта	Т	Да	Длина– 14			+
Оборудование_Услуга	Код_услуги	Т	Да	Длина– 6			+
	Инвент№	Ч	Да	Дл. целое			+
Оборудование_Заказ	Инвент№	Ч	Да	Дл. целое			+
	КодЗаказа	Ч	Да	Байт			+

Условные обозначения, принятые в таблицах:

- для типов полей: Т – текстовый, Ч – числовой, Д – Дата/Время, Л – логический, Дн – Денежный, М – МЕМО;

- для заголовков столбцов: Обяз – свойство обязательности значения поля, ПтК – является ли атрибут потенциальным ключом, ВК – является ли атрибут внешним ключом;
- прочих элементов: подчеркивание – имена атрибутов первичного ключа, + – значение Да.

На основе схемы базы данных, а именно, используя значения потенциальных и внешних ключей отношений, создаем схему связей между отношениями (см. рис. 20).

Чтобы построить схему связей, необходимо из окна БД СУБД Access вызвать соответствующий диалог. Далее добавить в основное окно диалога сконструированные таблицы, затем повторить процедуру формирования связей между парами таблиц, находящихся в отношении 1:1 или 1:М путем перетаскивания потенциального (первичного) ключа на соответствующий ему внешний ключ.

В результате получаем предварительный вариант схем БД, без учета ограничений целостности данных. Такая схема уже может быть использована для реальной работы с БД, однако надежность такой работы будет низкой и мало эффективной.