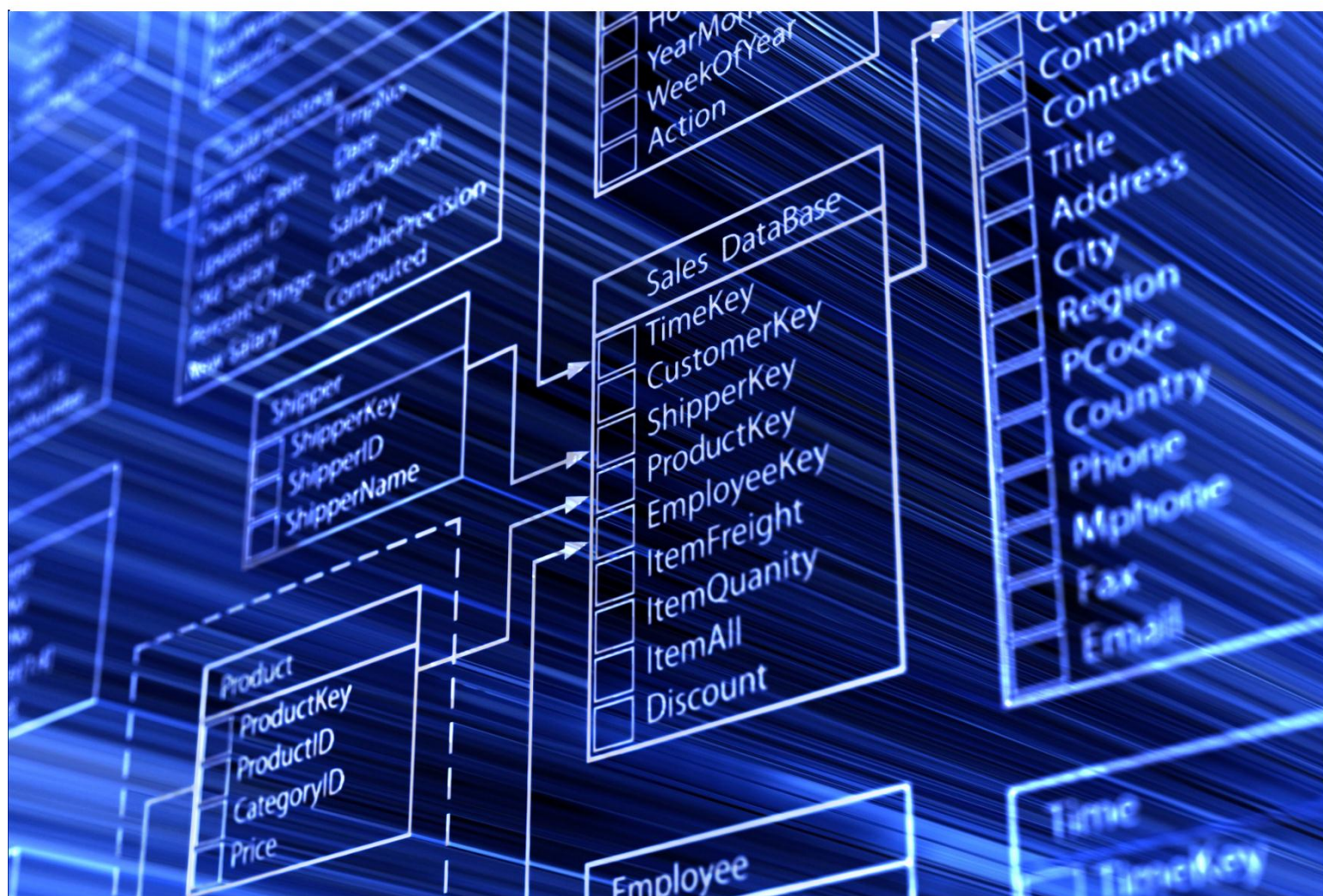


Э.Г. Галиаскаров, А.Ю. Крылов

Проектирование баз данных

Лабораторный практикум



Министерство образования и науки Российской Федерации
Ивановский государственный химико-технологический университет

Э.Г. ГАЛИАСКАРОВ, А.Ю. КРЫЛОВ

ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Иваново 2012

УДК 004.652

Галиаскаров, Э.Г.

Проектирование баз данных: лабораторный практикум / Э.Г. Галиаскаров, А.Ю. Крылов; Иван. гос. хим.-технол. ун-т.- Иваново, 2012.- 96 с.

Лабораторный практикум ставит себе целью познакомить студентов с основными положениями теории создания реляционных баз данных, привить навыки моделирования данных, проектирования баз данных и управления данными с использованием языка запросов SQL. В практикуме подробно рассмотрены вопросы использования современных средств проектирования данных. Представлен широкий набор примеров реализации запросов, триггеров, хранимых процедур. Предложен список задач на освоение SQL, отличающийся разнообразием и сложностью, от простых до очень сложных, что позволяет развить хорошие навыки использования SQL. Даны варианты заданий и рекомендации по выполнению курсового проекта по проектированию баз данных.

Практикум предназначен для студентов направления «Информационные системы и технологии», занимающихся по дисциплине «Управление данными» и может быть полезен при дипломном проектировании.

Печатается по решению редакционно-издательского совета Ивановского государственного химико-технологического университета

Рецензенты:

доктор технических наук, профессор А.Н. Лабутин (Ивановский государственный химико-технологический университет);
доктор технических наук, профессор Н.А. Коробов (Ивановская государственная текстильная академия).

© Галиаскаров Э.Г.,
Крылов А.Ю., 2012

© ФБГОУ ВПО «Ивановский
государственный химико-
технологический университет», 2012

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	5
ЛАБОРАТОРНАЯ РАБОТА №1. ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ МОДЕЛИ	7
Постановка задачи.....	7
Построение модели «Сущность-Связь».....	8
Построение модели, основанной на ключах	14
Полноатрибутивная модель данных.....	18
Контрольные вопросы и задания	20
ЛАБОРАТОРНАЯ РАБОТА №2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ	22
Преобразование полноатрибутивной логической модели данных в физическую для Microsoft SQL Server 2008 R2	22
Редактирование имен и типов данных	25
Переход к суррогатным ключам, индексирование	26
Определение процедур обеспечения ссылочной целостности	28
Создание представлений	29
Создание пользовательских триггеров	33
Создание хранимых процедур	38
Контрольные вопросы и задания	40
ЛАБОРАТОРНАЯ РАБОТА №3. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ БАЗЫ ДАННЫХ	41
Создание базы данных	41
Преобразование физической модели данных в базу данных SQL Server	42
Загрузка данных в базу данных	43
Создание, модифицирование, удаление таблиц и ограничений	45
Тестирование. Проверка исполнения базовых требований	47

Контрольные вопросы и задания	52
ПРИЛОЖЕНИЕ I. СКРИПТ ДЛЯ ГЕНЕРАЦИИ СХЕМЫ БАЗЫ ДАННЫХ	54
ПРИЛОЖЕНИЕ II. ЗАДАЧИ ДЛЯ САМОСТОЯТЕЛЬНОГО ИЗУЧЕНИЯ ЯЗЫКА SQL	61
Введение.....	61
Схема «Компьютерная фирма».....	62
Схема «Фирма вторсырья».....	63
Схема «Корабли».....	64
Схема «Аэрофлот»	65
Схема «Окраска»	66
Варианты заданий	67
ПРИЛОЖЕНИЕ III. РЕКОМЕНДАЦИИ К ВЫПОЛНЕНИЮ КУРСОВЫХ ПРОЕКТОВ	72
Требования к выполнению курсового проекта	72
Этапы выполнения курсового проекта	74
Варианты заданий на курсовое проектирование	74
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	96

ВВЕДЕНИЕ

Лабораторный практикум предназначен для студентов специальности «Информационные системы и технологии», занимающихся по дисциплине «Управление данными».

Целями данного лабораторного практикума являются:

- закрепление теоретических знаний и приобретение практических навыков анализа и моделирования предметной области;
- ознакомление с работой специализированных CASE-средств и получение навыков работы в них;
- изучение подхода к обработке данных на основе применения структурированного языка запросов SQL.

Лабораторный практикум предполагает последовательное выполнение студентами трех циклов работ: моделирование, создание и работа с базой данных; разработка приложения к базам данных средствами быстрой разработки приложений; выполнение курсового проекта.

В первом цикле лабораторных работ, представленных первой частью практикума, студенты приобретают навыки анализа и моделирования предметной области в среде CA ERwin Data Modeler, создания базы данных средствами MS SQL Server, работы с данными средствами языка запросов SQL.

В первой работе студент должен изучить предложенное задание, провести формализованный анализ по предложенной методике и построить инфологическую модель данных предметной области. Процесс построения инфологической модели разбит на три этапа. На первом этапе выделяются основные понятия предметной области и связи между ними. Здесь важно выделить сущности, дать им однозначно трактуемое наименование, определить характер и тип связей между сущностями. На втором этапе выделяются такие свойства каждой сущности, которые однозначно ее определяют, идентифицируют. Наконец, на последнем этапе определяются неключевые атрибуты каждой сущности, важные с точки зрения конкретной предметной области, диапазоны и типы значений этих атрибутов, ограничения на значения этих атрибутов, связанные с бизнес-правилами предметной области.

Вторая работа посвящена преобразованию инфологической модели в физическую реляционную модель данных, проектированию дополнительных структур в виде представлений, хранимых процедур, триггеров правил валидации и установки значений по умолчанию.

Третья работа заключается в создании базы данных, переносе физической модели, вводе различных тестовых данных и тестировании полученной базы с целью выявления ошибок проектирования и реализации. Студенты изучают язык запросов SQL, учатся получать различную информацию из базы данных,

осуществлять их модификацию, проводить перепроектирование базы данных при необходимости.

Каждая лабораторная работа содержит задачи, которые следует решить при ее выполнении, подробные примеры создания моделей и работы с ними, перечень заданий для самостоятельной работы, список вопросов для самоконтроля знаний.

Для более глубокого закрепления материала и получения разнообразных образцов использования языка запросов SQL студентам предлагается большой список разных по уровню сложности задач в различных предметных областях. Особенностью этих заданий является требование реализовать решение задачи в виде одного единственного запроса

В приложении представлены варианты заданий на курсовое проектирование по дисциплине и даны рекомендации по его выполнению. Изложение работ организовано таким образом, чтобы помочь успешной работе студентов над курсовым проектом по дисциплине.

ЛАБОРАТОРНАЯ РАБОТА №1. ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ МОДЕЛИ

Цель: получить навыки проведения анализа предметной области и построения инфологической модели в стандарте IDEF1x;
получить навыки работы с использованием CASE-средств.

Анализ предметной области для построения модели данных обычно состоит из следующих этапов [1]:

1. **Построение модели «Сущность-Связь».** На этом этапе основное внимание уделяется обнаружению и выделению сущностей предметной области и установлению связей между ними.

2. **Построение модели, основанной на ключах.** На этом этапе определяются атрибуты или наборы атрибутов, однозначно идентифицирующих каждый экземпляр сущности, выделенной на предыдущем этапе.

3. **Построение полноатрибутивной модели.** Заключительным этапом является определение дополнительных характеристик сущностей, важных с точки зрения предметной области.

Постановка задачи

View Ridge – небольшая художественная галерея, которая продает произведения мирового искусства, включая литографии (способ печати, при котором краска под давлением переносится с плоской печатной формы на бумагу), оригинальную живопись и фотографии [2]. Все литографии и фотографии подписаны и пронумерованы, и цена произведений находится в пределах от 30 тыс. до 1,5 млн. руб. Галерея View Ridge занимается продажами уже в течение 30 лет. У галереи есть бессменный владелец, три продавца и два рабочих, которые изготавливают рамы, развешивают работы в галерее и готовят их к отправке клиенту.

Галерея View Ridge занимается исключительно творчеством европейских и североамериканских художников. Она проводит выставки и другие мероприятия для привлечения клиентов. Произведения искусства выставляются также в офисах местных компаний, ресторанах и других общественных местах. Все продаваемые произведения являются собственностью View Ridge, комиссионной торговлей галерея не занимается.

Когда галерея покупает произведение, сведения о нем, его авторе, дате и стоимости приобретения записываются в базу данных. В отдельных случаях галерея может выкупить произведение у клиента и вновь выставить его на

продажу, так что одно и то же произведение может появляться в галерее неоднократно. При повторном приобретении информация о работе и ее авторе не вводится заново: записывается только дата и стоимость последнего приобретения. Когда работа продается, записывается дата совершения сделки, уплаченная сумма и сведения о покупателе.

Данные о предыдущих продажах необходимы продавцам для того, чтобы они могли уделять больше времени наиболее активным покупателям. Иногда эти записи используются для определения местонахождения ранее проданных произведений.

Для маркетинговых целей требуется, чтобы приложение базы данных выдавало список всех произведений, которые когда-либо появлялись в галерее, и их авторов. Владелец хотел бы также иметь возможность определять, насколько быстро продаются произведения каждого из художников и какова прибыль от их продаж. Наконец, приложение должно отображать список работ, имеющихся в наличии, на веб-странице, к которой потенциальные клиенты могли бы обращаться через интернет.

Требования к приложению для галереи View Ridge приведены ниже. Прежде всего, владелец и продавцы желают вести учет личных данных клиентов – их имен, адресов, номеров телефонов, и адресов электронной почты. Кроме того, они хотят знать, творчеством каких художников интересуется тот или иной клиент. Эта информация позволяет продавцам определять, кого следует извещать о поступлении новой работы, а также организовывать персональное общение с клиентом в устной или письменной форме.

Перечень требований к приложению для галереи View Ridge:

- вести учет покупателей и их художественных интересов;
- отслеживать приобретения, которые делает галерея;
- отслеживать покупки клиентов;
- вести список художников и произведений, когда-либо появлявшихся в галерее;
- генерировать отчет о том, насколько интенсивно продаются картины каждого художника;
- отображать на веб-странице список произведений, выставленных на продажу.

Построение модели «Сущность-Связь»

Определение сущностей

Анализируя предметную область, описанную в постановке задачи можно выделить 4 сущности: **Клиент**, **Художник**, **Работа** и **Транзакция**. Кратко опишем каждую из них.

Работа – произведение изобразительного искусства, включая литографию, живопись или фотографию, созданное каким-либо художником и участвующее в транзакции в качестве объекта купли/продажи.

Художник – лицо, создавшее работу и указываемое в качестве ее автора.

Транзакция – процесс купли/продажи работы галереей, должным образом зафиксированный и оформленный.

Клиент – лицо, заинтересованное в покупке работы в галерее и участвующее в транзакции в качестве покупателя.

Определение связей

Следующим этапом анализа предметной области является процесс определения устойчивых связей между сущностями. В нашем случае количество сущностей не велико, поэтому достаточно построить матрицу связей и проверить каждую пару сущностей на наличие связей.

	Работа	Художник	Транзакция	Клиент
Работа	Нет	Да	Да	Нет
Художник	Да	Нет	Нет	Да
Транзакция	Да	Нет	Нет	Да
Клиент	Нет	Да	Да	Да

Поясним результаты выполненного анализа.

Из постановки задачи нам ничего неизвестно о наличии каких-либо отношений между различными работами, поэтому в матрице связей мы поставили слово «нет».

Каждая работа имеет авторство, т.е. создана каким-либо художником. В матрице связи мы отметили факт наличия связи между работой и художником словом «да».

Работа, как следует из предметной области, является предметом купли/продажи, что явно указывает на наличие связи между работой и транзакцией. Ставим «да».

Клиент приобретает работу в ходе транзакции, поэтому прямой связи между работой и клиентом нет.

Рассуждая аналогичным образом, заполняем каждую строчку нашей матрицы. Обратите внимание, что мы указали на наличие связи между клиентом и художником. Это напрямую следует из постановки задачи:

работники галереи хотят знать, творчеством каких художников интересуется тот или иной клиент

Рассмотрим полученные в ходе анализа связи между сущностями:

- **Художник – Работа:** один и тот же художник может написать несколько работ, а каждая работа должна иметь одного и только одного художника. Получаем связь типа «один-ко-многим», идентифицирующая, т.е. у работы всегда есть автор-художник.
- **Клиент – Транзакция:** один и тот же клиент может совершать множество покупок в галерее, но в каждой такой транзакции может участвовать только один клиент. Получаем связь типа «один-ко-многим», неидентифицирующая.
- **Работа – Транзакция:** согласно требованиям одна и та же работа может появляться в галерее несколько раз, а в ходе сделки (транзакции) должна быть продана только одна работа. Связь типа «один-ко-многим», идентифицирующая, поскольку сделка не имеет смысла, если в ней не участвует предмет сделки – работа художника.
- **Клиент – Художник:** один и тот же клиент может интересоваться творчеством множества различных художников. В то же время творчеством одного и того же художника могут интересоваться различные клиенты. Связь типа «многие-ко-многим», согласно стандарту IDEF1x, неспецифическая.

Создание новой модели

Для построения графической модели будем использовать **CA ERwin Data Modeler** [3]. Это передовой программный продукт, реализующий средства CASE-технологий и позволяющий проводить описание, анализ и моделирование данных, поддерживающий нотацию IDEF1x, обеспечивающий генерацию разработанной схемы данных в различные СУБД.

Для создания новой модели, после запуска программы, необходимо:

- выбрать меню **“File”** → **“New”**;
- выбрать тип создаваемой модели: **Logical**. На этом этапе работы нас интересует только логическая модель данных, не зависящая от конкретной СУБД и ее возможностей;
- по завершению настроек новой модели нажать **«OK»**.

CA ERwin Data Modeler поддерживает две нотации: стандарт IDEF1x (Integration DEFinition for Infotmation Modeling), которую мы и будем использовать в дальнейшем для описания нашей модели данных, и стандарт информационного конструирования (Information Engineering). По умолчанию используется нотация IDEF1x. Выбор нотации доступен в меню **“Model”** → **“Model Properties”** → **“Notation”** (рис. 1.1).

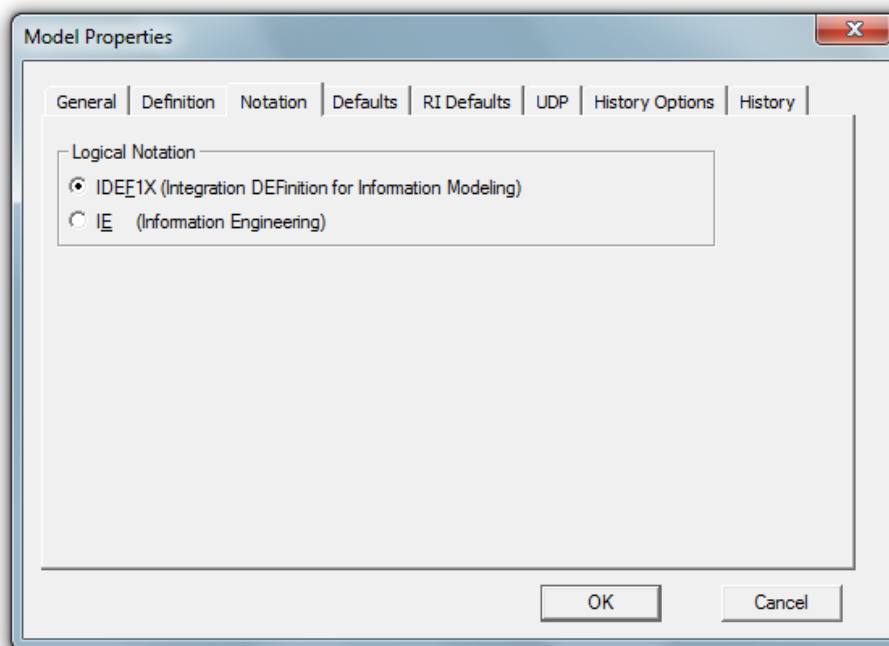


Рис. 1.1. Выбор нотации, используемой для визуального представления модели

В рабочем пространстве CA ERwin Data Modeler все необходимые для визуального моделирования данных элементы находятся на ToolBox-е (рис. 1.2), что позволяет быстро и удобно перемещать требуемый элемент в создаваемую модель – это сущности, связи и некоторые другие элементы нотации IDEF1x.



