

**Лабораторные работы по курсу
Базы данных**

**Лабораторная работа 4
«Создание логической и физической модели базы данных»**

Москва, 2023

Оглавление

Теоретические сведения	3
1.1. Проектирование базы данных	3
1.2. Логическое моделирование. ER-модель	3
1.3. Пример ER-диаграммы	5
1.4. Физическое проектирование	6
Лабораторное задание	8
Приложение	10
Работа с программой ERwin	10

Теоретические сведения

1.1. Проектирование базы данных

Проектирование баз данных в общем виде является сложной и трудоемкой задачей. В общем случае её можно сформулировать, как выбор подходящей логической структуры для заданного массива данных, который необходимо поместить в базу данных. [1] Другими словами, необходимо выделить требуемые отношения (таблицы) и содержащиеся в них атрибуты (столбцы). По окончании проектирования созданную базу данных необходимо преобразовать в форму, которая может быть воспринята конкретной СУБД.

Проектирование базы данных можно разделить на несколько этапов:

- Концептуальное (семантическое, инфологическое) – на данном этапе происходит анализ и определение понятий предметной области
- Логическое – создание схемы данных на основе заданной модели без учета специфики СУБД
- Физическое – создание базы данных с учетом специфики СУБД

Данная лабораторная работа посвящена созданию логической и физической моделей базы данных. Логическая модель данных будет изучаться на примере создания ER-модели или модели «сущность-связь». Физическая модель данных будет представлена в виде запросов для СУБД PostgreSQL.

1.2. Логическое моделирование. ER-модель

ER-модель является одним из самых распространенных средств для представления структуры баз данных. В ней структура данных отображается графически в виде диаграммы сущностей и связей. Данная диаграмма состоит из элементов трех типов:

- Сущности
- Атрибуты
- Связи

Сущность – абстрактный объект определенного типа. Например, в базе данных Студентов вуза можно выделить сущности «Студент», «Преподаватель», «Структурное подразделение» и др.

Атрибут – свойство множества сущностей. Например, сущности «Студент», могут быть поставлены в соответствии атрибуты «Фамилия», «Имя», «Номер студенческого билета» и др.

Среди атрибутов возможно выделить подмножество, обладающее свойствами уникальности и неизбыточности. Такое подмножество называется **Потенциальным ключом**. Из него возможно выбрать один ключ, который будет идентифицировать конкретную запись в таблице. Такой ключ называется **первичным (PK, от Primary Key)**. Если в качестве ключа выбирается дополнительный атрибут, содержащий порядковый номер записи или некоторое независимое от содержимого значения, то такое ключ называется **суррогатным**. На каждый атрибут можно наложить ограничения, чтобы не допускать добавление в базу данных некорректных значений.

В общем случае, в ER-модели не указывают типы атрибутов. Однако, в данной лабораторной работе будет разрабатываться модель под СУБД PostgreSQL

Связи – соединения между двумя и большим числом сущностей. Например, между сущностями «Студент» и «Группа» возможно провести связь – «студенты, обучающиеся в группе». Наиболее распространенными являются бинарные связи, соединяющие два множества сущностей.

Существуют несколько типов связи – 1:1, 1:N, N:N. Рассмотрим их подробнее.

Связь «один к одному» (1:1) – в случае данной связи каждой записи из одной таблицы будет соответствовать одна уникальная запись в другой. Например, у каждого студента есть студенческий билет, поэтому связь между сущностями «Студент» и «Студенческий билет» будет один к одному. Связь между таблицами происходит за счет совпадения значений первичных ключей в обеих таблицах.

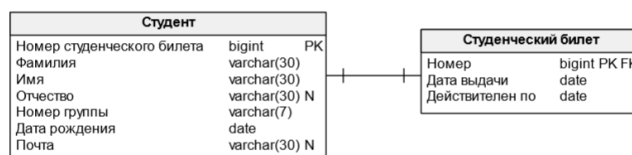


Рисунок 1 Связь 1:1

Связь «один ко многим» (1:N) – один из наиболее распространенных типов связи. В данном случае, одной записи главной таблицы можно сопоставить несколько записей подчинённой таблицы. Например, в одной группе может обучаться множество студентов. Таким образом, в данном случае связь между атрибутами будет один ко многим.

Связь между таблицами происходит за счет **внешних ключей (FK, от Foreign Key)** – атрибута отношения, значения которого полностью совпадают со значениями потенциального ключа другой таблицы. Например, атрибут «Номер группы» в таблице «Студент» является внешним ключом, т.к. он связан с первичным ключом «Номер группы» таблицы «Группа». Данный атрибут отмечен символом FK.

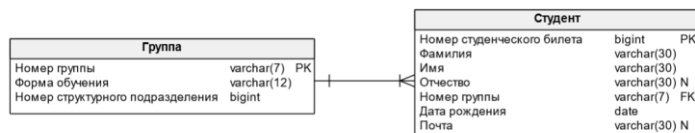


Рисунок 2 Связь 1:N

Связь «Многие ко многим» (N:N) предполагает возможность связи одного или нескольких элементов из одной таблицы с одним или несколькими элементами из другой таблицы. В случае рассматриваемого примера один преподаватель может работать в нескольких структурных подразделениях, а в одном структурном подразделении могут содержаться несколько преподавателей. Данный тип связи нереализуем в рамках физической модели, поэтому представляется в виде создания дополнительной таблицы, содержащей ключевые поля соединяемых отношений. Например, «Структурное подразделение» и «Преподаватель» соединены посредством таблицы «Трудоустройство».

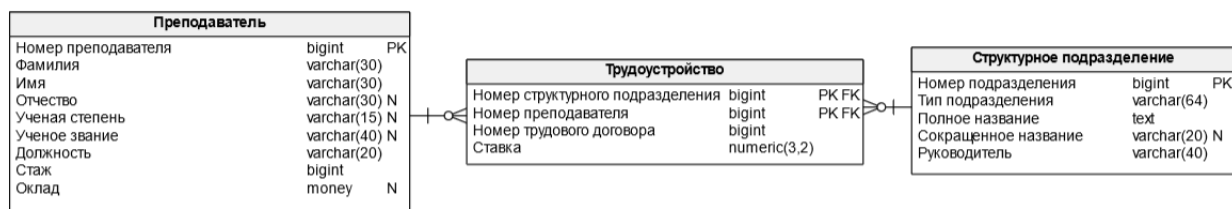


Рисунок 3 Связь N:N

Существует несколько вариантов нотаций записи ER-моделей в форме диаграмм. В данном лабораторном практикуме будет рассмотрена ER-диаграмма в нотации Гордона Эвереста.

Согласно данной нотации, **сущность** изображается в виде прямоугольника, содержащее её имя. Имя сущности должно быть уникальным в рамках одной модели. **Атрибуты** сущности записываются внутри данного прямоугольника. **Связь** изображается линией, которая связывает две сущности, участвующей в отношении. Каждая связь должна именоваться глаголом или глагольной фразой.

В зависимости от типа связи возможно различное обозначение конца линии.



Рисунок 4 Тип линии

В случае связи один к одному, на конце линии указывается вертикальная черта, в случае один ко многим указывается изображение вилки или вороньей лапки.

1.3. Пример ER-диаграммы

Рассмотрим пример разработки ER-диаграммы на основе базы данных о студентах вуза. В ней необходимо хранить информацию о студенте, группе его обучения, структурном подразделении, в котором состоит эта группа, оценках за дисциплины и преподавателей, поставивших эти оценки.

Первым делом, выделим требуемые сущности – Студент, Дисциплина, Группа, Преподаватель, Структурное подразделение. Рассмотрим сущности по отдельности.

Сущность «**Студент**» будет содержать следующие атрибуты: ФИО, Дата рождения, Номер студенческого билета, информацию о нем, Номер группы и Адрес электронной почты. Допустим, что каждого студента определяет номер его студенческого билета – он является уникальным. Поэтому, выберем его в качестве **Первичного ключа**. Добавим ограничение на формат электронной почты. Она будет представлять из себя строку вида [TEXT]@[TEXT].[DOMAIN], где TEXT может содержать любые английские буквы и числа, а домен только английские буквы.

Т.к. поля с информацией о студенческом билете зависят только от номера билета, но не от студента, то логично разделить данную сущность на две – одна будет описывать студента (человека), вторая – студенческий билет (документ).

Поэтому выделим еще одну сущность – «**Студенческий билет**» и соединим её со Студентом связью 1:1, т. е. каждому студенту будет соответствовать свой студенческий билет. В качестве атрибутов билета выберем дату выдачи и дату окончания действия.