Рис. 1.2. Графические элементы нотации IDEF1x

Одна и та же модель может быть отображена на различных экранах (Display) с различных точек зрения, скрывая или, наоборот, разворачивая различные детали модели. Как было сказано вначале при построении модели проходят три уровня представления, которые следует разместить на трех разных экранах. Один экран у нас уже есть по умолчанию **“Display1”**, переименуем его и добавим еще два. Для этого:

- в обозревателе модели найдите и выделите пункт **“Subject Areas”** → **“Main Subject Area”** → **“Stored Displays”** → **“Display1”**;
- нажмите клавишу **F2** и измените название на **“ER view”**;
- выполните команду **“Stored Displays”** → **“New”** и назовите следующий экран **“PK view”** (primary key view);
- повторите предыдущее действие, но дайте такое название новому экрану **“Full attribute view”**.

Зададим режим отображения экрана **“ER view”**. Для этого

- нажмем правой кнопкой мыши в любом пустом месте области построения модели экрана «**ER view**»;
- в открывшемся меню, изображенном на рис. 1.3, выберем пункт «**Display level**»;
- выберем режим «**Entity**», чтобы отобразить лишь имена сущностей и связи.

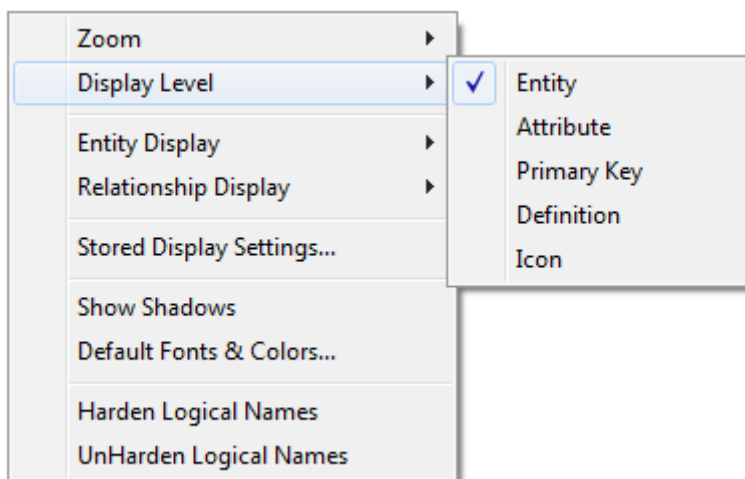

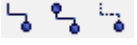


Рис. 1.3. Меню выбора режима отображения модели данных

Для создания новой сущности:

- на панели элементов (рис. 1.2) выберите пункт «**Entity**» , щелкнув по нему левой кнопкой мыши и отпустив;
- переместите курсор мыши в область построения модели (он будет иметь особый вид) и повторите щелчок левой кнопки мыши;
- введите имя сущности, например: «**Работа**»;
- аналогично разместим в области построения модели сущности «**Клиент**», «**Художник**» и «**Транзакция**».

Теперь необходимо указать связи между сущностями. Для этого выберем соответствующую связь на панели элементов нотации  и соединим ею две сущности, участвующие в этой связи. *Делается это следующим образом: выбрав связь, нажать левой кнопкой мыши на первую сущность, участвующую в связи, после чего нажать на вторую сущность.*

Действуя согласно рекомендациям, получите модель в соответствии с изображением на рис. 1.4.

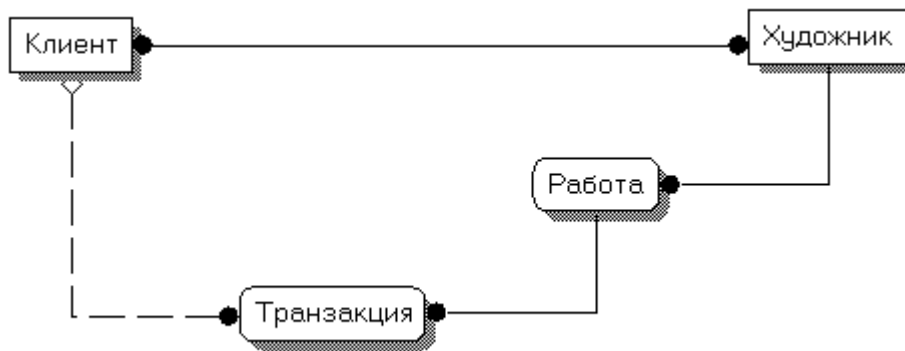


Рис. 1.4. Модель «сущность-связь». Первое приближение

Отредактируем полученные связи: для этого два раза щелкнем левой кнопкой мыши на выбранной связи. Открывшееся окно настроек связи изображено на рис. 1.5.

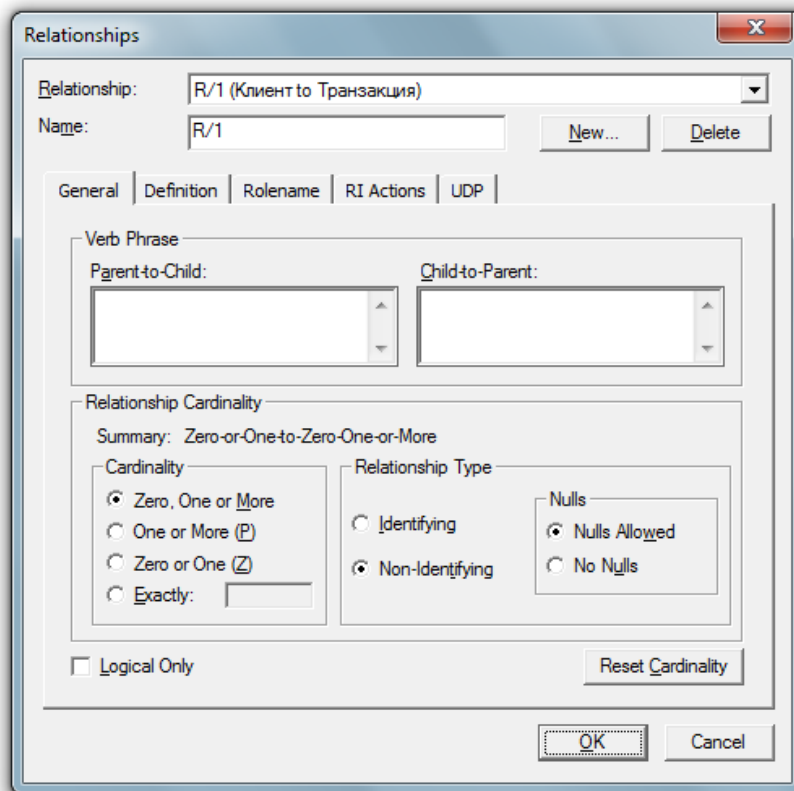


Рис. 1.5. Окно редактирования свойств связи

Здесь указываем кардинальность (**cardinality**), тип связи, обязательность родительской сущности в связи (**nulls**), имя (**Verb Phrase**) описание связи и некоторые другие ее свойства. Необязательность родительской сущности в связи изображается полым ромбом (рис. 1.4), а обязательность его отсутствием.

Например, для связи «Клиент - Транзакция» введем имя связи: «Приобретает», кардинальность оставим по умолчанию, тип: **Неидентифицирующая**, обязательность родительской сущности в связи: **Nulls**

Allowed, т.е. клиент может быть и потенциальным, не имеет еще никаких сделок с галереями.

Аналогичные действия сделаем и для остальных связей, руководствуясь результатами проведенного ранее анализа и моделью, изображенной на рис. 1.6.

- ✓ **ВНИМАНИЕ:** чтобы увидеть имена связей и значения кардинальности вызовите контекстное меню щелчком правой кнопки мыши в области модели, выберите пункт “**Relationship Displays**” и отметьте пункты **Verb Phrase** и **Cardinality**.

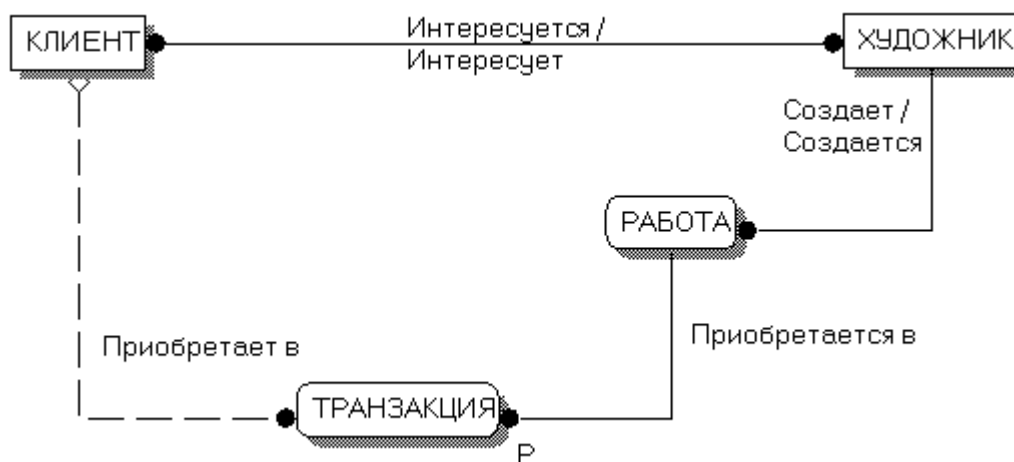


Рис. 1.6. Модель «сущность-связь». Конечный результат

Построение модели, основанной на ключах

Выделение атрибутов

Исходя из анализа предметной области, определим атрибуты, характеризующие каждую из выделенных ранее сущностей:

– Клиент:

- **Имя** – используется для полного (традиционного или официального) именованя клиента;
- **E-mail** – электронный адрес почты клиента, идентификатор;
- **Код города** – это короткий алфавитный или цифровой географический код, разработанный для представления стран и зависимых территорий в обработке данных и коммуникациях, в нашем случае в номере телефона;
- **Номер дома** – идентификатор здания, уникальный в некоторой окрестности (улицы, квартала, района), являющийся частью адреса;

- **Улица** – элемент дорожной инфраструктуры, определяется названием, часть адреса;
 - **Город** – населенный пункт, определяется названием, часть адреса;
 - **Регион** – административно-территориальная единица многих государств, часть адреса;
 - **Почтовый индекс** – последовательность букв или цифр, добавляемая к почтовому адресу с целью облегчения сортировки корреспонденции, в том числе автоматической, часть адреса;
 - **Страна** – территория, имеющая определенные границы, пользующаяся государственным суверенитетом или находящаяся под властью другого государства (колонии, подопечные территории), часть адреса;
 - **Телефон** – номер домашнего или контактного телефона.
- **Художник:**
- **Имя** – общепринятое полное имя художника;
 - **Национальность** – национальность художника (учитываются только европейские и североамериканские художники);
 - **Год рождения** – год рождения художника;
 - **Год смерти** – год смерти художника.
- **Работа:**
- **Название** – общепринятое наименование художественного произведения;
 - **Копия** – буквенно-цифровой код;
 - **Описание** – текст в произвольной форме, содержащий дополнительные сведения о работе.
- **Транзакция:**
- **Дата приобретения** – дата появления (в том числе и повторного) работы, оригинала или копии, в галереи;
 - **Цена приобретения** – стоимость работы на момент приобретения;
 - **Дата продажи** – дата продажи работы клиенту;
 - **Цена продажи** – стоимость продажи работы клиенту;
 - **Заявленная цена** – начальная стоимость продажи работы.

Определение идентификаторов

В качестве идентификаторов сущностей требуется выбирать такие атрибуты из имеющихся, значение которых будет уникальным для каждого экземпляра сущности:

- **Клиент: E-mail** – уникальность этого атрибута будет гарантироваться сервисами электронной почты в интернете;
- **Художник: Имя** – совпадение имен художников в нашем случае исключено;
- **Работа: Название + Копия** - в случае литографий и фотографий произведение может существовать в нескольких экземплярах. У работы всегда есть автор, потому она также идентифицируется через **Имя** художника;
- **Транзакция: Дата приобретения**, а также **Название + Копия** работы – такое сочетание отвечает требованию неоднократного приобретения галереей одной и той же работы. Поэтому чтобы различать разные сделки с одной и той же работой, мы используем атрибут «**Дата приобретения**».

Построение модели

Активируем ранее созданный экран “**PK view**”. Установим режим отображения “**Display Level**” → “**Primary key**” (рис. 1.3). Отредактируем модель согласно рис. 1.6. Активируйте, если необходимо, отображение имен связей и кардинальности, как это было сделано выше.

- ✓ **ВНИМАНИЕ:** для предотвращения автоматической миграции ключей вызовите контекстное меню щелчком правой кнопки мыши в области модели, выберите пункт “**Entity Displays**” и снимите флажок с пункта **Show Migrated Attributes**.

Для того чтобы добавить атрибуты-идентификаторы необходимо:

- вызвать окно редактирования атрибутов сущности двойным щелчком на изображении сущности (рис. 1.7);
- нажать кнопку «**New**», выбрать тип базового домена, задать имя атрибута;
- нажать «**OK**», сохранив введенные значения;
- для того чтобы сделать атрибут первичным ключом, выставите флаг «**Primary Key**».

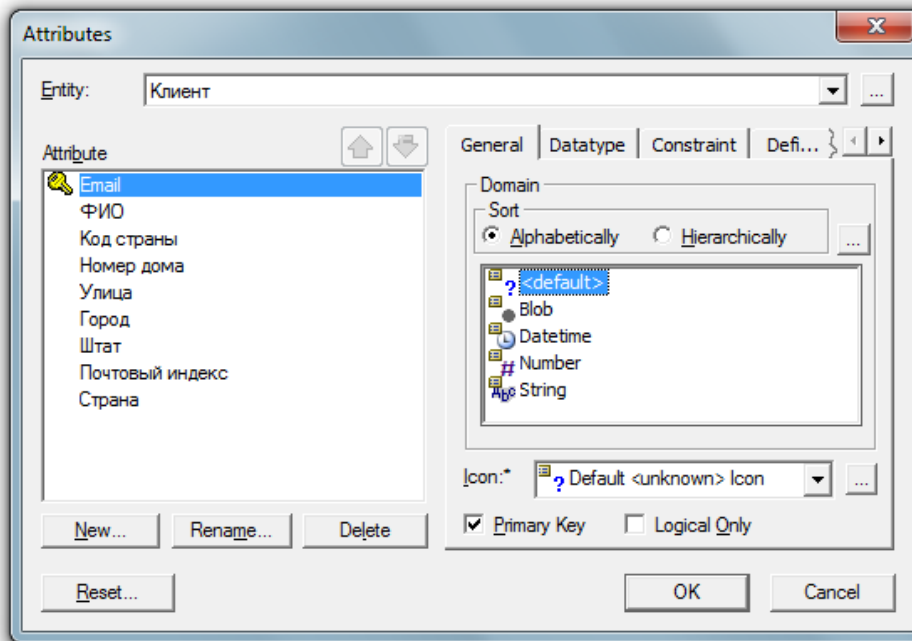


Рис. 1.7. Окно редактирования атрибутов сущности

Введите атрибуты-идентификаторы для каждой сущности, руководствуясь следующей таблицей.

Имя сущности	Название атрибута-идентификатора	Тип базового домена
Клиент	E-mail	String
Художник	Имя	String
Работа	Название Копия	String Number
Транзакция	Дата приобретения	Datetime

В результате должны получить модель, изображенную на рис. 1.8.

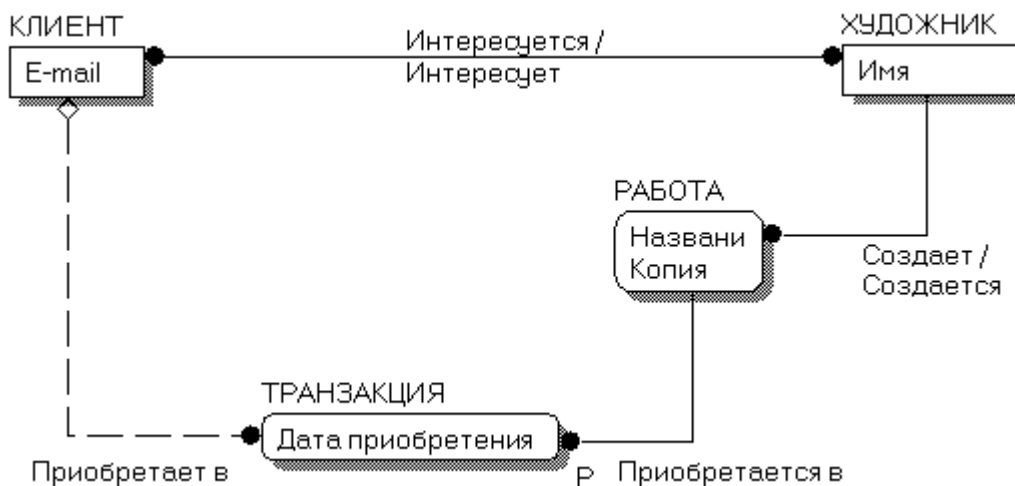


Рис. 1.8. Модель, основанная на ключах

Полноатрибутивная модель данных

Полноатрибутивная модель является конечным результатом создания инфологической модели данных. Она отличается от модели, основанной на ключах, тем, что содержит дополнительные неключевые атрибуты, которые необходимы для хранения важной для рассматриваемой предметной области информации.

Активируем ранее созданный экран **“Full Attribute view”**. Установим режим отображения **“Display Level”** → **“Attribute”** (рис. 1.3). Отредактируем модель согласно рис. 1.6 или рис. 1.8. Активируйте, если необходимо, отображение имен связей и кардинальности, как это было сделано выше. Для предотвращения автоматической миграции ключей снимите флажок с пункта **Show Migrated Attributes**.

Введите оставшиеся атрибуты для каждой сущности, руководствуясь следующей таблицей.

Имя сущности	Название атрибута-идентификатора	Тип базового домена
Клиент	Имя	String
	Код города	String
	Номер дома	String
	Улица	String
	Город	String
	Регион	String
	Почтовый индекс	String
	Страна	String
	Телефон	String
Художник	Национальность	String
	Год рождения	Number
	Год смерти	Number
Работа	Описание	String
Транзакция	Цена приобретения	Number
	Дата продажи	Datetime
	Цена продажи	Number
	Заявленная цена	Number

Полученная полноатрибутивная модель данных изображена на рис. 1.9.

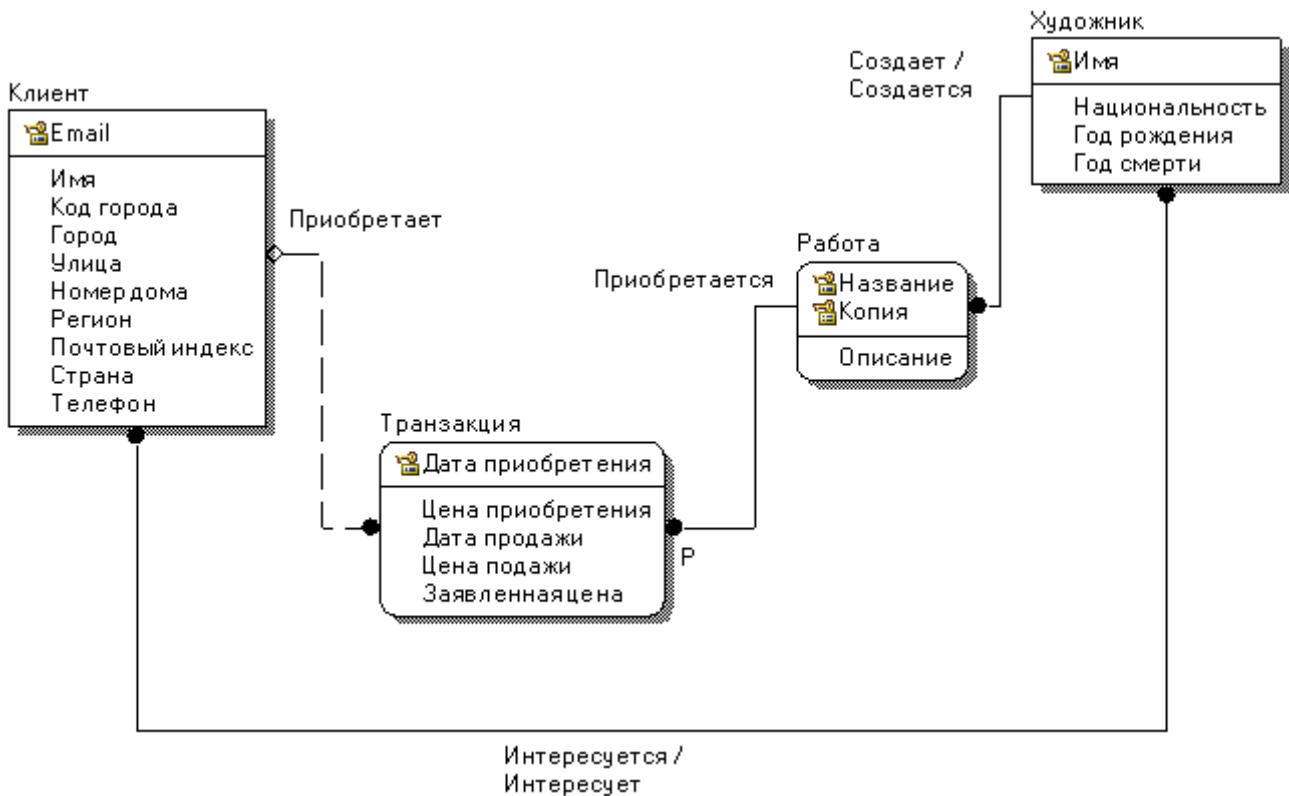



Рис. 1.9. Полноатрибутивная модель данных галереи View Ridge

Как известно,

галерея View Ridge работает только с произведениями европейских и североамериканских художников.


Чтобы учесть это требование, необходимо ввести ограничение на принимаемые значения атрибута **Национальность** сущности **Художник**. Сделать это можно следующим образом:

- откройте редактор атрибутов сущности **Художник** (рис. 1.7);
- выделите атрибут **Национальность**;
- найдите вкладку **Constraint** (Ограничение) и добавьте новое ограничение, нажав кнопку ;
- в появившемся окне редактора ограничений (Validation Rules) нажмите кнопку **New** и дайте название новому правилу, например: **ДопустимыеНациональности**;
- выберите тип **Valid Values List** (Список допустимых значений) и введите в столбцы **Valid Value** (Допустимое значение) и **Display Value** (Отображаемое значение) названия нужных национальностей (например, *канадец, англичанин, француз, итальянец, немец, мексиканец, русский, испанец, американец*);
- сохраните изменения, нажав кнопку ОК.

Существуют также ограничения, которые в явном виде не формулируются, но следуют из простого здравого смысла. Например, ясно, что

год рождения не может быть меньше года смерти.

Достаточно задать это ограничение в модели данных. Сделать это можно следующим образом:

- откройте редактор атрибутов сущности **Художник** (рис. 1.7);
- выделите атрибут **Год смерти**;
- найдите вкладку **Constraint** (Ограничение) и добавьте новое ограничение, нажав кнопку ;
- в появившемся окне редактора ограничений (Validation Rules) нажмите кнопку **New** и дайте название новому правилу, например: **ПроверкаДатыРождения**;
- выберите тип **User-Defined** (Определяемый пользователем) и в поле **Validation Expression** (Выражение проверки) введите следующее выражение: *Год рождения < Год смерти*;
- сохраните изменения, нажав кнопку **ОК**.

Из требований предметной области также известно, что

дата продажи работы не может быть меньше даты ее приобретения,

а также, что

цена продажи работы не может быть ниже 30 тыс. и превышать 1,5 млн рублей.

Учтите эти требования к модели, самостоятельно определив ограничения, руководствуясь приведенными ранее примерами.

Контрольные вопросы и задания

1. Что такое модель данных?
2. Какие виды моделей существуют?
3. Что такое инфологическая модель?
4. Какие нотации существуют для построения инфологических моделей?
5. Каковы основные элементы стандарта IDEF1x?
6. Что такое сущность? Какие виды сущностей приняты в рамках стандарта IDEF1x?
7. Каковы правила именования сущности?

8. Что такое экземпляр сущности?
9. Что такое идентификатор сущности?
10. Какие виды идентификаторов существуют?
11. Что такое домен?
12. К каким базовым доменам можно отнести атрибуты рассматриваемой модели?
13. Какие типизированные домены можно определить для изучаемой модели данных?
14. Какие типы связей приняты в рамках стандарта IDEF1x?
15. Что такое неспецифичная связь? Найдите ее на рассматриваемой модели данных?
16. Какая связь называется идентифицирующей? Какие требования предметной области она обеспечивает в рамках рассматриваемой задачи?
17. Что такое категориальная связь? Имеется ли она в изучаемой модели?
18. Как задать ограничение, определяющее тот факт, что галерея работает только с клиентами из стран СНГ?
19. Какие этапы включает разработка инфологической модели данных?
20. Измените правило проверки года рождения так, чтобы между возможными годом смерти и годом рождения было не менее 15 лет.
21. В чем недостаток ограничения Допустимые Национальности? Можно ли его задать иначе? Каким образом? Предложите свои варианты.

ЛАБОРАТОРНАЯ РАБОТА №2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Цель: получить навыки проектирования баз данных посредством трансформации ER-модели с использованием CASE-средств;
изучить способы создания представлений, триггеров и хранимых процедур CASE-средствами.

Преобразование полноатрибутивной логической модели данных в физическую для Microsoft SQL Server 2008 R2

Для преобразования разработанной модели данных в реляционную схему данных **CA ERwin Data Modeler** обладает встроенным механизмом. Чтобы осуществить такое преобразование следуйте рекомендациям, представленным ниже:

1. Выполните команду «**Tools**» → «**Derive New Model**».
2. Выбор параметров целевой модели (**Target Model**). В открывшемся диалоговом окне в «**New Model Type**» выбираем «**Physical**», а в «**Target Database**» устанавливаем «**SQL Server**» и версию «**2005/2008**». Нажимаем «**Далее**» переходим к дальнейшим настройкам.

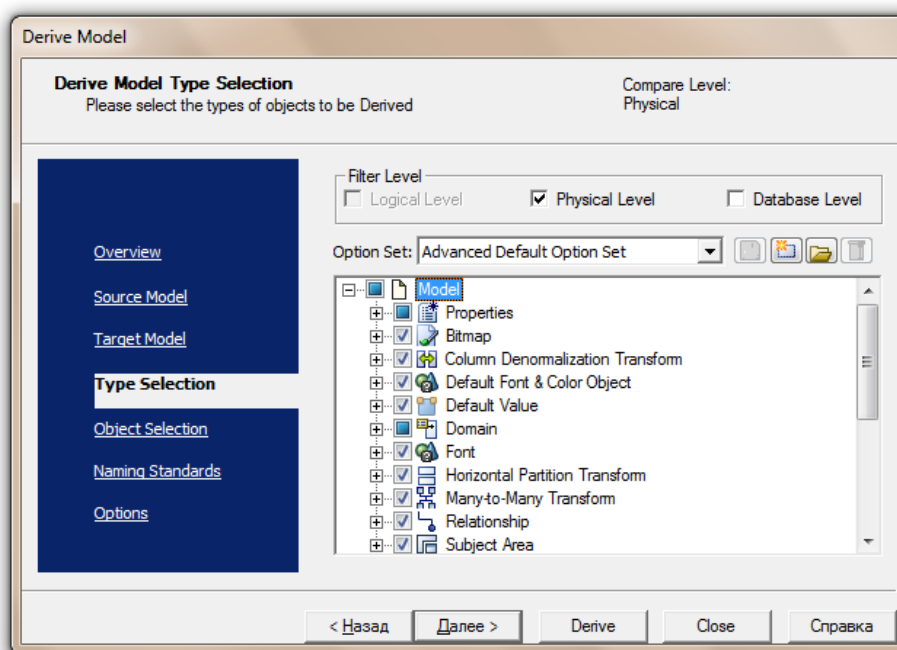


Рис. 2.1. Настройки новой модели

3. Следующий шаг (**Type Selection**) содержит в себе выбор из списка тех объектов модели данных, которые должны быть добавлены в физическую

модель из исходной логической модели данных. Нас интересует полная трансформация модели, мы оставляем параметр «**Option Set**» в значении «**Advanced Default Option Set**» и переходим к следующему шагу настроек, нажимая «Далее» (рис. 2.1).

4. На шаге «**Object Selection**» предлагается отметить в списке объекты модели данных, которые будут включены в физическую модель. Поскольку нет необходимости исключать что-либо из новой модели, то оставляем конфигурацию без изменений (рис 2.2).

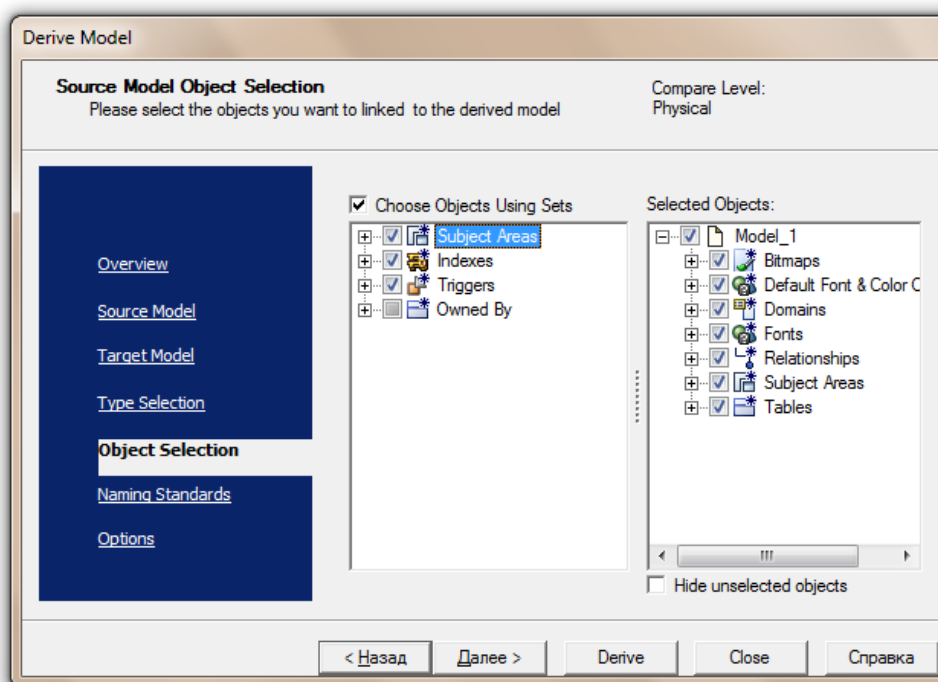


Рис. 2.2. Выбор объектов новой модели

5. На шаге «**Naming standards**» можно задать некоторые настройки стандартов имен в новой модели, в частности это префиксы атрибутов сущностей, действия в случае специальных символов (оставить, удалить, заменить) и др. настройки, котЗорые в нашей модели также не изменяются, т.е. нажимаем «Далее».
6. Наконец, на шаге «**Options**» можно задать опции автоматического преобразования связей «**многие-ко-многим**» и «**категориальную**». В нашем случае отметим флажок «**Many-to-Many Relationships**» и нажмем кнопку «**Derive**» (извлечь).

В результате из разработанной ранее модели данных получаем физическую модель данных SQL Server 2005/2008 (рис. 2.3). Как вы можете видеть, полученная модель мало отличается от исходной модели (рис. 1.9).

Исключением будет появление новой таблицы (на уровне физической модели сущности превращаются в таблицы) «**Художник_Клиент**». Как вы знаете, реляционная модель данных не допускает наличия связей «многие-ко-

многим». Поэтому в ходе преобразования произошла трансформация связи этого типа в конструкцию, состоящую из новой таблицы (ее называют таблицей связи) и двух идентифицирующих связей принадлежности типа «один-многим». Обратите внимание, что таблица связи предназначена для хранения только ключевых атрибутов.

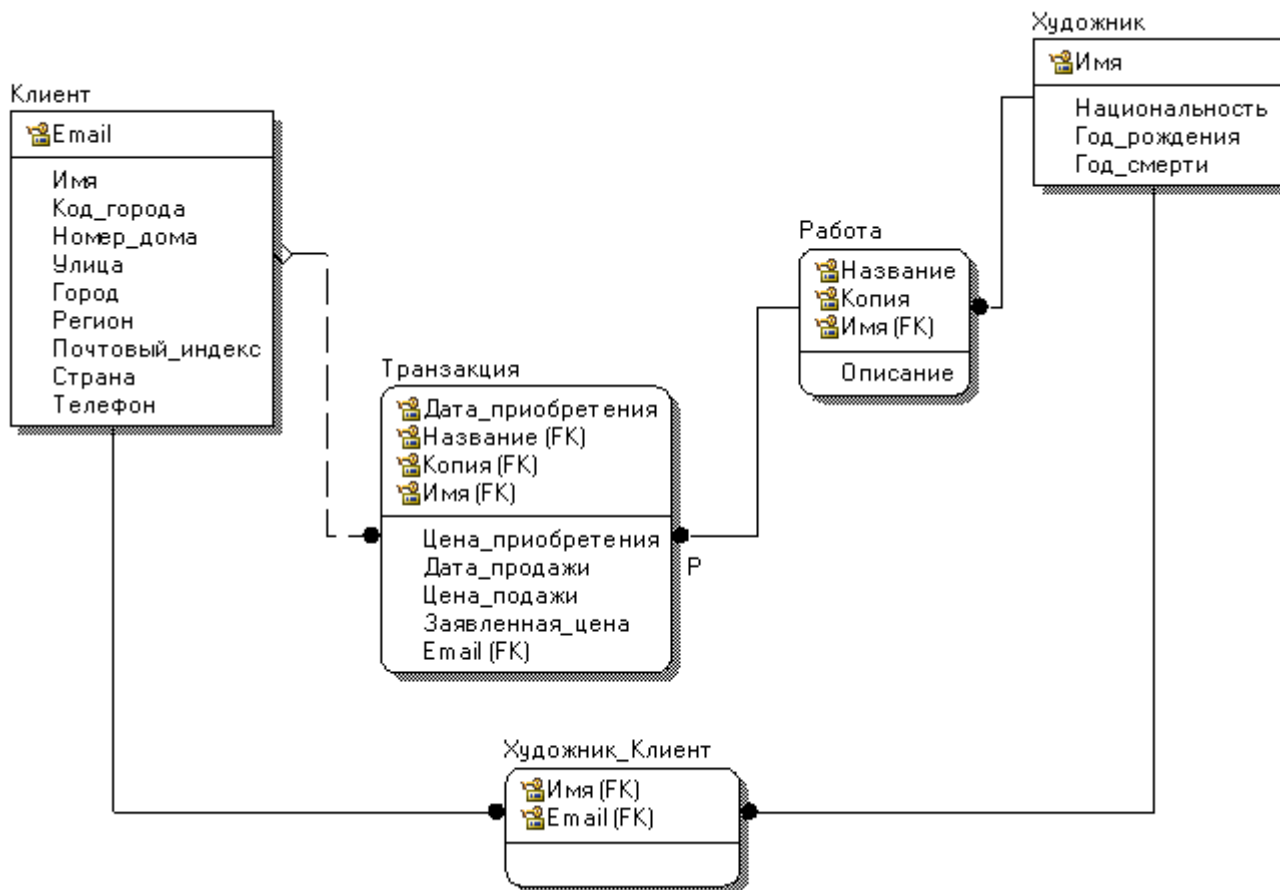


Рис. 2.3. Физическая модель данных

Другим отличием полученной модели является отображение внешних ключей. Нелишне напомнить, что в реляционной модели данных связь между двумя таблицами обеспечивается размещением ключевого поля родительской таблицы среди полей дочерней. Такие поля помечены (FK) – foreign key (внешний ключ).

Основная задача первичного ключа состоит в обеспечении уникальности каждой сущности или в случае реляционной модели уникальности каждой записи отдельной таблицы. При этом этот же ключ играет определяющую роль в образовании связей между таблицами. Поэтому он должен быть достаточно компактным и коротким. Как вы можете видеть, для полученной нами физической модели это далеко не так. Большинство ключей представляют собой строку, а таблицы «Работа» и «Транзакция» и вовсе имеют довольно громоздкие, составные первичные ключи. Обычно в таких случаях используют так называемые суррогатные ключи. Суррогатные ключи – это короткие числа, которые никогда не меняются.

В целях совместимости с СУБД, которая будет использоваться в дальнейшем для хранения схемы, и чтобы избежать проблем при разработке приложения, следует заменить названия сущностей, атрибутов и связей либо используя их англоязычный вариант, либо заменив русские символы на латинские.

Редактирование имен и типов данных

В соответствии с представленной ниже таблицей, осуществите «латинизацию» используемых в модели имен и отредактируйте типы данных.

Клиент		CUSTOMER	
Email	Email	Varchar(50)	NOT NULL
Имя	Name	Varchar(25)	NOT NULL
Код города	AreaCode	Varchar(4)	NULL
Номер дома	HouseNumber	Varchar(5)	NULL
Улица	Street	Varchar(30)	NULL
Город	City	Varchar(35)	NULL
Регион	Region	Varchar(30)	NULL
Почтовый индекс	ZipPostalCode	Varchar(6)	NULL
Страна	Country	Varchar(50)	NULL
Телефон	PhoneNumber	Varchar(8)	NULL
Художник		ARTIST	
Имя	Name	Varchar(25)	NOT NULL
Национальность	Nationality	Varchar(30)	NULL
Год рождения	BirthYear	Numeric(4)	NULL
Год смерти	DeceaseYear	Numeric(4)	NULL
Работа		WORK	
Название	Title	Varchar(25)	NOT NULL
Копия	Copy	Varchar(8)	NULL
Описание	Description	Varchar(100)	NULL
Транзакция		TRANS	
Дата приобретения	DateAcquired	Datetime	NOT NULL
Цена приобретения	AcquisitionPrice	Numeric(8,2)	NULL
Дата продажи	PurchaseDate	Datetime	NULL
Цена продажи	SalesPrice	Numeric(8,2)	NULL
Заявленная цена	AskingPrice	Numeric(8,2)	NULL

Кроме переименования таблиц и столбцов, следует также переименовать и отредактировать правила проверки, созданные в предыдущей лабораторной работе, такие как: **ДопустимыеНациональности**, **ПроверкаДатыРождения** и другие, созданные на этапе моделирования данных. **Выполните эту задачу самостоятельно.**

- ✓ **Примечание.** При редактировании правил проверки и значений по умолчанию не забудьте выставить флаг **CHECK constraint**.

Переход к суррогатным ключам, индексирование

Как правило, суррогатный ключ — это просто числовое поле, в которое заносятся значения из возрастающей числовой последовательности. В MS SQL Server существует свойство **IDENTITY** для целых типов данных. Если такое свойство указано для числового поля, то при добавлении записи в таблицу автоматически записывается уникальное для этой таблицы числовое значение, так называемый «автоинкремент».

Для перехода к суррогатным ключам нам необходимо выполнить некоторые изменения в модели.

Первое. Измените все идентифицирующие связи на неидентифицирующие: а именно, связь **ARTIST – WORK** и **WORK - TRANS**:

- щелчком мыши по связи откройте окно свойств связи «**Relationships**»;
- смените тип связи с **Identifying** на **Non-identifying**.

Второе. Для каждой таблицы, исключая **CUSTOMER_ARTIST_INT**, выполните следующие действия:

- откройте редактор поле таблицы (двойной щелчок мыши по таблице);
- выделите ключевые поля и снимите флажок «**Primary Key**»;
- добавьте новое поле типа **Number**, дав ему наименование <Имя таблицы>**ID**, например, **CustomerID**;
- установите для этого поля флаг «**Primary Key**»;
- перейдите на вкладку «**SQL Server**», укажите тип данных «**int**»;
- установите свойство **IDENTITY**, в поле справа введите текст «**1,1**»;
- перейдите на вкладку «**Index**» и нажмите кнопку выбора (кнопка с тремя точками);
- добавьте новый индекс «**New Unique Index**», имя оставим по умолчанию;
- активируйте вкладку «**Members**» и отметьте поля, которые ранее были первичным ключом для данной таблицы, например, «**Email**».

- ✓ **Примечание.** Формат **IDENTIFY (n,m)** представляет собой пару значений, где **n** – начальное значение суррогатного ключа, а **m** – приращение.

Третье. В окне модели вызовите контекстное меню и выполните три команды: Table Display -> Primary Key Designator; Table Display -> Foreign Key Designator (FK); Table Display -> Alternative Key Designator (AK). Сверьте результат с моделью, представленной на рис. 2.4.