Сущность «**Группа**» будет содержать в себе следующие атрибуты: «Номер группы» - в качестве первичного ключа, Форму обучения и Номер структурного подразделения, в котором состоит эта группа. На номер группы будет наложено ограничение – [текст]-[МВ Номер], где текст – любое буквосочетание, а номер – любые цифры. Форма обучения может быть только очной, заочной или очно-заочной.

Сущность «**Структурное подразделение**» будет содержать в себе название подразделения – полное и сокращенное, тип подразделения (кафедра, институт) и ФИО руководителя. В качестве ключа будет выбран суррогатный ключ – Номер подразделения.

Сущность «**Преподаватель**» будет содержать в себе его ФИО, должность, ученую степень (в формате [к/д].[текст].н, где текст – название отрасли наук), ученое звание, стаж и оклад. В качестве ключа выбран суррогатный ключ – Номер преподавателя.

Сущность «**Дисциплина**» будет включать в себя суррогатный ключ – Номер преподавателя, название дисциплины, зачетные единицы трудоемкости (ЗЕТ), структурное подразделение и преподаватель – внешней ключи, для связи с одноименными сущностями.

Т.к. один студент может иметь оценки по разным дисциплинам и одной дисциплине могут обучаться несколько студентов, то связь между этими сущностями будет многие ко многим. Для реализации данной связи добавим еще одну сущность, «**Результат освоения дисциплины**», содержащую в себе два ключа – Студент и Дисциплина. Данные атрибуты одновременно будут являться и внешними ключами, для связи с одноименными сущностями. Также данная сущность будет содержать в себе атрибут с оценкой за дисциплину. Оценка может принимать значения от 2 до 5 включительно.

Аналогично, связь между преподавателем и структурным подразделением будет многие ко многим. Поэтому добавим сущность «**Трудоустройство**», содержащую ключи из связанных таблиц, номер трудового договора и ставку.

Итоговая ER-диаграмма приведена на Рисунок 5.

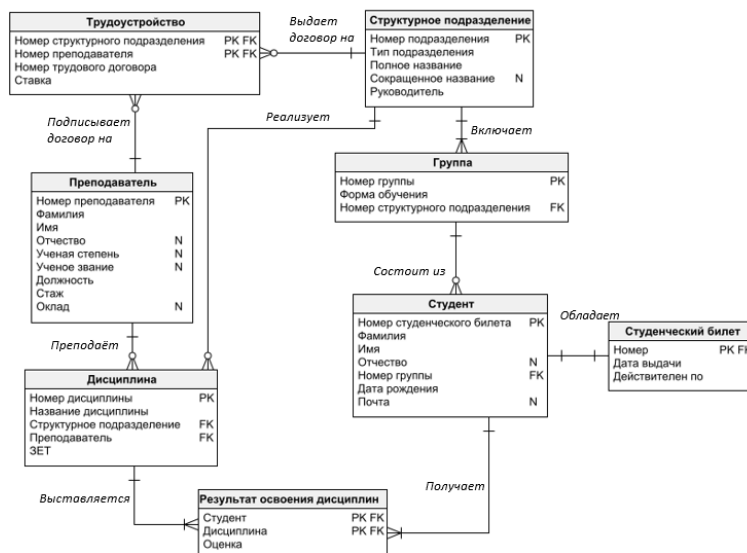


Рисунок 5 ER-диаграмма студентов

1.4. Физическое проектирование

После разработки логической модели возможно создать физическую модель, связанную с конкретной СУБД. В качестве СУБД выбрана PostgreSQL.

Создание таблиц в PostgreSQL происходит с помощью команды CREATE TABLE. Сокращенный синтаксис приведен ниже.

```
CREATE TABLE [ IF NOT EXISTS ] имя_таблицы
(
    имя_столбца тип_данных [ ограничение_столбца [ ... ] ]
    [ ограничение_таблицы ]
);
```

1.4.1. Типы данных в PostgreSQL

PostgreSQL предоставляет пользователям большой выбор встроенных типов данных. В таблице Таблица 1 приведены основные из них.

Таблица 1 Типы данных PostgreSQL

Имя	Описание
bigint	знаковое целое из 8 байт
boolean	логическое значение (true/false)
date	календарная дата (год, месяц, день)
integer	знаковое четырёхбайтное целое
json	текстовые данные JSON
money	денежная сумма
numeric [(p, s)]	вещественное число заданной точности
real	число одинарной точности с плавающей точкой (4 байта)
smallint	знаковое двухбайтное целое
serial	четырёхбайтное целое с автоувеличением
text	символьная строка переменной длины
time [(p)] [without time zone]	время суток (без часового пояса)
uuid	универсальный уникальный идентификатор
varchar(n)	строка ограниченной переменной длины

Например, для создания таблицы «Структурное подразделение» возможно выполнить следующий SQL запрос.

```
CREATE TABLE "Структурное подразделение" (
    "Номер подразделения" BIGINT,
    "Тип подразделения" VARCHAR(64),
    "Полное название" TEXT,
    "Сокращенное название" VARCHAR(20),
```

```
"Руководитель" VARCHAR(40)
);
```

1.4.2. Ограничения

Типы данных накладывают множество ограничений на данные, которые можно сохранить в таблице. Однако для многих баз данных такие ограничения слишком грубые. Например, в случае установки для значения оценки типа BIGINT, это значение должно лежать в пределах от 2 до 5. Для указания ограничений для атрибутов существуют специальные ключевые слова.

Таблица 2 Ключевые слова для установки ограничений

Ключевое слово	Описание
PRIMARY KEY	Первичный ключ
FOREIGN KEY	Внешний ключ
NOT NULL	Значение не может принимать значение NULL
NULL	Значение может принимать значение NULL
CHECK (условие)	Проверка условия
UNIQUE	Уникальность атрибута
DEFAULT	Установка значения по умолчанию

Самый простой способ установки ограничений – использование ключевого слова CHECK. В его определении возможно указать, что значение данного столбца будет удовлетворять логическому выражению (проверке истинности).

Например, укажем, что Номер подразделения может быть только положительным числом:

```
"Номер подразделения" BIGINT CHECK ("Номер подразделения" > 0)
```

Или укажем возможный диапазон оценки:

```
"Оценка" BIGINT CHECK (Оценка >=2 AND Оценка <=5)
```

Для указания того, что атрибут является первичным ключом, используются ключевые слова PRIMARY KEY. Например,

```
"Номер подразделения" BIGINT PRIMARY KEY CHECK ("Номер подразделения" > 0)
```

Аналогично можно указать, что значения атрибута могут/не могут принимать значения NULL, являются уникальными. Для этого используются слова NOT NULL, NULL, UNIQUE.

Например:

```
"Отчество" VARCHAR(30) NULL,
"Дата рождения" DATE NOT NULL,
"Почта" VARCHAR(30) UNIQUE
```

Каждому ограничению возможно присвоить отдельное имя. Для этого перед ограничением, указывается ключевое слово **CONSTRAINT** и название ограничения после него.

```
CONSTRAINT проверка_почты
CHECK ("Почта" ~* '^[A-Za-z0-9._%+@][A-Za-z0-9.-]+[.][A-Za-z]+$')
```

Для организации связей между таблицами необходимо добавить внешние ключи. Для этого в строке атрибута указывается, ключевое слово REFERENCES далее за ним идет название связываемой таблицы и в скобках указывается имя связываемого атрибута.

```
"Студент" BIGINT NOT NULL REFERENCES "Студент" ("Номер студенческого билета")
```

Все ограничения возможно выносить за пределы строки с атрибутом.

Рассмотрим пример создания таблицы «Студент»

```
CREATE TABLE "Студент" (
    "Номер студенческого билета" BIGINT NOT NULL,
    "Фамилия" VARCHAR(30) NOT NULL,
    "Имя" VARCHAR(30) NOT NULL,
    "Отчество" VARCHAR(30) NULL,
    "Номер группы" VARCHAR(7) NOT NULL,
    "Дата рождения" DATE NOT NULL,
    "Почта" VARCHAR(30) UNIQUE,
    PRIMARY KEY ("Номер студенческого билета"),
    CONSTRAINT клГруппа
        FOREIGN KEY ("Номер группы")
            REFERENCES "Группа" ("Номер группы")
            ON DELETE CASCADE,
```

```
CONSTRAINT проверка_почты
CHECK ("Почта" ~* '^[A-Za-z0-9._+%~]+@[A-Za-z0-9.-]+[.][A-Za-z]+$' )
);
```

В данном случае, все ограничения на ключи и CHECK были вынесены за пределы строк с атрибутами. Для указания внешнего ключа использовалось ключевое слово FOREIGN KEY. Для обеспечения каскадного удаления всех связанных с записью строк по ключам используются ключевые слова ON DELETE CASCADE.