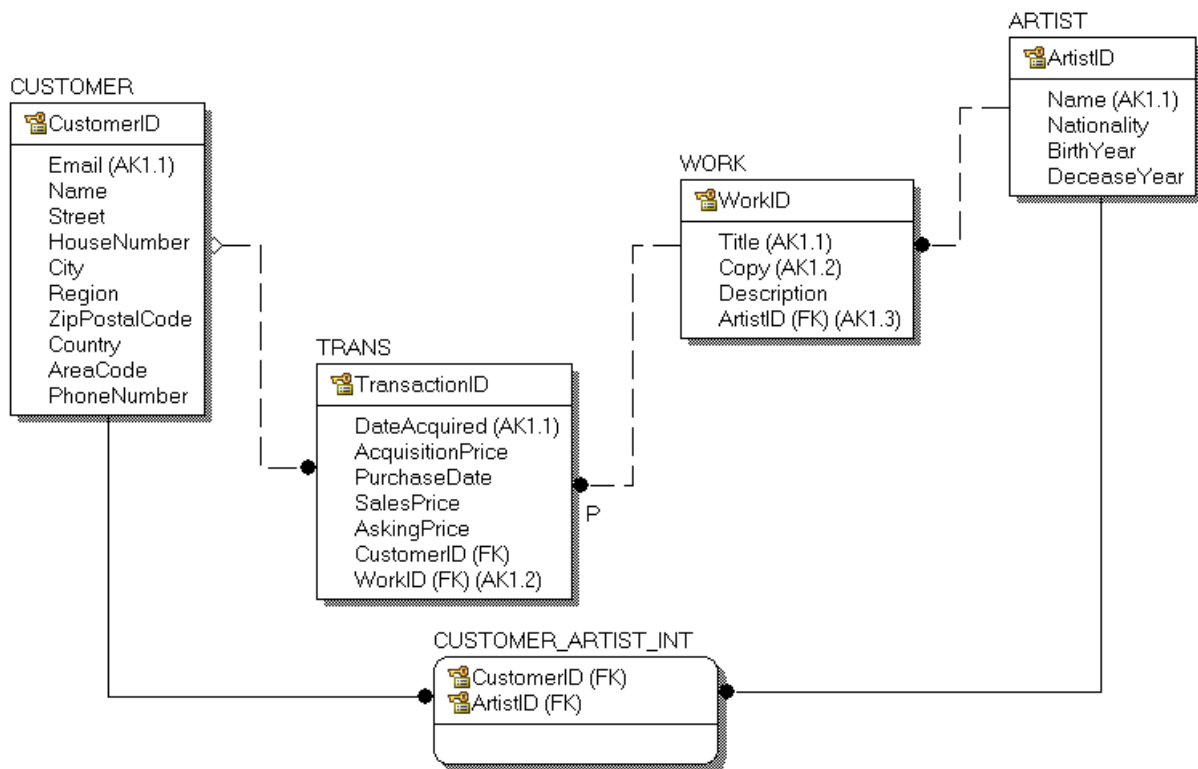


Рис. 2.4. Физическая модель данных с суррогатными ключами

Обратите внимание, что при переходе к суррогатным ключам каждая таблица, исключая **CUSTOMER_ARTIST_INT**, стала идентификационно-независимой, а все связи – неидентифицирующими.

Очевидно, что это нарушает исходные требования модели и создает условия для их нарушения. В частности, каждый клиент должен был идентифицироваться по **Email**, т.е. **Email** должен быть уникальным для нашей базы данных. Именно с этой целью, нам пришлось ввести дополнительный индекс АК - «альтернативный ключ», который позволяет СУБД контролировать это ограничение автоматически и запрещать появление записей, содержащих неуникальный адрес электронной почты - **Email**.

Для таблицы **ARTIST**, согласно исходным требованиям, следует создать индекс альтернативного ключа по полю **Name**.

Создание уникального индекса по нескольким атрибутам в таблицах **TRANS** и **WORK**, также позволяет обеспечить выполнение требований предметной области.

Следует заметить, что в реальной практике эту задачу можно решить и другими приемами, выбор которых определяется задачами эффективности, производительности и безопасности.

Определение процедур обеспечения ссылочной целостности

Наличие связи между двумя таблицами (здесь мы будем рассматривать только бинарные связи) означает ограничение на значения внешнего ключа. Другими словами внешний ключ – поле дочерней таблицы, представляющее ссылку на первичный ключ родительской – может принимать только те значения, которые определены в родительской таблице. Такой механизм называется ссылочной целостностью.

Контроль целостности должен осуществляться в ходе вставки новой записи, изменения или удаления имеющейся записи на стороне родительской и дочерней таблиц. Такой контроль СУБД осуществляет с помощью триггеров.

Триггер ([2], гл. 7, с. 329) – это специальная программа, назначаемая таблице или представлению. Триггер вызывается СУБД, когда пользователь запрашивает вставку, обновление или удаление строки из таблицы, которой принадлежит данный триггер.

Принято различать стандартные процедуры обеспечения ссылочной целостности (**Referential Integrity Trigger Actions**) и триггеры, назначаемые пользователями.

Триггеры, определяемые пользователем, используются для проверки допустимости вводимых данных, присвоения значений по умолчанию, процедур обеспечения ссылочной целостности и некоторых других операций, которые не могут быть обеспечены стандартными средствами.

Для каждой связи может быть определено 6 процедур обеспечения ссылочной целостности: 3 со стороны родительской таблицы и 3 со стороны дочерней. Для того чтобы задать требуемые процедуры, необходимо

- двойным щелчком мыши по связи войти в редактор свойств связи;
- перейти на вкладку **RI Actions** и задать требуемые процедуры вставки, обновления и удаления как на стороне дочерней (**Child**) таблицы, так и на стороне родительской (**Parent**).

Для отображения процедур на диаграмме, нужно

- в окне модели вызвать контекстное меню;
- выполнить команду **Relationship Display > Referential Integrity**.

Выполните настройку связей в соответствии с результатом, представленном на рис. 2.5.

Приведем некоторые пояснения полученных результатов. Рассмотрим связку CUSTOMER – CUSTOMER_ASRTIST_INT – ARTIST. Со стороны родительских таблиц назначены процедуры каскадного обновления (U:C) и удаления (D:C). Впрочем, наличие процедуры каскадного обновления лишнее, поскольку суррогатный ключ и так не может быть изменен.

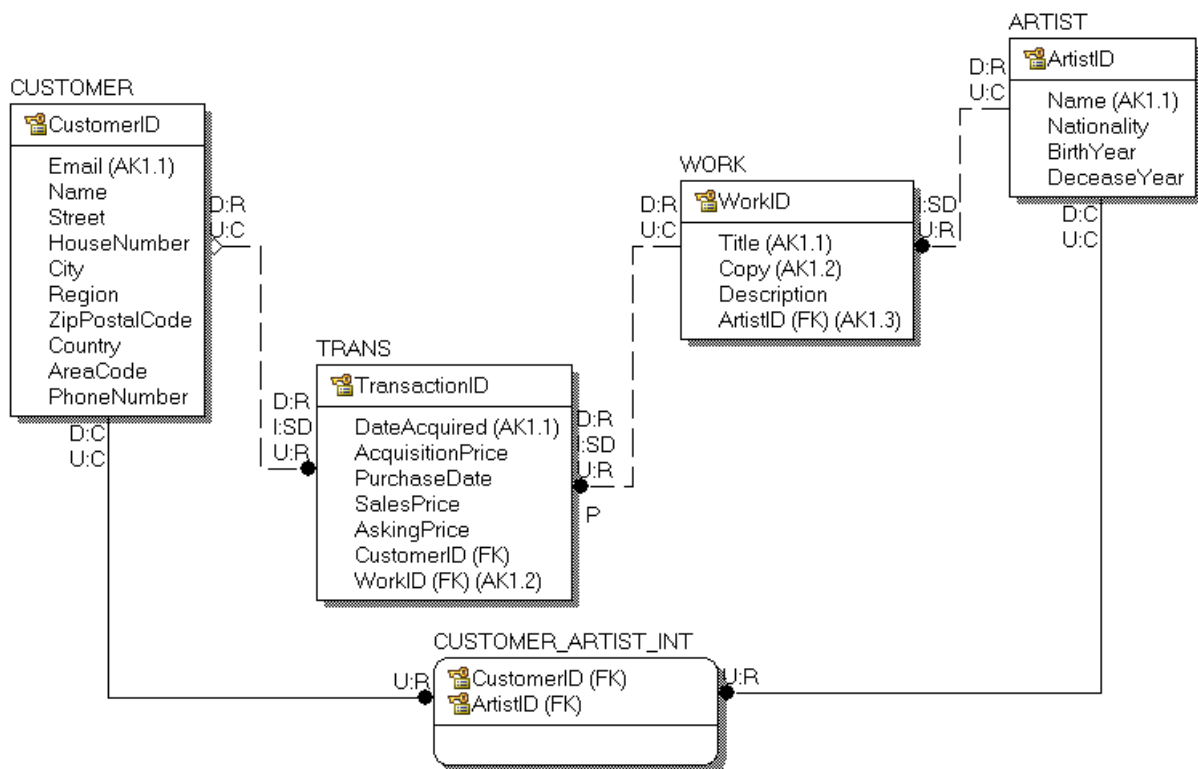


Рис. 2.5. Реляционная модель с процедурами ограничения ссылочной целостности

А вот когда сведения о клиенте или художнике удаляются из базы данных, нет нужды сохранять информацию о предпочтениях данного клиента или интересе к данному художнику. Поэтому вместе с удалением записи о клиенте или художнике каскадно удаляются соответствующие записи из таблицы CUSTOMER_ARTIST_INT.

Рассмотрим связку CUSTOMER – TRANS. При удалении записей о клиенте необходимо проверять наличие связанных с ним записей в таблице TRANS, если такие записи существуют, то СУБД запрещает (Restrict) удаление записи об этом клиенте (D:R). Аналогично нельзя удалить запись о сделке, если уже был указан какой-то клиент (D:R). Вообще удаление записей о сделке контролируется и со стороны связи с таблицей WORK. Значение столбца TRANS.CustomerID изменяется при первой продаже работы. После этого, однако, значение CustomerID меняться не должно. Таким образом, если столбец CustomerID пуст, то его обновление возможно, в противном случае обновление не допускается. Реализовать подобное правило стандартными способами невозможно. Однако можно попробовать решить данную задачу с помощью пользовательских триггеров.

Создание представлений

SQL-представление – это виртуальная таблица, составленная из других таблиц или представлений. Представление не имеет своих собственных данных,

а объединяет данные из таблиц или представлений, которые в него входят. Представления создаются с помощью оператора `SELECT`; единственное ограничение заключается в том, что они не могут включать в себя конструкцию `ORDER BY` (по стандарту SQL-92), однако промышленные СУБД, такие как Oracle и MS SQL Server, допускают данную конструкцию в определении представления. Сортировка должна обеспечиваться оператором `SELECT`, который обрабатывает представление.

Представления в основном используются для сокрытия столбцов и строк, отображения вычислительных столбцов, сокрытия сложного синтаксиса и других целей.

ERwin также поддерживает визуальное создание представлений в схеме, которые создаются аналогично сущностям таблиц. Для этого необходимо выбрать элемент «**View Table**» на панели инструментов. Поместив объект представления в области модели, необходимо указать его имя, после чего в ToolBox-е выбрать элемент «**View Relationship**», с помощью которого соединить используемую представлением таблицу с самим представлением. При этом, добавлять отображаемые в представлении столбцы данных можно просто перетаскивая их из используемых таблиц – связь «**View Relationship**» в таком случае будет появляться автоматически. При двойном щелчке на созданном представлении открывается окно настроек выбранного представления, позволяющее расширенно редактировать представление – например, добавлять вычисляемые поля, устанавливать права доступа и т.д.

Использование представлений для скрывания столбцов и строк

С помощью представлений можно скрывать отдельные столбцы таблиц. Это делается для того, чтобы возвращаемый результат имел более простой вид, а также для предотвращения доступа к конфиденциальным данным. Предположим, что пользователям базы данных View Ridge нужны только имена и номера телефонов клиентов, но не их домашние адреса или адреса электронной почты. Следующий оператор создает представление **BasicCustomerData**, содержащее только эти данные:

```
CREATE VIEW          BasicCustomerData AS
SELECT              Name, AreaCode, PhoneNumber
FROM                CUSTOMER;
```

Использование представлений для отображения вычисляемых столбцов

Еще одно применение представлений – отображение результатов вычислений, не прибегая к вводу формул пользователем. Например, следующее представление объединяет столбцы **AreaCode** и **PhoneNumber** и форматирует результат:

```
CREATE VIEW          CustomerPhone AS
```

```
SELECT      Name, ('(' + AreaCode + ') ' + PhoneNumber) As Phone
FROM        CUSTOMER;
```

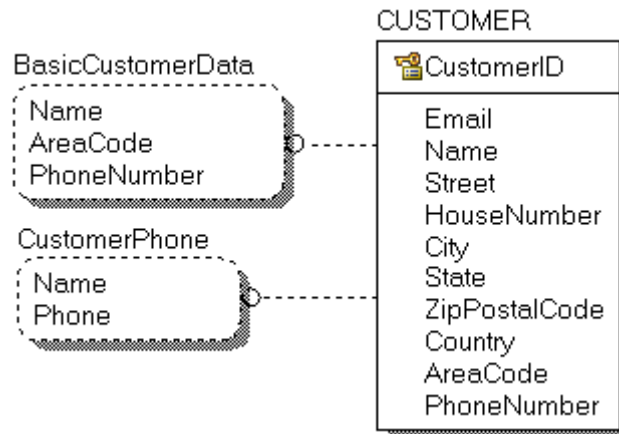


Рис. 2.6. Результат визуального отображения представления **BasicCustomerData** и **CustomerPhone**

Выполнение необходимых вычислений в представлениях имеет два преимущества. Во-первых, это избавляет пользователей от необходимости вводить математическое выражение, чтобы получить желаемый результат (а также от необходимости знать, как это делается). Во-вторых, это обеспечивает единообразие результатов. Если каждый разработчик, использующий вычисления, будет писать собственные выражения, то они, скорее всего, будут написаны по-разному, из-за чего результаты будут иметь неодинаковый вид.

Использование представлений для скрытия сложного синтаксиса

Это может делаться для того, чтобы избавить разработчиков от необходимости вводить сложный оператор всякий раз, когда им требуется определенное представление, или для того, чтобы разработчики, не знающие SQL, могли тем не менее воспользоваться преимуществами, которые предоставляют сложные SQL-операторы. Кроме того, как и в случае использования представлений для вычислений, это обеспечивает единообразие результатов.

Продавцам-консультантам галереи View Ridge необходимо знать, какими художниками интересуется тот или иной клиент. Поскольку связь между сущностями **CUSTOMER** и **ARTIST** имеет вид N:M, она представлена в виде таблицы пересечения. Таким образом, чтобы отобразить сведения об интересах клиентов, необходимо выполнить два соединения: сначала соединить таблицы **CUSTOMER** и **CUSTOMER_ARTIST_INT**, а затем полученный результат соединить с таблицей **ARTIST**. Представление, содержащее эти два соединения, конструируется с помощью следующего SQL-оператора:

```
CREATE VIEW      CustomerInterests AS
SELECT          C.Name as Customer, A.Name as Artist
```



```

FROM      CUSTOMER C
JOIN      CUSTOMER_ARTIST_INT CI
         ON C.CustomerID = CI.CustomerID
JOIN      ARTIST A
         ON CI.ArtistID = A.ArtistID;

```

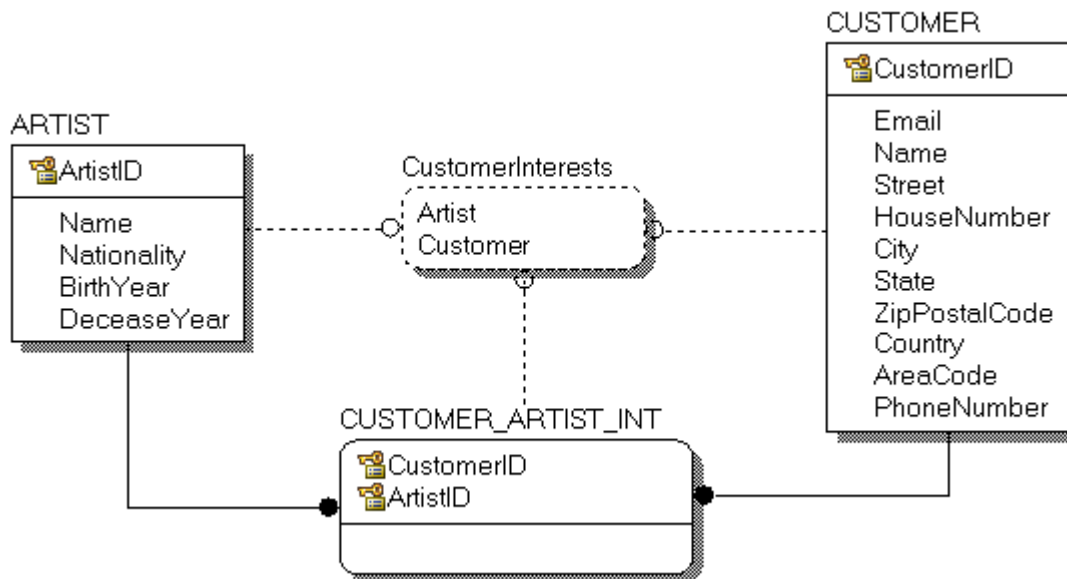


Рис. 2.7. Результат визуального отображения представления CustomerInterests

Скрытие группировки и встроенных функций

Представления используются также для скрытия группировки и встроенных функций. Рассмотрим следующее определение представления:

```

CREATE VIEW      ArtistWorkNet AS
SELECT          W.WorkID, Name, Title, Copy, AcquisitionPrice,
SalesPrice, (SalesPrice - AcquisitionPrice) AS NetPrice
FROM            TRANS T
JOIN            WORK W
              ON T.WorkID = W.WorkID
JOIN            ARTIST A
              ON W.ArtistID = A.ArtistID;

```

Это представление соединяет таблицы **TRANS**, **WORK** и **ARTIST** и создает вычисляемый столбец **NetPrice**.

Это представление можно использовать для определения другого представления:

```

CREATE VIEW      WorkNet AS
SELECT          Name, Title, Copy, sum(NetPrice) AS TotalNet
FROM            ArtistWorkNet
GROUP BY       Name, Title, Copy;

```

Это позволит отобразить прибыль от продажи каждой картины. Аналогичным образом можно определить представление ArtistNet:

```

CREATE VIEW      ArtistNet AS
SELECT          Name, sum(NetPrice) AS TotalNet
FROM           ArtistWorkNet
GROUP BY       Name;

```

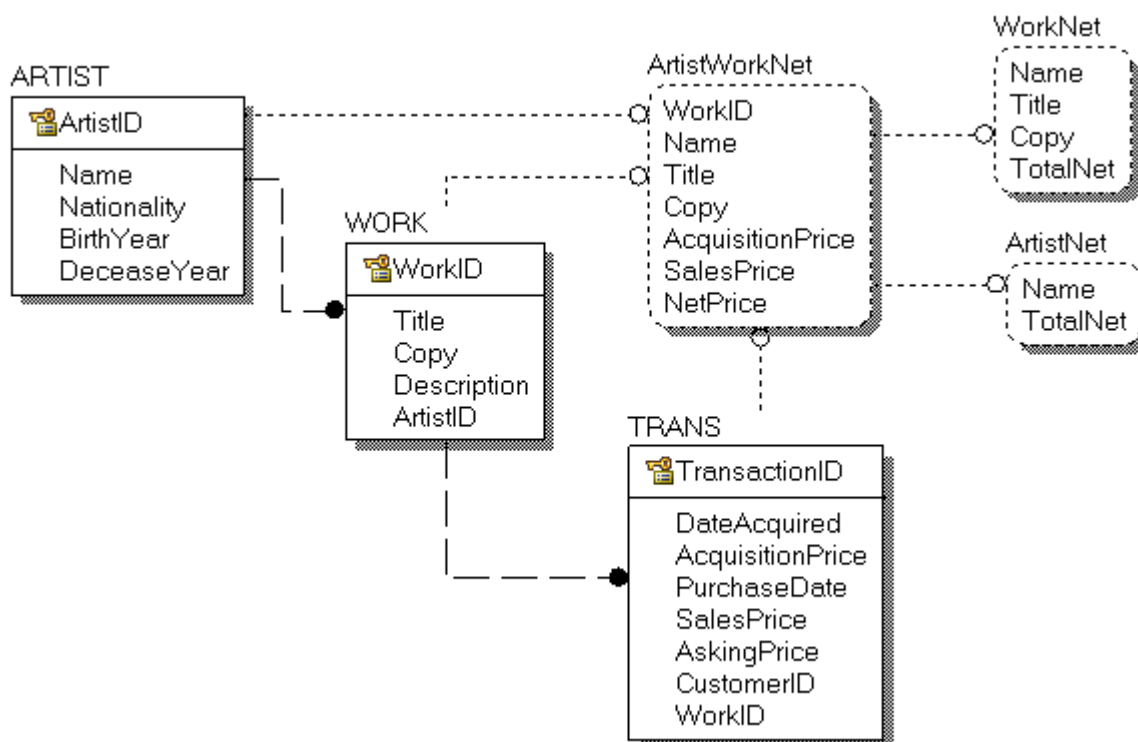


Рис. 2.8. Результат визуального отображения представлений **ArtistWorkNet**, **WorkNet** и **ArtistNet**

Создание пользовательских триггеров

Триггеры, которые пользователь разрабатывает сам, можно добавить двумя способами: встроенными средствами CA ERWin Data Modeler и вручную через консоль управления SQL Server. При помощи CA ERWin Data Modeler это делается в меню «**Database**» → «**Triggers**», где можно выбрать область действия создаваемого триггера: табличный, представления, базы данных и серверный.