В завершении работы приведем итоговую схему данных с учетом всех заданных типов.

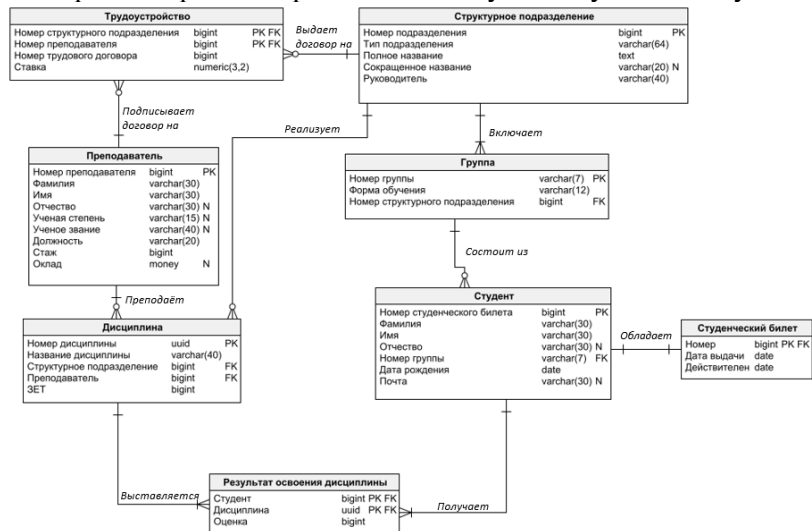


Рисунок 6 Схема данных

Лабораторное задание

Задание 1. В соответствии с вариантом разработать схему базы данных в любой доступной вам программе. В приложении к лабораторной работе приведен пример работы в программе ERwin. Разрабатываемая база данных должна содержать не менее 6 таблиц.

Задание 2. В соответствии с вариантом создать базу данных в СУБД PostgreSQL.

Варианты заданий

№	Описание предметной области
1	База данных для учета книг в домашней библиотеке.
2	База данных для хранения сведений о безработных, обратившихся в центр занятости населения, вакансиях, предлагаемых работодателями, и трудоустроенных гражданах.
3	База данных для хранения сведений о торговых операциях аптечного склада, занимающегося оптовой продажей лекарств и аптечных принадлежностей (градуслики, шприцы, бинты и т. д.) больницам и аптекам города.
4	База данных для хранения сведений о работе компании по организации грузоперевозок с описанием следующих сущностей: транспортные средства, заказы, маршруты, оплата, потери, склады.
5	База данных для хранения сведений о заселении муниципальных общежитий (комнат и проживающих).
6	База данных для хранения сведений обо всех транспортных средствах города, их владельцах, технических осмотрах транспортных средств и об угонах.
7	База данных для хранения сведений о работе столовой с описанием следующих сущностей: поставки продуктов, меню, сотрудники, план производства, касса, жалобы.
8	База данных для хранения сведений о работе поликлиники с описанием следующих сущностей: отделения, врачи, пациенты, прием, анализы, диагнозы, назначения.

9	База данных для хранения информации об оценках просмотренных кинофильмов.
10	База данных для хранения сведений об имуществе университетского городка. В состав имущества входит несколько зданий. В зданиях располагаются аудитории, кафедры, лаборатории, вычислительные центры, деканаты и т. д. Любое помещение университета относится к какому-либо подразделению.
11	База данных для хранения сведений об аренде рекламных щитов. На этих щитах может быть размещена реклама по заказу любой организации города. Срок размещения, стоимость аренды щита и стоимость изготовления самой рекламы – договорные.
12	База данных для хранения сведений об изготовлении и выдаче технических паспортов на объекты недвижимости их собственникам.
13	База данных для хранения сведений о преподавателях, ведущих занятия по различным дисциплинам в студенческих группах.
14	База данных для хранения информации об оценках просмотренных спектаклей.
15	База данных для хранения сведений о работе страховой компании с описанием следующих сущностей: страховые продукты, агенты, продажа, страховые случаи, выплаты.
16	База данных для учета сведений о студентах-дипломниках, выпускаемых кафедрами института, и их руководителях.
17	База данных для хранения сведений о партиях грузов, перевозимых судами. На судне может находиться несколько партий грузов для различных грузополучателей из различных стран и городов; у одной партии груза может быть только один отправитель и только один получатель.
18	База данных для хранения сведений о работе библиотеки с описанием следующих сущностей: сотрудники, помещения, учет книг (получение, размещение, выдача, списание, обмен), читатели.
19	База данных для хранения сведений о деятельности администратора гостиницы. В обязанности администратора гостиницы входит подбор наиболее подходящего номера для гостя, регистрация гостей, выписка отъезжающих.
20	База данных для учета сведений о продаже туристических путевок конкретным клиентам различными турфирмами.
21	База данных для хранения сведений об абитуриентах, поступающих на факультеты института, и о результатах сдачи ими вступительных экзаменов.
22	База данных для хранения сведений о процессе ремонта телевизоров, поступающих от заказчиков, мастерами телеателье.
23	База данных для хранения сведений о студентах, обучающихся на факультетах института, с учетом мест прохождения практики.
24	База данных для хранения сведений о кадровом составе кафедр института с учетом данных о детях сотрудников.
25	База данных для регистрации граждан, находящихся в санатории, с учетом распределения их по комнатам и назначения им лечебных процедур.
26	База данных для учета автомобилей, продаваемых гражданам и организациям.
27	База данных для учета заявок, поступающих на радио от слушателей, с просьбой передать музыкальные произведения.
28	База данных для учета использования аудиторий для занятий по различным дисциплинам в студенческих группах.
29	База данных для учета оплаты дополнительных занятий, проводимых преподавателями кафедр института.
30	База данных для хранения сведений о студентах, обучающихся на факультетах института, с учетом изучаемых дисциплин.