Рассмотрим инструмент создания табличного триггера средствами CA ERWin Data Modeler: «**Database**» → «**Triggers**» → «**Table...**». В редакторе нам предлагаются такие параметры:

- **Table** – имя таблицы, для которой создается триггер;
- **Name** – имя создаваемого триггера (по-умолчанию Trigger-N, где N это номер триггера);
- **Schema** – схема, в которой создается триггер;
- **Insert/Update/Delete** – события, запускающие выполнение триггера;
- **Trigger Timing** – момент запуска триггера;

- **Generate** – генерация триггера в создаваемой схеме.

Вкладки:

- **General** – настройки исполнения триггера;
- **Code** – язык реализации и исходный код триггера;
- **Expanded** – просмотр и редактирование кода триггера в расширенном режиме;
- **Comment** – комментарии к триггеру;
- **UDP** – определяемые пользователем свойства.

Таким образом, средство работы с триггерами в CA ERWin Data Modeler позволяет создавать, редактировать и удалять триггеры для каждой из выбранных областей (таблицы, представления, базы, сервера). К сожалению, текстовый редактор CA ERWin Data Modeler лишен таких возможностей, как подсветка синтаксиса, развитые средства форматирования кода и т.п. Поэтому при создании кода триггера удобнее воспользоваться другими редакторами (например, Notepad++). После создания кода триггера во внешних редакторах, он просто добавляется в модель для поддержания и сохранения ее актуальности. Впоследствии, когда будет создана база данных и модель будет перенесена на СУБД, для создания и редактирования кода триггеров удобнее использовать встроенные средства данной СУБД.

Использование триггеров для присвоения значения по умолчанию

Триггеры можно использовать для присвоения начальных значений, расчет которых слишком сложен, чтобы его можно было выполнить с помощью ограничения **DEFAULT** в определении столбца. Например, согласно ценовой политике галереи View Ridge, запрашиваемая цена произведения, которая устанавливается по умолчанию, зависит от того, появлялось ли это произведение в галерее в прошлом. Если нет, она устанавливается равной удвоенной стоимости приобретения. Если произведение уже появлялось в галерее прежде, запрашиваемая цена устанавливается равной большему из двух значений – удвоенной стоимости приобретения или стоимости приобретения плюс средняя чистая прибыль от продажи произведения в прошлом.

Завершающий триггер, представленный ниже реализует эту ценовую политику. Сначала из таблицы `inserted` получают значения **WorkID** и **AcquisitionPrice**. Затем выполняется оператор **SELECT FROM TRANS** для подсчета количества строк с данным значением **WorkID**. Переменная **@countPriorRows** устанавливается равной **@@rowCount – 1**, поскольку данный триггер является завершающим, и новая строка будет уже добавлена в базу данных. Следовательно, количество удовлетворяющих условию строк в таблице **TRANS**, которое было в базе данных до выполнения вставки, равно **(@@rowCount – 1)**

```

-- Триггер, присваивающий значение по умолчанию --
CREATE TRIGGER SetAskingPrice ON TRANS
FOR INSERT
AS
DECLARE
    @WorkID          as int,
    @TransID         as int,
    @aqPrice         as money,
    @SumNetPrice     as money,
    @countPriorRows as int,
    @newPrice        as money,
    @netPrice        as int

/* Получаем новые значения WorkID и AcquisitionPrice */
SELECT    @WorkID = WorkID, @aqPrice = AcquisitionPrice,
          @TransID = TransactionID
FROM      inserted

/* Проверяем, не появлялось ли произведение в галерее ранее */
SELECT * FROM    TRANS T
WHERE          T.WorkID = @WorkID

/* Это завершающий триггер, поэтому число строк будет включать
   только что вставленную строку. Вычитаем единицу, что бы учесть это
*/
Set @countPriorRows = @@rowCount - 1

/* Устанавливаем newPrice и при необходимости корректируем ее значение
*/
Set @newPrice = 2 * @aqPrice
If @countPriorRows > 0
BEGIN
/* Функция AVG не подходит, поскольку в делителе она будет учтена
   текущая строка. Используем @@rowCount - 1 из предыдущих вычислений
*/
SELECT    @sumNetPrice = SUM(NetPrice)
FROM      ArtistWorkNet AW
WHERE     AW.WorkID = @WorkID
GROUP BY  AW.WorkID

/*Если необходимо, корректируем netPrice */
If @netPrice < (@SumNetPrice / @countPriorRows) + @aqPrice
    Set @newPrice = (@SumNetPrice / @countPriorRows) + @aqPrice
END

/* Теперь записываем в таблицу TRANS новое значение AskingPrice */
UPDATE    TRANS
SET       AskingPrice = @newPrice
WHERE     TransactionID = @TransID

```

Далее переменная **@newPrice** устанавливается равной удвоенному значению **AcquisitionPrice**; при необходимости ее значение будет скорректировано ниже. Если **@countPriorRows** больше нуля, то в таблице **TRANS** до выполнения вставки уже были записи о транзакциях с данным произведением, и с помощью представления **ArtistWorkNet** вычисляется

суммарная чистая прибыль от продажи этого произведения. Встроенную функцию усреднения использовать нельзя, поскольку она будет вычислять среднее на базе значения `@@rowCount`, а не `@countPriorRows`. Затем текущее значение `@newPrice` сравнивается с суммой средней чистой прибыли и стоимости произведения. Если оно оказывается меньше, переменная `@newPrice` устанавливается равной сумме средней чистой прибыли и стоимости произведения. Наконец, только что вставленная строка обновляется, в процессе чего в столбец `AskingPrice` записывается вычисленное значение `@newPrice`.

Использование триггеров в процедурах обеспечения ссылочной целостности

Согласно описанию, таблица **WORK** в базе данных View Ridge имеет обязательного потомка в виде таблицы **TRANS**. Это означает, что всякий раз, когда в таблицу **WORK** вставляется строка, необходимо вставить соответствующую ей строку в таблицу **TRANS**. Реализовать такие ограничения ссылочной целостности непросто. Ниже приведен триггер, добавляющий строку в таблице **TRANS** после создания строки в таблице **WORK**. Сначала он получает новое значение **WorkID** из псевдотаблицы `inserted`, а затем проверяет, нет ли доступной строки в таблице **TRANS**. Такой строки быть не должно, поскольку транзакция, создавшая строку в таблице **WORK**, еще не получила шанса вставить соответствующую строку в таблицу **TRANS**. После этого, если `@@rowCount` равняется нулю, триггер создает в таблице **TRANS** строку по умолчанию.

```
-- Триггер, реализующий процедуру обеспечения ссылочной целостности (а)--
CREATE TRIGGER EnforceTransChild ON WORK
FOR INSERT
AS
DECLARE    @newWorkID as int
/* Получаем новый WorkID */
SELECT    @newWorkID = WorkID
FROM      inserted
/* Проверяем, имеется ли доступная строка в таблице TRANS */
SELECT    *
FROM      TRANS
WHERE     WorkID = @newWorkID
          AND CustomerID is null
/* Если подходящей строки не найдено, вставляем строку */
If @@ROWCOUNT = 0
    INSERT INTO TRANS (DateAcquired, WorkID)
        VALUES (GETDATE(), @newWorkID)

-- Триггер, реализующий процедуру обеспечения ссылочной целостности (б)--
CREATE TRIGGER RemoveDupTrans ON TRANS
FOR INSERT
AS
DECLARE    @WorkID as int,
           @TransID as Int
SELECT    @WorkID = WorkID, @TransID = TransactionID
```

```

FROM      inserted
/* Удаляем прочие доступные строки из таблицы TRANS */
DELETE   FROM TRANS
WHERE    WorkID = @WorkID
        AND CustomerID is null
        AND TransactionID <> @TransID

```

Использование триггеров для обновления представления

Как правило, СУБД не способна обновлять таблицы, лежащие в основе представлений, созданных при помощи операции соединения. Однако, зная специфику конкретного приложения, можно определить, как следует интерпретировать запрос на обновление соединенного представления.

Рассмотрим представление **CustomerInterests**. Оно содержит строки таблиц **CUSTOMER** и **ARTIST**, соединенные через таблицу пересечения. Столбцу **CUSTOMER.Name** дан псевдоним **Customer**, а столбцу **ARTIST.Name** – **Artist**.

Запрос на изменение имени клиента в представлении **CustomerInterests** можно интерпретировать как запрос на изменение столбца **Name** в таблице **CUSTOMER**. Такой запрос, однако, может быть обработан лишь в том случае, если это имя является уникальным в таблице **CUSTOMER**. В противном случае невозможно будет определить, какую из строк следует обновлять.

Замещающий триггер, текст которого приведен ниже, реализует эту логику. Сначала триггер получает старые и новые значения столбца **Customer** представления **CustomerInterests**. Затем с помощью коррелированного подзапроса с предложением **EXIST** проверяется, является ли старое значение столбца **CUSTOMER.Name** уникальным. Если да, имя изменяется; если нет, никаких изменений не производится.

```

-- Триггер, обновляющий представление --
CREATE TRIGGER CustomerNameUpdate ON CustomerInterests INSTEAD OF UPDATE
AS
DECLARE      @newName as Varchar(30),
            @oldName as Varchar(30)

/* Получаем старое и новое имя */
SELECT      @newName = Customer
FROM        inserted

SELECT      @oldName = Customer
FROM        deleted

/* Подсчитываем количество синонимов в таблице CUSTOMER */
SELECT      *
FROM        CUSTOMER C1
WHERE       C1.Name = @oldName
            AND EXISTS
            (
                SELECT      *
                FROM        CUSTOMER C2
                WHERE C1.Name = C2.Name
            )

```

```

                                AND C1.CustomerID <> C2.CustomerID
                                )

/* Теперь обновляем данные о клиенте, если имя было уникальным */
If @@ROWCOUNT = 0
    UPDATE          CUSTOMER
    SET             CUSTOMER.Name = @newName
    WHERE          CUSTOMER.Name = @oldName

```

Создание хранимых процедур

Теперь попробуем обеспечить проверку при добавлении работы, содержится ли в базе данных художник, являющийся автором добавляемой картины и есть ли данная работа в базе данных. Если в ходе проверки выясняется, что запись о картине ссылается на художника, которого нет в базе, обработка должна завершиться с соответствующим сообщением. Также, если проверка наличия картины в базе показывает, что работа с данными характеристиками уже есть в базе данных, то добавление не должно происходить – должна создаваться лишь новая транзакция (сделка) с этой картиной. Для реализации описанной ситуации воспользуемся хранимой процедурой.

Хранимая процедура ([2], гл. 7, с. 338) – это программа, которая выполняет некоторые действия с информацией в базе данных и при этом сама хранится в базе данных. Хранимые процедуры могут принимать входные параметры и возвращать результаты. В отличие от триггеров, которые принадлежат определенной таблице или представлению, хранимые процедуры принадлежат базе данных в целом. Они могут вызываться любым процессом, использующим базу данных, при условии, что у этого процесса достаточные права доступа. Хранимые процедуры используются для многих целей. Хотя администраторы баз данных используют их для выполнения рутинных задач администрирования, главной областью их применения являются все же приложения баз данных. Эти процедуры могут вызываться из прикладных программ. Кроме того, эти процедуры можно вызывать в интерактивном режиме из командной оболочки СУБД – например, Management Studio в SQL Server.

Использование хранимых процедур для обеспечения ссылочной целостности при добавлении новых записей в таблицы

Хранимая процедура, регистрирующая приобретение произведения:

```

CREATE PROCEDURE Add_WORK
(
    @ArtistID int, /*Художник должен уже присутствовать
                   в базе данных */
    @Title Varchar(25),
    @Copy Varchar(8),
    @Description varchar(100),

```

```

        @AcquisitionPrice Numeric(6,2)
    )
/* Хранимая процедура, регистрирующая приобретение галереей
произведения. Если произведение никогда раньше не появлялось
в галерее, в таблицу WORK добавляется новая строка.
В противном случае используется существующая строка. Далее
добавляется строка в таблицу TRANS, а столбец
DateAcquired в ней устанавливается равным системной дате */
AS
    DECLARE @rowcount as int
    DECLARE @WorkID as int

/* Сначала следует убедиться, что ArtistID имеет допустимое
значение */
    SELECT @rowcount = Count(*)
    FROM ARTIST A
    WHERE A.ArtistID = @ArtistID

IF @rowcount = 0
/* Нет такого художника */
    BEGIN
        Print 'Нет художника с ID = ' + Str(@ArtistID)
        Print 'Обработка прервана.'
        return
    END

/* Теперь смотрим, есть ли это произведение в базе данных */
    SELECT @rowcount = Count(*)
    FROM WORK W
    WHERE W.ArtistID = @ArtistID and
           W.Title = @Title and
           W.Copy = @Copy

    IF @rowcount = 0
        /* Произведения нет в базе, записываем его */
        INSERT INTO WORK (Title, Copy, Description, ArtistID)
        VALUES (@Title, @Copy, @Description, @ArtistID)

/* Получаем значение суррогатного ключа WorkID */
    SELECT @WorkID = W.WorkID
    FROM WORK W
    WHERE W.ArtistID = @ArtistID and
           W.Title = @Title and
           W.Copy = @Copy

/* Вставляем новую строку в таблицу TRANS */
    INSERT INTO TRANS (DateAcquired, AcquisitionPrice, WorkID)
    VALUES (GetDate(), @AcquisitionPrice, @WorkID)

RETURN

```

Процедура подразумевает, что переданное ей значение **ArtistID** является действующим идентификатором. Чтобы убедиться в этом, первый блок операторов подсчитывает количество строк, имеющих указанное значение **ArtistID**. Если это количество равно нулю, значит, процедуре было передано недопустимое значение **ArtistID**, в этом случае процедура выводит сообщение

об ошибке и завершает работу. В противном случае процедура проверяет, не появлялось ли это произведение в галерее в прошлом. Если да, то в таблице **WORK** уже есть строка, содержащая данные о художнике, названии произведения и номере копии. Если нет, то в таблице **WORK** создается новая строка. После этого с помощью оператора **SELECT** процедура получает значение **WorkID**. Если мы имеем дело с только что созданной строкой таблицы **WORK**, этот оператор необходим, чтобы получить новое значение суррогатного ключа **WorkID**. Если произведение уже существовало в базе данных, то с помощью этого оператора мы получаем **WorkID** уже существующей строки.

Получив значение **WorkID**, процедура вставляет новую строку в таблицу **TRANS**. Обратите внимание, что для присвоения значения по умолчанию столбцу **DateAcquired** используется системная функция **GetDate()**.

Контрольные вопросы и задания

1. Что такое первичный ключ?
2. Что такое суррогатный ключ?
3. Чем различаются модели данных, основанные на первичных ключах, извлекаемых из сведений о предметной области, от основанных на суррогатных ключах?
4. В какой ситуации необходимо использовать суррогатные ключи?
5. Как из разработанной логической модели данных получить ее реляционную схему?
6. Что такое хранимая процедура?
7. Что такое триггер?
8. Что такое представление?
9. Каким образом достигается обеспечение ограничений ссылочной целостности со стороны родительских и дочерних таблиц?
10. Разработайте хранимую процедуру для добавления картины такую, что если имя картины длиннее указанного размера соответствующего поля, то часть имени обрезается и добавляется в ее описание.

ЛАБОРАТОРНАЯ РАБОТА №3. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ БАЗЫ ДАННЫХ

Цель: получить навыки создания базы данных и генерации схемы данных с использованием CASE-средств;

познакомиться с основами работы в среде Microsoft SQL Server Management Studio;

приобрести навыки написания и выполнения SQL-запросов.

Создание базы данных

Прежде чем приступить к завершающей стадии преобразования физической модели в базу данных, готовую к использованию, необходимо создать пустую базу данных. Для этого:

- запустите **Microsoft SQL Server Management Studio** [4];
 - в окне соединения к серверу введите:
 - Тип сервера = **Database Engine**;
 - Имя сервера = <Имя компьютера>\SQLEXPRESS (например, DK13_01\SQLEXPRESS);
 - Проверка подлинности = **Windows Authentication**;
 - установив параметры, нажмите кнопку «Соединить»;
 - в появившемся окне найдите и выделите пункт «**Базы данных**»;
 - щелчком правой кнопки мыши вызовите контекстное меню и выберите пункт Новая база данных;
 - введите имя для новой базы данных, например: **viewridgedb**.
- ✓ **ВНИМАНИЕ.** Убедитесь, что установленный экземпляр SQL Server допускает смешанную проверку подлинности (аутентификацию). Для этого правой кнопкой мыши в **Object Explorer** щелкните по активному соединению с SQL Server, выберите пункт «Свойства» и далее страницу «Безопасность», установите переключатель на свойстве смешанной проверки подлинности (Windows и SQL), если это было не так. Закройте Management Studio, запустите «Службы» компьютера и перезапустите службу SQL Server, чтобы сделанные изменения вступили в силу.

Для последующей работы со своей базой данных создадим учетную запись, для чего:

- в окне **Object Explorer** Microsoft SQL Server Management Studio найдите и выделите пункт **Security** (Безопасность) -> **Logins** (Точки входа);
- щелчком правой кнопки мыши вызовите контекстное меню и выберите пункт **New Login**;

- задайте имя – например, user;
- укажите **SQL Server Authentication** и дважды введите пароль;
- уберите все флажки с чекбоксов;
- укажите в качестве базы данных по умолчанию свою базу – например, viewridgedb;
- язык по умолчанию можно оставить как есть или указать English;
- перейти к пункту **User Mapping ()**;
- в верхнем окне отметить флажком свою базу данных, указать имя пользователя (если оно не указано автоматически) – user, а в поле **Default Schema** (Схема по умолчанию) ввести dbo;
- в нижнем окне найти и выделить роль **db_owner** (владелец базы данных);
- нажать кнопку **ОК**, подтвердив изменения.

Преобразование физической модели данных в базу данных SQL Server

Для трансформации физической модели данных необходимо произвести подключение к базе данных. Для этого:

- выберите пункт меню «Database» → «Database Connection...»;
- в окне SQL Server Connection установите параметры:
 - Authentication = Database Authentication;
 - User Name = user (введите Ваш login);
 - Password = user (введите Ваш пароль);
 - Server = <Имя компьютера>\SQLEXPRESS (например, DK13_01\SQLEXPRESS);
 - Database = viewridgedb.

Если подключение проходит успешно, то окно закрывается и пользователю не выдается никакого сообщения. В противном случае выдается сообщение об ошибке подключения.

После того, когда установлено подключение, можно приступить к генерации схемы данных на сервере, выбрав «Tools» → «Forward Engineer» → «Schema Generation...». В результате загружается мастер генерации схемы (рис. 3.1), в котором можно выбрать элементы модели данных, которые будут генерироваться в схему на сервере. В целом мы оставим все настройки по умолчанию. **Единственное, что следует изменить, – это способ генерации ограничений столбцов.** Для свойств **Validation** и **Default** необходимо указать способ определения с использованием оператора **ALTER**, как это показано на рис. 3.1. Если этого не сделать могут быть проблемы при автогенерации, исправить которые можно только вручную.

В ходе настройки и проверки схемы в любой момент можно предварительно создать sql-скрипт создания схемы данных (кнопка

«Preview...») для того, чтобы просмотреть, отредактировать и сохранить его в виде файла для отдельного использования, например для выполнения непосредственно на сервере.