Приложение

Работа с программой ERwin

ERwin имеет два уровня представления модели – **логический и физический**.

Логический уровень — это абстрактное описание данных, на нем данные представляются так, как выглядят **в реальном мире**, и могут называться так, как они называются в реальном мире (например, на кириллице и с использованием специальных символов).

Логическая модель данных разрабатывается на основе существующих моделей данных (например, реляционной), но никак не связана с конкретной реализацией системы управления базы данных (СУБД) и прочих физических условий реализации. Она может быть построена на основе другой логической модели, например, на основе модели потоков данных или процессов.

Интерфейс ERwin. Уровни отображения модели

При создании новой логической модели с целью дальнейшего создания на ее основе модели физической необходимо установить переключатель типа модели в положение «Logical/Physical» остальные значения оставить без изменения (см. рис. 1).

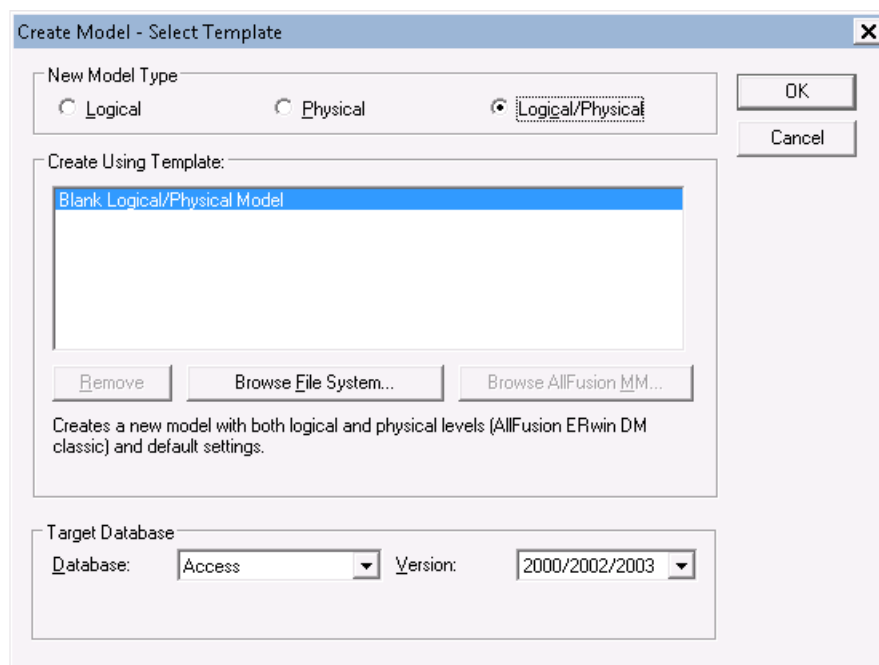


Рисунок 7 Создание новой модели

Палитра инструментов выглядит различно на разных уровнях отображения модели.