По нажатию кнопки «Generate» при установленном подключении к СУБД запустится процесс генерации схемы, все подробности которого будут выведены в логе, что позволит отслеживать ошибки при создании схемы на сервере. Полный код скрипта (без стандартных процедур обеспечения ссылочной целостности), сгенерированного средой ERWin по разработанной нами модели, приведен в приложении.

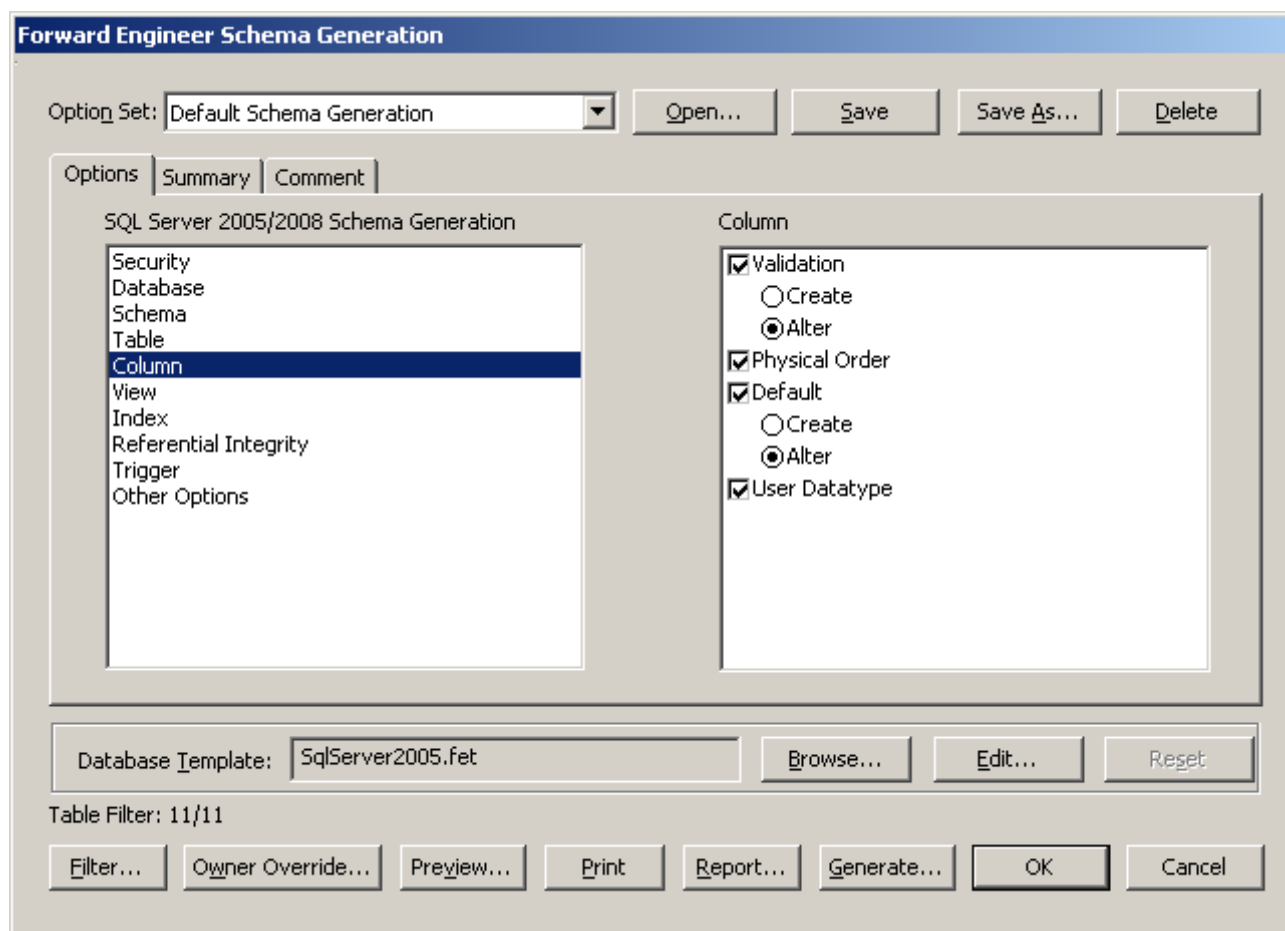


Рис. 3.1. Мастер настройки и генерации модели в схему базы данных

Загрузка данных в базу данных

Самый общий способ загрузки данных в только что созданную базу – это выполнение SQL-запросов на вставку. Рассмотрим несколько примеров.

Вставка новой записи в таблицу CUSTOMER:

```
INSERT INTO Customer  
(Name, AreaCode, City, HouseNumber, Street, Region, ZipPostalCode,  
Country, Email)
```

```
values ('Sam Fisher', '4932', 'Ivanovo', '777-23-49', 'Lenina Street',
'37', '153072', 'Russia', 'fisher@host.ru')
```

Стоит заметить, что ни в одном из запросов мы не указываем значение поля CustomerID, т.к. при разработке базы данных мы сделали это поле как IDENTITY, таким образом, при добавлении новой записи СУБД автоматически генерирует для него уникальное значение, являющееся первичным ключом новой добавленной записи.

Используя функцию TransactSQL IDENT_CURRENT('table_name'), мы получаем значение первичного ключа добавленного клиента, чтобы впоследствии по нему добавлять покупки и интересы этого клиента:

```
SELECT IDENT_CURRENT('Customer') AS Current_Identity
```

Данный запрос возвращает значение последнего сгенерированного в системе значения типа IDENTITY. В нашем случае это значения, например 1001 и 1002.

```
INSERT INTO Customer (Name, AreaCode, HouseNumber, Street, City, Region,
ZipPostalCode, Country, Email) values ('Gordon Freeman', '7/3', '374-300-
4', 'Tashkentskaya Street', 'Ivanovo', '37', '153026', 'Russia', 'g-
freeman@black-mesa.com')
SELECT IDENT_CURRENT('Customer') AS Current_Identity
```

Аналогичным образом добавим в базу данных двух художников, две картины, две сделки, два интереса клиентов к работам художников.

```
INSERT INTO Artist (Name, BirthYear, DeceasedYear, Nationality) values
('Raffaello Santi', 1483, 1590, 'Итальянец')
SELECT IDENT_CURRENT('Artist') AS Current_Identity
```

Данный запрос возвращает значение последнего сгенерированного в системе поля значения типа IDENTITY. Полученное значение первичного ключа сохраним, чтобы использовать при добавлении в базу данных информации о картинах, написанных этим художником. В нашем случае этот код равняется 1.

```
INSERT INTO Artist (Name, BirthYear, DeceasedYear, Nationality) values
('Michelangelo', 1475, 1564, 'Итальянец')
```

Вновь вышеописанным способом получаем идентификатор художника.

(2)

```
INSERT INTO Work (ArtistID, Title, Copy, Description) values (1, 'Mond
Crucifixion', '7/4', 'темная')
SELECT IDENT_CURRENT('Work') AS Current_Identity
INSERT INTO Work (ArtistID, Title, Copy, Description) values (2, 'The
Last Judgment', '37/2', 'хорошая')
```

```
SELECT IDENT_CURRENT('Work') AS Current_Identity
```

Теперь запишем в базу данных историю пребывания этих двух картин в галерее, используя полученные выше значения идентификаторов WorkID:

```
INSERT INTO Trans (WorkID, CustomerID, DateAcquired, AcquisitionPrice, PurchaseDate, SalesPrice, AskingPrice) values (1, 1001, '10.24.2008', 800.00, '10.27.2008', 850.00, 870.00)
```

```
INSERT INTO Trans (WorkID, CustomerID, DateAcquired, AcquisitionPrice, PurchaseDate, SalesPrice, AskingPrice) values (2, 1002, '11.24.2008', 600.00, '11.28.2008', 620.00, 650.00)
```

Внимание! При стандартной конфигурации Microsoft SQL Server 2008 R2 во время добавления значения типа **date** значение поля нужно указывать в формате 'ММ.ДД.YYYY' а не 'ДД.ММ.YYYY'.

Запишем в базу данных информацию об интересах клиентов к работам определенных художников. Для простоты примера в нашем случае запишем тех же художников, картины которых были куплены этими клиентами:

```
INSERT INTO Customer_Artist (CustomerID, ArtistID) values (1001, 1)
```

Так как все добавляемые в эту таблицу данные указываются непосредственно в запросе, поэтому, сохранив порядок следования полей и значений, мы можем добавлять данные, не указывая имена полей, в которые происходит добавление:

```
INSERT INTO Customer_Artist values (1002, 2)
```

Стоит заметить, что эти данные были добавлены лишь для тестирования SELECT-запросов, т.к. например, для добавления новой картины в базу данных ранее мы разработали соответствующую хранимую процедуру, использование которой будет показано далее.

Создание, модифицирование, удаление таблиц и ограничений

Проверим возможность добавления в базу данных художника, у которого значение года рождения больше значения года смерти, СУБД сгенерирует ошибку:

```
INSERT INTO ARTIST (Name, BirthYear, DeceasedYear, Nationality) values ('tester', 1900, 1800, 'Испанец');
```

Результат:

The `INSERT` statement conflicted with the `CHECK` constraint "BirthValuesCheck". The conflict occurred in database "viewridge", table "dbo.ARTIST".

The `statement` has been terminated.

Добавим ограничение на вводимую в поле «Год Рождения» и «Год смерти» последовательность цифр:

```
ALTER TABLE ARTIST
add constraint ValidBirthYear
CHECK (BirthYear LIKE '[1-2][0-9][0-9][0-9]')
```

```
ALTER TABLE ARTIST
add constraint ValidDeathYear
CHECK (DeceasedYear LIKE '[1-2][0-9][0-9][0-9]')
```

Теперь, при попытке добавить некорректный код рождения или смерти художника СУБД сгенерирует ошибку, выдав соответствующее сообщение:

```
INSERT INTO ARTIST (Name, BirthYear, DeceasedYear, Nationality) values
('tester', 2900, 1800, 'Испанец');
```

Все дальнейшие действия проверяйте на копии исследуемой базы. Попробуем удалить таблицу `CUSTOMER_ARTIST`. Для этого сначала удалим ограничения ссылочной целостности, связанные с таблицей `CUSTOMER_ARTIST`. Чтобы их узнать, выполним команду:

```
exec sp_help CUSTOMER_ARTIST
```

В результате выполнения команды будет выведена информация о таблице, из которой можно получить названия связанных с ней ограничений, в нашем случае они следующие:

```
ALTER TABLE CUSTOMER_ARTIST DROP CONSTRAINT HAS_INTEREST_IN
ALTER TABLE CUSTOMER_ARTIST DROP CONSTRAINT INTERESTED_FOR
```

Теперь можно удалить таблицу `CUSTOMER_ARTIST`:

```
DROP TABLE CUSTOMER_ARITST
```

Восстановим удаленную таблицу `CUSTOMER_ARTIST`, т.е. создадим ее заново:

```
CREATE TABLE CUSTOMER_ARTIST
(
    ArtistID int NOT NULL,
```

```

CustomerID int NOT NULL,
CONSTRAINT CustomerArtistPK
PRIMARY KEY (ArtistID, CustomerID),
CONSTRAINT Customer_Artist_Int_ArtistFK
FOREIGN KEY (ArtistID)
REFERENCES ARTIST (ArtistID)
ON UPDATE CASCADE
ON DELETE CASCADE,
CONSTRAINT Customer_Artist_Int_CustomerFK
FOREIGN KEY (CustomerID)
REFERENCES CUSTOMER (CustomerID)
ON UPDATE CASCADE
ON DELETE CASCADE
);

```

Проверим выполнение ограничений ссылочной целостности созданной таблицы, добавим несуществующие в базе данных идентификаторы (коды) художника и клиента:

```
INSERT INTO CUSTOMER_ARTIST VALUES (111, 222)
```

Изменим схему таблицы, добавив поле «Описание»:

```
ALTER TABLE CUSTOMER_ARTIST
ADD description Varchar(25)
```

Изменим тип данных добавленного столбца:

```
ALTER TABLE CUSTOMER_ARTIST
ALTER COLUMN description varchar(100)
```

Удалим добавленный столбец:

```
ALTER TABLE CUSTOMER_ARTIST
DROP COLUMN description
```

Тестирование. Проверка исполнения базовых требований

1. Вывести список клиентов и количество купленных каждым из них работ, после чего отсортировать их в порядке убывания, для определения самых активных из них:

```

select      t.Name, COUNT(t.Name) as SelQuant
from (select Customer.Name
      from Customer
      inner join TRANS
      on Customer.CustomerID = TRANS.CustomerID
      ) as t
group by t.Name
order by SelQuant desc

```


2. Вывести данные о текущем хозяине и местонахождении картины «Mi Vida», копии под номером «7/100»:

```
select      Name,
            AreaCode,
            HouseNumber,
            Street,
            City,
            Region,
            ZipPostalCode,
            Country
from        Customer
where       Customer.CustomerID =
(
    select   TRANS.CustomerID
    from     TRANS
    where    WorkID =
            (
                select      WorkID
                from        WORK
                where Work.Title like '%Mi Vida%'
                and
                Work.Copy = '7/100'
            )
    and
    TransactionID =
        (
            select      MAX(TransactionID)
            from        TRANS
            where       WorkID =
                (
                    select      WorkID
                    from        WORK
                    where Work.Title like '%Mi Vida%'
                    and
                    Work.Copy = '7/100'
                )
        )
)
)
```

Принцип работы запроса – производится поиск кода соответствующей работы, а также последней транзакции (с максимальным значением кода), вследствие чего определяется клиент, купивший указанную работу.

3. Вывести список всех произведений, когда-либо появлявшихся в галерее, а также их авторов:

```
select      WORK.Title, Artist.Name
from        WORK
            inner join ARTIST
            on Work.ArtistID = ARTIST.ArtistID
where       Work.WorkID in
            (
                select distinct WorkID
                from TRANS
            )
```

)

4. Определить, насколько быстро продаются работы художника «Miro»:

```
select      Work.Title,
            DATEDIFF(d, DataAcquired, PurchaseDate) as DayPeriod
from        TRANS
            inner join WORK
            on Trans.WorkID = Work.WorkID
where       TRANS.WorkID in
            (
              select WorkID
              from Work
              where ArtistID =
                (
                  select      ArtistID
                  from        ARTIST
                  where       Name = 'Miro'
                )
            )
```

5. Определить, насколько быстро продаются работы художников в галерее (иной вариант реализации запроса 4):

```
select      Artist.Name, Work.Title,
            DATEDIFF(d, DataAcquired, PurchaseDate) as DayPeriod
from        TRANS
            inner join WORK
            on TRANS.WorkID = Work.WorkID
            inner join ARTIST
            on Work.ArtistID = Artist.ArtistID
where       not TRANS.CustomerID in null
```

6. Какова прибыль от продаж работ художника «Miro»?

Для этого можно использовать созданное ранее представление ArtistNet

```
select      *
from        ArtistNet
where       Name = 'Miro'
```

7. Вывести список художников, интересных клиенту G-man:

Если учесть, что ранее мы создали представление CustomerInterests, выводящее список клиентов и их интересов, то можно применить его, значительно упростив запрос

```
select * from CustomerInterests
where CustomerInterests.Customer = 'G-man'
```

8. Вывести список содержащихся в галерее произведений, закупленных с 12.01.2002 по 12.12.2002 включительно:

```

select      Title from WORK
where      WorkID in (
           select      WorkID
           from TRANS
           where      DateAcquired >= '12.01.2001' and
                    DateAcquired <= '12.01.2002' and
                    CustomerID is NULL
           )

```

9. Вывести список всех картин, купленных клиентом по имени *Sam Fisher*:

```

select      TRANS.PurchaseDate, WORK.Title
from      TRANS
          inner join vrdb.dbo.Work
          on TRANS.WorkID = Work.WorkID
where     TRANS.CustomerID = (
                               select      CustomerID
                               from      CUSTOMER
                               where      Name='Sam Fisher' );

```

10. Вывести список произведений и их художников, содержащихся в базе данных галереи:

```

select      W.Title, A.Name
from      WORK W
          inner join ARTIST A
          on WORK.ArtistID = ARTIST.ArtistID

```

11. Вывести список художников и количество работ каждого из них в галерее:

```

select      Artist.Name, COUNT(Work.WorkID) as WorksQuant
from      ARTIST
          inner join WORK
          on Artist.ArtistID = Work.ArtistID
where     Work.WorkID in
           (
             select      WorkID
             from TRANS
             where      CustomerID is NULL
           )
group by Artist.Name

```

12. Вывести список клиентов галереи и количество покупок, сделанных ими:

```

select      Customer.Name, COUNT(TRANS.TransactionID) AS q
from      Customer
          left join TRANS
          on Customer.CustomerID = TRANS.CustomerID
group by Customer.Name
order by q desc

```

13. Вывести среднюю стоимость картин в галерее:

```
select      AVG(AcquisitionPrice)
from        TRANS
where CustomerID in null
```

14. Вывести список распределения клиентов по регионам:

```
select      State, COUNT(CustomerID) AS CustQuant
from        Customer
group by    State
order by    CustQuant desc
```

15. Вывести список потенциальных клиентов (т.е. те, кто зарегистрирован в базе, но не купил ни одной картины):

```
select      Customer.Name
from        Customer
where CustomerID not in
            (
              select      CustomerID
              from        TRANS
            )
```

16. Вывести список работ, находящихся у клиентов:

```
select      Artist.Name, Work.Title, Customer.Name
from        ARTIST
            inner join WORK
              on Artist.ArtistID = Work.ArtistID
            inner join TRANS
              on Work.WorkID = Trans.WorkID
            inner join Customer
              on Customer.CustomerID = TRANS.CustomerID
where TRANS.TransactionID =
            (
              select      MAX(TransactionID)
              from        TRANS
              where TRANS.WorkID = Work.WorkID
            )
            and
            not TRANS.CustomerID is NULL
order by    Title
```

17. Протестируем хранимую процедуру. Добавить в базу данных картину, в качестве автора ссылающуюся на художника, записи о котором нету в базе данных. В этом случае процедура должна вернуть ошибку, указывающую на отсутствие указанного художника в базе данных:

```
exec add_work '99', 'work title', '4', 'description', '200';
```

18.Протестируем триггеры. Попробуем добавить сделку, в которой цена продажи картины составляет менее 90% от запрашиваемой цены:

```
INSERT INTO Trans (WorkID, CustomerID, DateAcquired, AcquisitionPrice, PurchaseDate, SalerPrice, AskingPrice) values (2, 1002, '11.24.2008', 600.00, '11.28.2008', 620.00, 950.00)
```

19.Список произведений, имеющих в галерее с одинаковым названием, но зарегистрированных под разными номерами:

```
SELECT W1.Title, W1.Copy
From WORK W1
WHERE W1.Title IN
(SELECT W2.Title
FROM WORK W2
WHERE W1.Title = W2.Title
AND W1.WorkID <> W2.WorkID);
```

20.Произведениями каких художников интересуются все без исключения клиенты:

```
SELECT A.Name
From ARTIST A
WHERE NOT EXISTS
(SELECT C.CustomerID
FROM CUSTOMER C
WHERE NOT EXISTS
(SELECT CI.CustomerID
FROM CUSTOMER_ARTIST_INT CI
WHERE C.CustomerID = CI.CustomerID
AND A.ArtistID = CI.ArtistID));
```

Контрольные вопросы и задания

1. Какие параметры можно настраивать в ERWin Data Modeler при генерации SQL-запроса, создающего разработанную базу данных?
2. Какими способами можно создать на сервере разработанную в среде СА ERWin Data Modeler схему базы данных?
3. Каким образом создается новая база данных MS SQL Server 2008 R2?
4. Какие методы аутентификации поддерживает MS SQL Server 2008 R2?
5. Разработайте и протестируйте несколько SQL-запросов, которые:
 - добавляют записи и таблицы в базу данных (INSERT, CREATE TABLE);
 - модифицируют записи и таблицы в базе данных (UPDATE, ALTER TABLE);
 - удаляют из базы данных записи и таблицы (DELETE, DROP TABLE);

6. Сделайте копию исходной разработанной базы данных и модифицируйте ее таким образом, чтобы в базе данных:
- могла храниться фотография картины (копии);
 - номер дома и номер квартиры содержались в отдельных полях информации о клиенте;
 - больше не хранились интересы клиентов.
7. Разработайте следующие запросы:
- по указанному коду клиента вывести список его интересов;
 - вывести список имеющихся и выставленных на продажу картин, информацию о каждой из них, включающую имя художника, написавшего ее;
 - добавьте картину в базу данных при помощи хранимой процедуры, производящей проверку наличия художника с указанным в запросе кодом.
8. Сделайте копию исходной разработанной базы данных и модифицируйте ее таким образом, чтобы ограничение ДопустимыеНациональности, было преобразовано в таблицу Национальность. Выполните модификацию путем выполнения скрипта, содержащего последовательность модифицирующих запросов.

ПРИЛОЖЕНИЕ I.

СКРИПТ ДЛЯ ГЕНЕРАЦИИ СХЕМЫ БАЗЫ ДАННЫХ

```
CREATE TABLE ARTIST
(
    ArtistID          int IDENTITY ( 1,1 ) ,
    Name              varchar(25)  NULL ,
    Nationality       varchar(30)  NULL
    CONSTRAINT NationalityList_101528401
        CHECK ( [Nationality]='русский' OR [Nationality]='немец' OR
[Nationality]='итальянец' ),
    BirthYear         numeric(4)   NULL ,
    DeceaseYear       numeric(4)   NULL
    CONSTRAINT BirthYearValidate_585202096
        CHECK ( BirthYear < DeceaseYear )
)
go

ALTER TABLE ARTIST
    ADD CONSTRAINT XPKARTIST PRIMARY KEY CLUSTERED (ArtistID ASC)
go

ALTER TABLE ARTIST
    ADD CONSTRAINT XAK1ARTIST UNIQUE (Name ASC)
go

CREATE TABLE CUSTOMER
(
    CustomerID        int IDENTITY ( 1,1 ) ,
    Email              varchar(50)  NOT NULL ,
    Name               varchar(25)  NULL ,
    Street             varchar(30)  NULL ,
    HouseNumber        varchar(6)   NULL ,
    City               varchar(35)  NULL ,
    Region             varchar(50)  NULL ,
    ZipPostalCode      varchar(6)   NULL ,
    Country            varchar(50)  NULL ,
    AreaCode           varchar(3)   NULL ,
    PhoneNumber        varchar(8)   NULL
)
go

ALTER TABLE CUSTOMER
    ADD CONSTRAINT XPKCUSTOMER PRIMARY KEY CLUSTERED (CustomerID ASC)
go

ALTER TABLE CUSTOMER
    ADD CONSTRAINT XAK1CUSTOMER UNIQUE (Email ASC)
go

CREATE TABLE CUSTOMER_ARTIST_INT
(
    CustomerID        int NOT NULL ,
    ArtistID          int NOT NULL
)
```

go

```
ALTER TABLE CUSTOMER_ARTIST_INT
    ADD CONSTRAINT XPKCUSTOMER_ARTIST_INT PRIMARY KEY CLUSTERED
(CustomerID ASC,ArtistID ASC)
go
```

```
CREATE TABLE TRANS
(
    TransactionID          int IDENTITY ( 1,1 ) ,
    DateAcquired           datetime NULL,
    AcquisitionPrice       numeric(8,2) NULL ,
    PurchaseDate           datetime NULL
    CONSTRAINT ValidTransDate_872289017
        CHECK ( DateAcquired <= PurchaseDate ),
    SalesPrice             numeric(8,2) NULL
    CONSTRAINT SalesPriceRange_2000292590
        CHECK ( [SalesPrice]>=(30000) AND [SalesPrice]<=(1500000) ),
    AskingPrice           numeric(8,2) NULL ,
    CustomerID            int NULL ,
    WorkID                int NOT NULL
)
go
```

```
ALTER TABLE TRANS
    ADD CONSTRAINT XPKTRANS PRIMARY KEY CLUSTERED (TransactionID ASC)
go
```

```
ALTER TABLE TRANS
    ADD CONSTRAINT XAK1TRANS UNIQUE (DateAcquired ASC,WorkID
ASC,CustomerID ASC)
go
```

```
CREATE TABLE WORK
(
    WorkID                int IDENTITY ( 1,1 ) ,
    Title                 varchar(25) NULL ,
    Copy                  varchar(8) NULL ,
    Description           varchar(100) NULL ,
    ArtistID             int NOT NULL
)
go
```

```
ALTER TABLE WORK
    ADD CONSTRAINT XPKWORK PRIMARY KEY CLUSTERED (WorkID ASC)
go
```

```
ALTER TABLE WORK
    ADD CONSTRAINT XAK1WORK UNIQUE (Title ASC,Copy ASC,ArtistID ASC)
go
```

```
CREATE VIEW BasicCustomerData (Name,AreaCode,PhoneNumber)
AS
SELECT CUSTOMER.Name,CUSTOMER.AreaCode,CUSTOMER.PhoneNumber
FROM CUSTOMER
```



```

go

CREATE VIEW CustomerPhone (Name, Phone)
AS
SELECT CUSTOMER.Name, '(' + AreaCode + ') ' + PhoneNumber
FROM CUSTOMER
go

CREATE VIEW
ArtistWorkNet (WorkID, Name, Title, Copy, AcquisitionPrice, SalesPrice, NetPrice
)
AS
SELECT
WORK.WorkID, ARTIST.Name, WORK.Title, WORK.Copy, TRANS.AcquisitionPrice, TRANS
.SalesPrice, SalesPrice - AcquisitionPrice
FROM WORK, ARTIST, TRANS
WHERE TRANS.WorkID = WORK.WorkID
AND
WORK.ArtistID = ARTIST.ArtistID
go

CREATE VIEW WorkNet (Name, Title, Copy, TotalNet)
AS
SELECT
ArtistWorkNet.Name, ArtistWorkNet.Title, ArtistWorkNet.Copy, sum (NetPrice)
FROM ArtistWorkNet
GROUP BY Artist, Title, Copy
go

CREATE VIEW ArtistNet (Name, TotalNet)
AS
SELECT ArtistWorkNet.Name, sum (NetPrice)
FROM ArtistWorkNet
GROUP BY Name
go

CREATE VIEW CustomerInterests (Artist, Customer)
AS
SELECT ARTIST.Name, CUSTOMER.Name
FROM ARTIST, CUSTOMER, CUSTOMER_ARTIST_INT
WHERE CUSTOMER.CustomerID = CUSTOMER_ARTIST_INT.CustomerID
AND
CUSTOMER_ARTIST_INT.ArtistID = ARTIST.ArtistID
go

ALTER TABLE CUSTOMER_ARTIST_INT
ADD CONSTRAINT Èíðáðãñóáðñý FOREIGN KEY (ArtistID) REFERENCES
ARTIST (ArtistID)
ON DELETE CASCADE
ON UPDATE CASCADE
go

ALTER TABLE CUSTOMER_ARTIST_INT
ADD CONSTRAINT R_6 FOREIGN KEY (CustomerID) REFERENCES
CUSTOMER (CustomerID)
ON DELETE CASCADE
ON UPDATE CASCADE

```

```

go

ALTER TABLE TRANS
    ADD CONSTRAINT R_1 FOREIGN KEY (CustomerID) REFERENCES
CUSTOMER(CustomerID)
    ON UPDATE CASCADE

go

ALTER TABLE TRANS
    ADD CONSTRAINT R_2 FOREIGN KEY (WorkID) REFERENCES WORK(WorkID)
    ON UPDATE CASCADE

go

ALTER TABLE WORK
    ADD CONSTRAINT R_3 FOREIGN KEY (ArtistID) REFERENCES
ARTIST(ArtistID)
    ON UPDATE CASCADE

go

CREATE PROCEDURE Add_WORK
(
    @ArtistID int, /*Художник должен уже присутствовать
                   в базе данных */
    @Title varchar(25),
    @Copy varchar(8),
    @Description varchar(1000),
    @AcquisitionPrice Numeric(6,2)
)

/* Хранимая процедура, регистрирующая приобретение галереей
произведения. Если произведение никогда раньше не появлялось
в галерее, в таблицу WORK добавляется новая строка.
В противном случае используется существующая строка. Далее
добавляется строка в таблицу TRANSACTION, а столбец
DateAcquired в ней устанавливается равным системной дате */

AS

    DECLARE @rowcount as int
    DECLARE @WorkID as int

/* Сначала следует убедиться, что ArtistID имеет допустимое
значение */
    SELECT @rowcount = Count(*)
    FROM ARTIST A
    WHERE A.ArtistID = @ArtistID

IF @rowcount = 0
/* Нет такого художника */
    BEGIN
        Print 'Нет художника с ID = ' + Str(@ArtistID)
        Print 'Обработка прервана.'
        return
    END

/* Теперь смотрим, есть ли это произведение в базе данных */
    SELECT @rowcount = Count(*)

```

```

FROM WORK W
WHERE W.ArtistID = @ArtistID and
      W.Title = @Title and
      W.Copy = @Copy

IF @rowcount = 0
    /* Произведения нет в базе, записываем его */
    INSERT INTO WORK (Title, Copy, Description, ArtistID)
    VALUES (@Title, @Copy, @Description, @ArtistID)

/* Получаем значение суррогатного ключа WorkID */
SELECT @WorkID = W.WorkID
FROM WORK W
WHERE W.ArtistID = @ArtistID and
      W.Title = @Title and
      W.Copy = @Copy

/* Вставляем новую строку в таблицу TRANSACTION */
INSERT INTO TRANS (DateAcquired, AcquisitionPrice, WorkID)
VALUES (GetDate(), @AcquisitionPrice, @WorkID)

RETURN

Go

CREATE TRIGGER RemoveDupTrans ON TRANS
FOR INSERT
AS
DECLARE          @WorkID as int,
                 @TransID as Int

SELECT          @WorkID = WorkID, @TransID = TransactionID
FROM inserted

/* Удаляем прочие доступные строки из таблицы TRANS */
DELETE FROM TRANS
WHERE WorkID = @WorkID
      AND CustomerID is null
      AND TransactionID <> @TransID

Go

-- Триггер, присваивающий значение по умолчанию --
CREATE TRIGGER SetAskingPrice ON TRANS
FOR INSERT
AS
DECLARE
    @WorkID          as int,
    @TransID         as int,
    @aqPrice         as money,
    @SumNetPrice     as money,
    @countPriorRows as int,
    @newPrice        as money,
    @netPrice        as int

/* Получаем новые значения WorkID и AcquisitionPrice */
SELECT          @WorkID = WorkID, @aqPrice = AcquisitionPrice,
    @TransID = TransactionID
FROM inserted

```

```

/* Проверяем, не появлялось ли произведение в галерее ранее */
SELECT * FROM TRANS T
WHERE T.WorkID = @WorkID

/* Это завершающий триггер, поэтому число строк будет включать
только что вставленную строку. Вычитаем единицу, что бы учесть это
*/
Set @countPriorRows = @@rowCount - 1

/* Устанавливаем newPrice и при необходимости корректируем ее значение
*/
Set @newPrice = 2 * @aqPrice
If @countPriorRows > 0
BEGIN
/* Функция AVG не подходит, поскольку в делителе она будет учтена
текущая строка. Используем @@rowCount - 1 из предыдущих вычислений
*/
SELECT @sumNetPrice = SUM(NetPrice)
FROM ArtistWorkNet AW
WHERE AW.WorkID = @WorkID
GROUP BY AW.WorkID

/*Если необходимо, корректируем netPrice */
If @netPrice < (@SumNetPrice / @countPriorRows) + @aqPrice
Set @newPrice = (@SumNetPrice / @countPriorRows) + @aqPrice
END

/* Теперь записываем в таблицу TRANS новое значение AskingPrice */
UPDATE TRANS
SET AskingPrice = @newPrice
WHERE TransactionID = @TransID

go

-- Триггер, реализующий процедуру обеспечения ссылочной целостности --
CREATE TRIGGER EnforceTransChild ON WORK
FOR INSERT
AS
DECLARE @newWorkID as int
/* Получаем новый WorkID */
SELECT @newWorkID = WorkID
FROM inserted
/* Проверяем, имеется ли доступная строка в таблице TRANS */
SELECT *
FROM TRANS
WHERE WorkID = @newWorkID
AND CustomerID is null
/* Если подходящей строки не найдено, вставляем строку */
If @@ROWCOUNT = 0
INSERT INTO TRANS (DateAcquired, WorkID)
VALUES (GETDATE(), @newWorkID)

go

-- Триггер, обновляющий представление --
CREATE TRIGGER CustomerNameUpdate ON CustomerInterests INSTEAD OF UPDATE
AS

```

```

DECLARE          @newName as Varchar(30),
                 @oldName as Varchar(30)

/* Получаем старое и новое имя */
SELECT          @newName = Customer
FROM            inserted

SELECT          @oldName = Customer
FROM            deleted

/* Подсчитываем количество синонимов в таблице CUSTOMER */
SELECT          *
FROM            CUSTOMER C1
WHERE           C1.Name = @oldName
               AND EXISTS
                 (
                   SELECT          *
                   FROM            CUSTOMER C2
                   WHERE           C1.Name = C2.Name
                               AND C1.CustomerID <> C2.CustomerID
                 )

/* Теперь обновляем данные о клиенте, если имя было уникальным */
If @@ROWCOUNT = 0
    UPDATE      CUSTOMER
    SET         CUSTOMER.Name = @newName
    WHERE      CUSTOMER.Name = @oldName

```

ПРИЛОЖЕНИЕ II. ЗАДАЧИ ДЛЯ САМОСТОЯТЕЛЬНОГО ИЗУЧЕНИЯ ЯЗЫКА SQL

Введение

Для выполнения представленных заданий Вам потребуется **Среда SQL Server Management Studio**.

Для подключения к серверу Вам необходимо:

- запустить программу: **Пуск-> Microsoft SQL Server 2008-> Среда SQL Server Management Studio**;
- в появившемся окне «Соединение с сервером» в поле «Имя сервера» ввести **SRV1** (рис. II.1). Нажать кнопку «Соединить»;

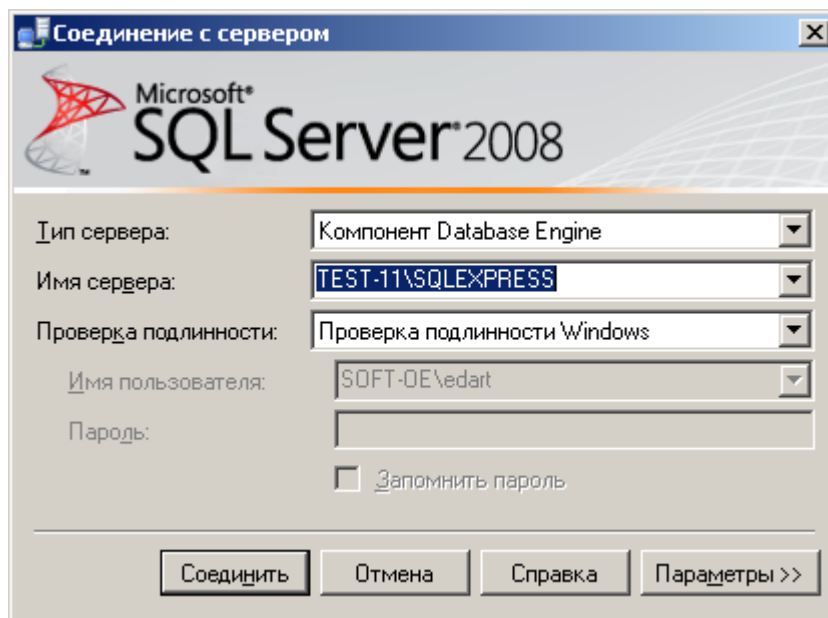


Рис. II.1. Окно подключения к экземпляру MS SQL сервера

- после успешного соединения появится окно среды с базой данных по умолчанию Aeroflot;
- в случае, если Вам необходима другая база данных, выберите ее из списка;
- нажмите кнопку «**Создать запрос**», чтобы начать работу по выполнению запросов.

При выполнении задания подготовьте документ отчета, в котором сохраните номер и текст задания, текст запроса и результат выполнения запроса.

Задание выполняется только одним оператором SELECT. Вы можете использовать вложенные подзапросы. Создание представлений, функций,

определяемых пользователем, временных таблиц или разделение запроса на несколько независимых запросов НЕ РАЗРЕШЕНО.

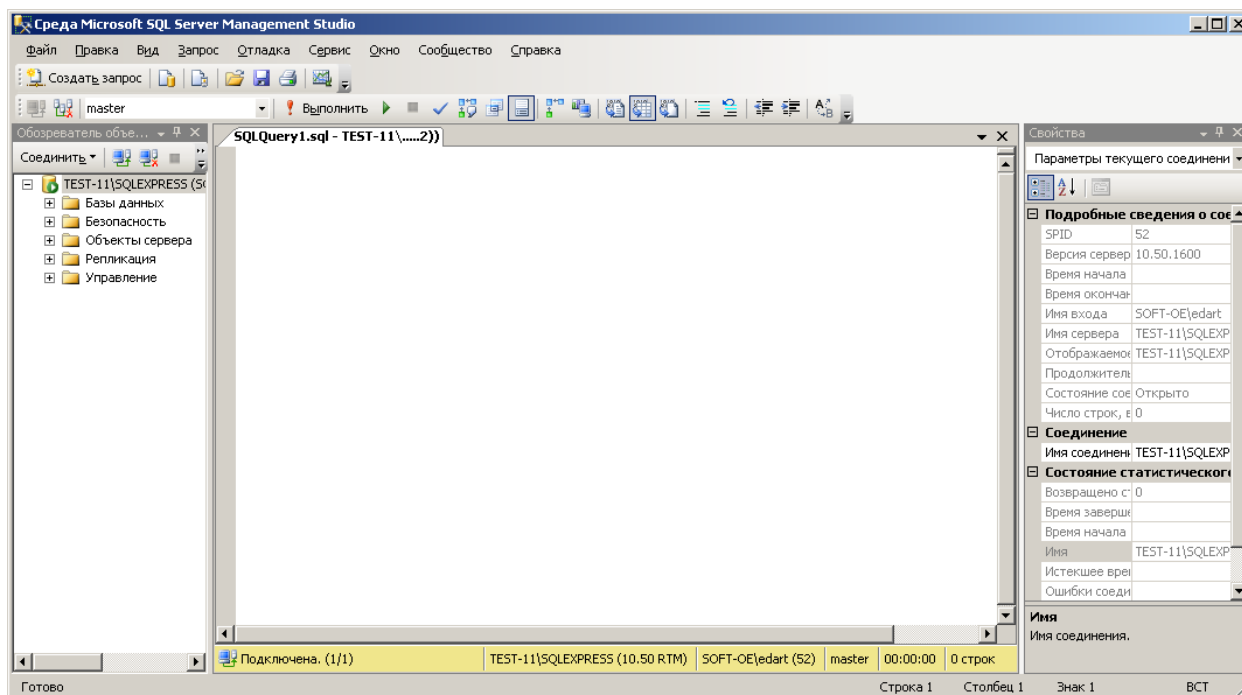


Рис. П.2. Окно среды для работы с базой данных

Для справки по использованию оператора SELECT обращайтесь к ресурсу [5, 6].

Ниже представлено несколько тестовых схем, скрипты создания с данными для которых можно взять у преподавателя или скачать самостоятельно по ссылке, указанной в источнике [7]. В этом случае Вы можете развернуть экземпляр SQL сервера на своем персональном компьютере, создать из скриптов базы данных с набором исходных тестовых данных и решать задачи самостоятельно дома.

Схема «Компьютерная фирма»

Схема БД состоит из четырех отношений:

- Product (maker, model, type)
- PC (code, model, speed, ram, hd, cd, price)
- Laptop (code, model, speed, ram, hd, screen, price)
- Printer (code, model, color, type, price)

Отношение Product представляет производителя (maker), номер модели (model) и тип (PC - ПК, Laptop - ПК-блокнот или Printer - принтер).

Предполагается, что номера моделей уникальны для всех производителей и типов продуктов.

В отношении PC для каждого номера модели, обозначающего ПК, указаны скорость -speed (процессора в мегагерцах), общий объем RAM (в мегабайтах), размер диска -hd (в гигабайтах), скорость считывающего устройства CD (например, 4x) и цена - price.

Отношение Laptop аналогично отношению PC за исключением того, что вместо скорости CD содержится размер экрана -screen (в дюймах).

В отношении Printer для каждой модели принтера указывается, является ли он цветным - color ('y', если цветной), тип принтера - type (лазерный - Laser, струйный - Jet или матричный - Matrix) и цена.