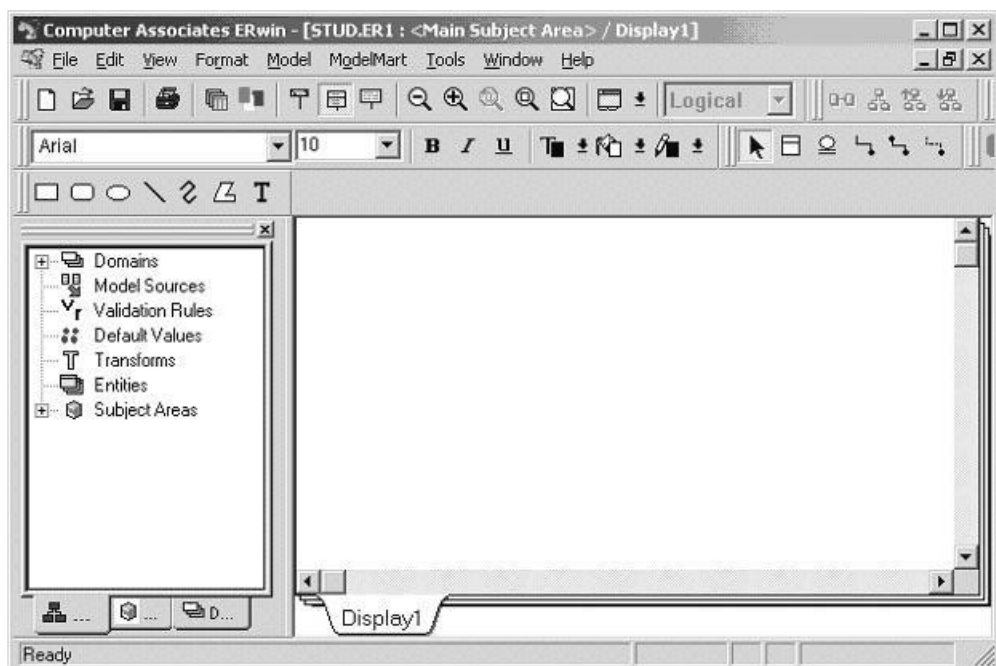


Рисунок 8 Окно отображения модели

Рассмотрим кратко основные функции ERwin по отображению модели, а также панель и палитру инструментов.

Таблица 1 Основная панель инструментов

Кнопки	Назначение кнопок
	Создание, открытие, сохранение и печать модели
	Изменение уровня просмотра модели: уровень сущностей, уровень атрибутов и уровень определений
	Изменение масштаба просмотра модели
	Переключение между областями модели - Subject Area
	Диалоги для генерации отчетов по модели
	Палитра инструментов
	Панель инструментов Font and Color Toolbar
	Панель Суперкласс – подкласс
	Панель для рисования графических объектов





Для создания типов сущностей модели и связывания их между собой используются палитра инструментов на рис. 3



Рисунок 9 Палитра инструментов

Палитра инструментов

Кнопки	Назначение кнопок	Описание
	Указатель	кнопка указателя (режим мыши) - в этом режиме можно установить фокус на каком-либо объекте модели
	Сущность	кнопка внесения сущности - для внесения сущности нужно щелкнуть левой кнопкой

		мышь по кнопке внесения сущности и один раз по свободному пространству на модели. Для редактирования сущностей или других объектов модели необходимо перейти в режим указателя
	Категория	категория, или категориальная связь, - специальный тип связи между сущностями, которая будет рассмотрена ниже. Для установления категориальной связи нужно щелкнуть левой кнопкой мыши по кнопке категории, затем один раз щелкнуть по сущности - родовому предку, затем - по сущности-потомку
	Идентифицирующая связь	связь между независимой и зависимой сущностями (более подробно описана ниже по тексту)
	Связь «Многие-ко-многим»	экземпляр одной сущности может быть связан со многими экземплярами другой сущности и наоборот (возможна только на уровне логической модели)
	Неидентифицирующая связь	связь между независимыми сущностями (более подробно описана ниже по тексту)

Сущности и атрибуты

Основные компоненты диаграммы ERWin – это сущности, атрибуты и связи.

Сущность можно определить как объект, событие или концепцию, информация о которой должна сохраняться. Сущности должны иметь наименование с четким смысловым значением, фактически это имя ее экземпляра. Например, сущность *Заказчик* с атрибутами *Номер заказчика*, *Фамилия заказчика*, *Адрес заказчика*.

Entity Editor в контекстном меню для сущности позволяет определить имя, описание, комментарии, иконку.

Для описания **атрибутов сущности** выбирается пункт Attribute Editor. Здесь можно указать имя нового атрибута и домен, который будет использоваться при определении типа колонки на уровне физической модели. Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение. Каждый атрибут должен быть определен (закладка Definition), при этом следует избегать циклических определений и производных атрибутов. Для атрибутов первичного ключа (это атрибут или группа атрибутов, идентифицирующая сущность) необходимо сделать пометку в окне выбора Primary Key.

Связи

Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой (Relationship Verb Phrases). Имя связи облегчает чтение диаграммы, например:

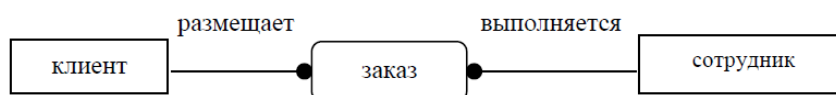


Рисунок 10 Пример диаграммы типа Сущность-Связь

По умолчанию имя связи на диаграмме не показывается. Для отображения имени следует в контекстном меню для свободного места диаграммы выбрать пункт Display Option/relationship и включить опцию Verb Phrase.

На логическом уровне можно установить идентифицирующую связь один-ко-многим, связь многие-ко-многим и неидентифицирующую связь один-ко-многим (кнопки в палитре инструментов). Тип сущности определяется ее связью с другими сущностями. Различают зависимые и независимые сущности.

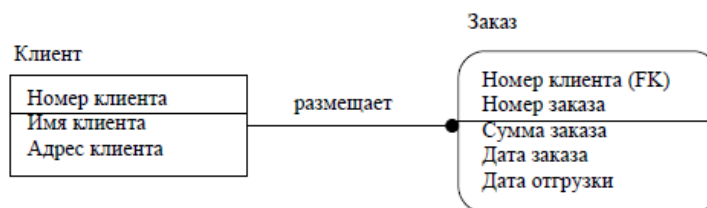
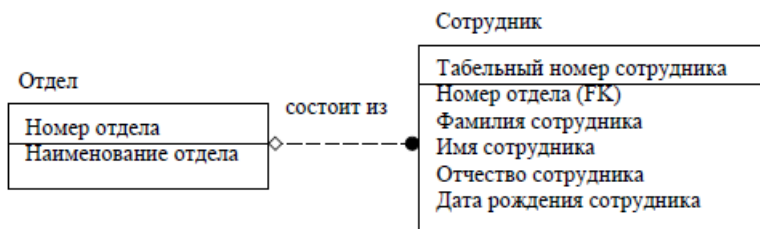


Рисунок 5. Идентифицирующая связь между независимой и зависимой сущностью

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 (в предыдущем примере сущность Заказ). Информация о заказе не может быть внесена и не имеет смысла без информации о клиенте, который ее размещает.

При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности и помечается в дочерней сущности как внешний ключ (FK). Эта операция называется миграцией атрибутов. В дальнейшем, при генерации схемы БД, атрибуты первичного ключа получают признак NOT NULL, что означает невозможность внесения записи в таблицу заказов без информации о номере клиента.



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. 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.

Во вкладке *General* меню *Relationship Editor* можно задать мощность, имя и тип связи.

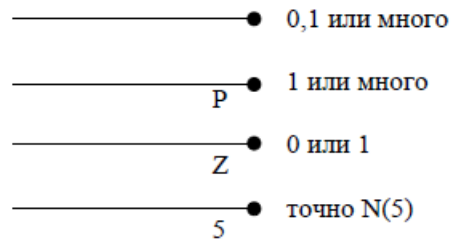
Мощность связи (Cardinality) – служит для обозначения отношения числа экземпляров дочерней родительской сущности к числу экземпляров дочерней.

Можно использовать одну из четырех типов мощности:

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

По умолчанию символ, обозначающий мощность связи, не показывается на диаграмме. Для отображения имени следует в контекстном меню для диаграммы (в месте не занятом объектами модели) выбрать пункт *Display Options/Relationship* и затем включить опцию *Cardinality*.

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.



Связь многие-ко-многим возможна только на логическом уровне. При переходе к физическому уровню Egwin автоматически преобразует связь многие-ко-многим, добавляя новую таблицу и устанавливая две новые связи один-ко-многим от старых к новой таблице. Имя новой таблицы присваивается автоматически как «Имя1_Имя2».