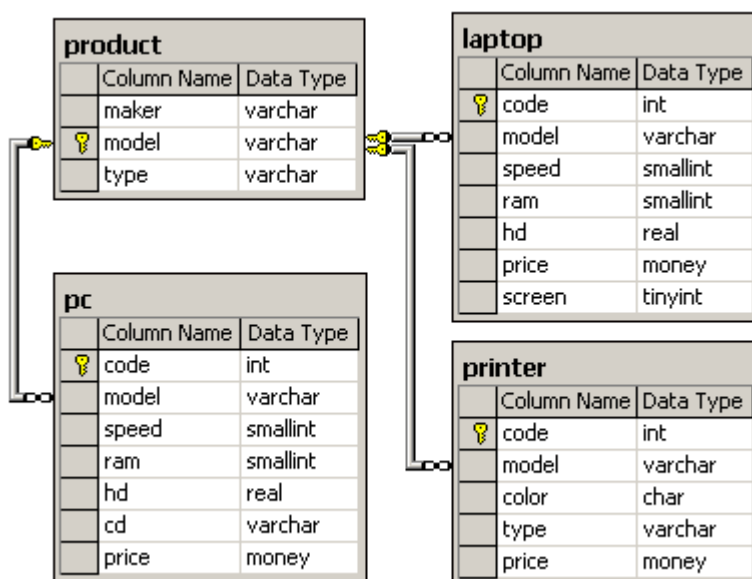


Рис. П.3. Схема базы данных «Компьютерная фирма»

Схема «Фирма вторсырья»

Фирма имеет несколько пунктов приема вторсырья. Каждый пункт получает деньги для их выдачи сдатчикам вторсырья. Сведения о получении денег на пункт приема записываются в таблицу:

Income_o(point, date, inc)

Первичным ключом является (point, date), т.е. прием денег (inc) производится не чаще одного раза в день. Сведения о выдаче денег за вторсырье записываются в таблицу:

Outcome_o(point, date, out)

В этой таблице также первичный ключ (point, date) гарантирует отчетность каждого пункта о выданных деньгах (out) не чаще одного раза в день.

В случае, когда приход и расход денег может фиксироваться несколько раз в день, используются таблицы (первичный ключ code):

Income(code, point, date, inc)

Outcome(code, point, date, out)

Income		
	Column Name	Data Type
🔑	code	int
	point	tinyint
	[date]	datetime
	inc	smallmoney

Outcome		
	Column Name	Data Type
🔑	code	int
	point	tinyint
	[date]	datetime
	out	smallmoney

Income_o		
	Column Name	Data Type
🔑	point	tinyint
🔑	[date]	datetime
	inc	smallmoney

Outcome_o		
	Column Name	Data Type
🔑	point	tinyint
🔑	[date]	datetime
	out	smallmoney

Рис. П.4. Схема базы данных «Фирма вторсырья»

Схема «Корабли»

Рассматривается БД кораблей, участвующих во второй мировой войне. Имеются следующие отношения:

- Classes (class, type, country, numGuns, bore, displacement)
- Ships (name, class, launched)
- Battles (name, date)
- Outcomes (ship, battle, result)

Корабли в «классах» построены по одному и тому же проекту, и классу присваивается название первого корабля этого класса.

Отношение Classes содержит имя класса, тип (bb для боевого (линейного) корабля или bc для боевого крейсера), страну, в которой построен корабль, число главных орудий, калибр орудий (диаметр ствола орудия в дюймах) и водоизмещение (вес в тоннах).

В отношении Ships записаны название корабля, имя его класса и год спуска на воду.

В отношении Battles включены название и дата битвы, в которой участвовали корабли, а в отношении Outcomes – результат участия данного корабля в битве (потоплен-sunk, поврежден - damaged или невредим - ОК).

Замечание. В отношении Outcomes могут входить корабли, отсутствующие в отношении Ships.

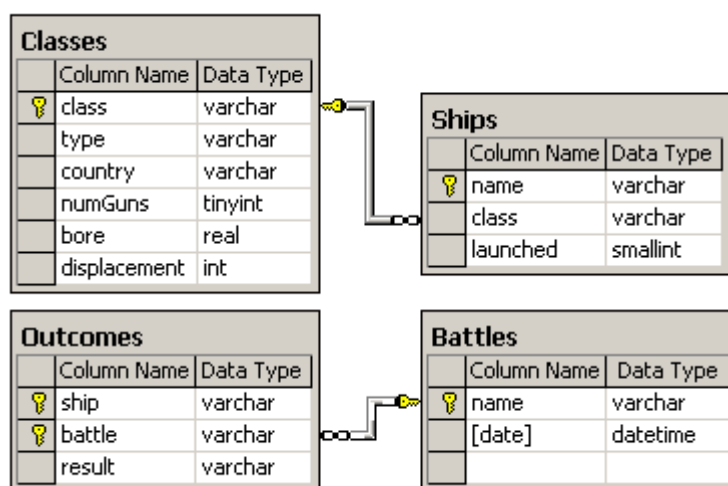


Рис. П.5. Схема базы данных «Корабли»

Схема «Аэрофлот»

Схема БД состоит из четырех отношений:

- Company (ID_comp, name)
- Trip(trip_no, id_comp, plane, town_from, town_to, time_out, time_in)
- Passenger(ID_psg, name)
- Pass_in_trip(trip_no, date, ID_psg, place)

Таблица Company содержит идентификатор и название компании, осуществляющей перевозку пассажиров.

Таблица Trip содержит информацию о рейсах: номер рейса, идентификатор компании, тип самолета, город отправления, город прибытия, время отправления и время прибытия.

Таблица Passenger содержит идентификатор и имя пассажира.

Таблица Pass_in_trip содержит информацию о полетах: номер рейса, дату вылета, идентификатор пассажира и место, на котором он сидел во время полета.

При этом следует иметь в виду, что

- длительность полета менее суток;
- время и дата учитываются относительно одного часового пояса;
- среди пассажиров могут быть однофамильцы (одинаковые имена);
- связи и ограничения показаны на схеме данных.

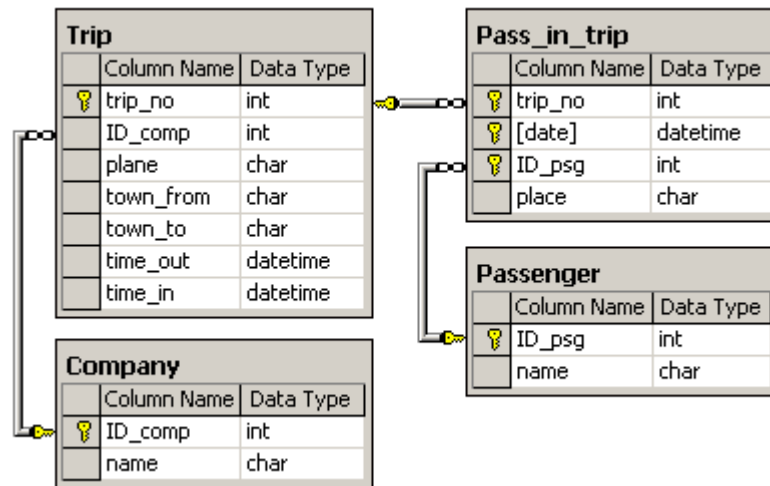


Рис. II.6. Схема базы данных «Аэрофлот»

Схема «Окраска»

Схема базы данных состоит из трех отношений:

- utQ (Q_ID int, Q_NAME varchar(35))
- utV (V_ID int, V_NAME varchar(35), V_COLOR char(1))
- utB (B_Q_ID int, B_V_ID int, B_VOL tinyint, B_DATETIME datetime)

Таблица utQ содержит идентификатор и название квадрата, цвет которого первоначально черный.

Таблица utV содержит идентификатор, название и цвет баллончика с краской.

Таблица utB содержит информацию об окраске квадрата баллончиком: идентификатор квадрата, идентификатор баллончика, количество краски и время окраски.

При этом следует иметь в виду, что:

- баллончики с краской могут быть трех цветов - красный V_COLOR='R', зеленый V_COLOR='G', голубой V_COLOR='B' (латинские буквы).
- объем баллончика равен 255 и первоначально он полный;
- цвет квадрата определяется по правилу RGB, т.е. R=0,G=0,B=0 - черный, R=255, G=255, B=255 - белый;
- запись в таблице закрасок utB уменьшает количество краски в баллончике на величину B_VOL и соответственно увеличивает количество краски в квадрате на эту же величину;
- значение $0 < B_VOL \leq 255$;

- количество краски одного цвета в квадрате не превышает 255, а количество краски в баллончике не может быть меньше нуля.

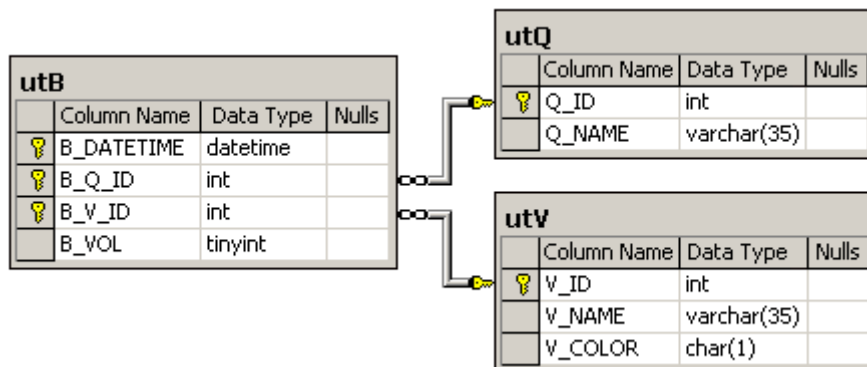


Рис. II.7. Схема базы данных «Окраска»

Варианты заданий

1. Выбрать все белые квадраты, которые окрашивались только из баллончиков, пустых к настоящему времени. Вывести имя квадрата.
2. Выбрать из таблицы Tgr такие города, названия которых содержат минимум 2 разные буквы из списка (a,e,i,o,u) и все имеющиеся в названии буквы из этого списка встречаются одинаковое число раз.
3. Вывести все записи из Outcome и Income, даты которых отстоят не менее чем на 2 календарных месяца от максимальной даты в обеих таблицах (т.е. при максимальной дате 2009-12-05 последняя выводимая дата должна быть меньше 2009-10-01). Выполнить помесечное разбиение этих записей, присвоив порядковый номер каждому месяцу (с учётом года), попавшему в выборку. Вывод: порядковый номер месяца, первый день месяца в формате "уууу-mm-dd", последний день месяца в формате "уууу-mm-dd", код записи, пункт, дата, сумма (для таблицы Outcome должна быть отрицательной)
4. Дима и Миша пользуются продуктами от одного и того же производителя. Тип Таниного принтера не такой, как у Вити, но признак "цветной или нет" - совпадает. Размер экрана Диминого ноутбука на 3 дюйма больше Олиного. Мишин ПК в 4 раза дороже Таниного принтера. Номера моделей Витиноного принтера и Олиного ноутбука отличаются только третьим символом. У Костиного ПК скорость процессора, как у Мишиного ПК; объем жесткого диска, как у Диминого ноутбука; объем памяти, как у Олиного ноутбука, а цена -

- как у Витиноного принтера. Вывести все возможные номера моделей Костиного ПК.
5. Для каждого значения скорости найдите среднюю стоимость ПК с такой же скоростью процессора. Вывести: скорость, среднюю цену.
 6. Для каждого значения скорости ПК, превышающего 600 МГц, определите среднюю цену компьютера с такой же скоростью. Вывести: speed, среднюю цену.
 7. Для каждого класса определите год, когда был спущен на воду первый корабль этого класса. Если год спуска на воду головного корабля неизвестен, определите минимальный год спуска на воду кораблей этого класса. Вывести: класс, год.
 8. Для каждого производителя найдите средний размер экрана выпускаемых им ПК-блокнотов. Вывести: maker, средний размер экрана.
 9. Для каждого пункта приема в базе с отчетностью не чаще одного раза в день подсчитать общее количество поступивших денег.
 10. Для каждого сражения определить день, являющийся последней пятницей месяца, в котором произошло данное сражение. Вывод: сражение, дата сражения, дата последней пятницы месяца. Даты представить в формате "уууу-mm-dd".
 11. Для каждой компании, перевозившей пассажиров, подсчитать время, которое провели в полете самолеты с пассажирами. Вывод: название компании, время в минутах.
 12. Для каждой пятой модели (в порядке возрастания номеров моделей) из таблицы Product определить тип продукции и среднюю цену модели.
 13. Для пятого по счету пассажира из числа вылетевших из Ростова в апреле 2003 года определить компанию, номер рейса и дату вылета.
 14. Для семи последовательных дней, начиная от минимальной даты, когда из Ростова было совершено максимальное число рейсов, определить число рейсов из Ростова. Вывод: дата, количество рейсов.
 15. Для таблицы Outcomes преобразовать названия кораблей, содержащих более одного пробела, следующим образом. Заменить все символы между первым и последним пробелами (исключая сами эти пробелы) на символы звездочки (*) в количестве, равном числу замененных символов. Вывод: название корабля, преобразованное название корабля.
 16. Для таблицы Product получить результирующий набор в виде таблицы со столбцами maker, pc, laptop и printer, в которой для каждого производителя требуется указать, производит он (yes) или нет (no) соответствующий тип продукции. В первом случае (yes) указать в

скобках без пробела количество имеющихся в наличии (т.е. находящихся в таблицах PC, Laptop и Printer) различных по номерам моделей соответствующего типа.

17. Какое максимальное количество черных квадратов можно было бы окрасить в белый цвет оставшейся краской?
18. Методом наименьших квадратов найти линейную зависимость мгновенного расхода краски от времени: $V = at + b$, где V - расход краски; t - время в секундах, отсчитываемое от первой окраски ($t = 0$). Вывод: a с 8-ю знаками после десятичной точки; b - с 2-мя знаками после десятичной точки
19. Найдите названия всех тех кораблей из базы данных, о которых можно определенно сказать, что они были спущены на воду до 1941 г.
20. Найдите названия кораблей, имеющих наибольшее число орудий среди всех имеющихся кораблей такого же водоизмещения (учесть корабли из таблицы Outcomes).
21. Найдите пары моделей PC, имеющих одинаковые скорость и RAM. В результате каждая пара указывается только один раз, т.е. (i,j) , но не (j,i) , Порядок вывода: модель с большим номером, модель с меньшим номером, скорость и RAM.
22. Найдите производителей принтеров, которые производят ПК с наименьшим объемом RAM и с самым быстрым процессором среди всех ПК, имеющих наименьший объем RAM. Вывести: Maker.
23. Найдите средний размер диска ПК (одно значение для всех) тех производителей, которые выпускают и принтеры. Вывести: средний размер HD.
24. Найти такие пункты приема, которые имеют в таблице Outcome записи на каждый рабочий день в течение некоторой недели (календарные дни, исключая субботу и воскресенье). Вывод: номер пункта, даты понедельника полной рабочей недели в формате "YYYY-MM-DD", суммарное значение out за эту рабочую неделю.
25. Написать запрос, который выводит все операции прихода и расхода из таблиц Income и Outcome в следующем виде: дата, порядковый номер записи за эту дату, пункт прихода, сумма прихода, пункт расхода, сумма расхода. При этом все операции прихода по всем пунктам, совершённые в течение одного дня, упорядочены по полю code, и так же все операции расхода упорядочены по полю code. В случае, если операций прихода/расхода за один день было не равное количество, выводить NULL в соответствующих колонках на месте недостающих операций.
26. Опираясь на утверждение, что одна секунда полета каждого пассажира приносит перевозчику 1 цент (0.01\$) дохода, провести ABC-анализ

привлекательности пассажиров вне зависимости от перевозчика. В основе метода лежит принцип Парето - 20% всех товаров дают 80% оборота. При выполнении анализа пассажиры делятся на 3 категории по степени их ценности: А, В и С. Порядок проведения анализа: 1) Рассчитываем долю дохода пассажира от общей суммы дохода, приносимой всеми пассажирами, с накопительным итогом. Доля с накопительным итогом высчитывается путём прибавления доли конкретного пассажира к сумме долей пассажиров с меньшей долей дохода (при одинаковой доле дохода меньший накопительный итог будет у пассажира, имя которого идет раньше в алфавитном порядке). 2) Выделяем категории А,В и С. Категория А – округленная до двух десятичных знаков накопительная доля с 0,00% по 80,00% включительно, категория В – с 80,01% до 95,00%, категория С – с 95,01% до 100%. Вывод: имя пассажира, сумма прибыли в долларах, доля с накопительным итогом в процентах (точность – 2 знака после запятой), категория (А, В или С – буквы латинские).

27. Определить имена разных пассажиров, которые летали только между двумя городами (туда и/или обратно).
28. Определить имена разных пассажиров, которым чаще других доводилось лететь на одном и том же месте. Вывод: имя и количество полетов на одном и том же месте.
29. Определить лидера по сумме выплат в соревновании между каждой парой пунктов с одинаковыми номерами из двух разных таблиц – outcome и outcome_o - на каждый день, когда осуществлялся прием вторсырья хотя бы на одном из них. Вывод: Номер пункта, дата, текст: – "once a day", если сумма выплат больше у фирмы с отчетностью один раз в день; – "more than once a day", если – у фирмы с отчетностью несколько раз в день; – "both", если сумма выплат одинакова.
30. Определить пассажиров, которые больше других времени провели в полетах. Вывод: имя пассажира, общее время в минутах, проведенное в полетах. Рекомендуются использовать SELECT TOP 1 WITH TIES.
31. Посчитать сумму цифр в номере каждой модели из таблицы Product Вывод: номер модели, сумма цифр.
32. Предполагая, что среди идентификаторов квадратов имеются пропуски, найти минимальный и максимальный "свободный" идентификатор в диапазоне между имеющимися максимальным и минимальным идентификаторами. Например, для последовательности идентификаторов квадратов 1,2,5,7 результат должен быть 3 и 6. Если пропусков нет, вместо каждого искомого значения выводить NULL.

33. При условии, что баллончики с красной краской использовались более одного раза, выбрать из них такие, которыми окрашены квадраты, имеющие голубую компоненту. Вывести название баллончика.
34. Сгруппировать все окраски по дням, месяцам и годам. Идентификатор каждой группы должен иметь вид "уууу" для года, "уууу-mm" для месяца и "уууу-mm-dd" для дня. Вывести только те группы, в которых количество различных моментов времени (b_datetime), когда выполнялась окраска, более 10. Вывод: идентификатор группы, суммарное количество потраченной краски.
35. Среди пассажиров, которые пользовались услугами не менее двух авиакомпаний, найти тех, кто совершил одинаковое количество полётов самолетами каждой из этих авиакомпаний. Вывести имена таких пассажиров.
36. Среди пассажиров, летавших на самолетах только одного типа, определить тех, кто прилетал в один и тот же город не менее 2-х раз. Вывести имена пассажиров.
37. Среди тех, кто пользуется услугами только какой-нибудь одной компании, определить имена разных пассажиров, летавших чаще других. Вывести: имя пассажира и число полетов.
38. Считая, что пункт самого первого вылета пассажира является местом жительства, найти не москвичей, которые прилетали в Москву более одного раза. Вывод: имя пассажира, количество полетов в Москву.
39. Укажите сражения, в которых участвовало по меньшей мере три корабля одной и той же страны.
40. Фирма открывает новые пункты по приему вторсырья. При открытии, каждому из них были выданы "подъемные" в размере 20 тыс. р. Каждому из пунктов была поставлена задача об увеличении первоначального капитала до 150%, с отчетностью - один раз в день. Используя одну только таблицу Outcome_o и при условии, что пункты работают с двойной накруткой, то есть на каждый выплаченный сдатчику рубль они получают доход 2 рубля, найти: для пунктов, справившихся с заданием, определить дату его выполнения и сумму денежных средств, полученных сверх плана на эту дату; для пунктов, которые не справились с заданием, определить на последнюю отчетную дату сумму денежных средств, недостающих до его выполнения. Вывод: пункт, дата выполнения (или последний день), сумма сверх плана (или недостающую сумму до плана).

ПРИЛОЖЕНИЕ III. РЕКОМЕНДАЦИИ К ВЫПОЛНЕНИЮ КУРСОВЫХ ПРОЕКТОВ

Требования к выполнению курсового проекта

Необходимо:

1. **Изучить** описание предметной области согласно выданному заданию. Если необходимо осуществить исследование предметной области, используя доступные Вам источники информации: книги, статьи, лекции, интернет-ресурсы, личный опыт.

2. **Сформулировать** требования к проекту и **определить** список вопросов (**не менее 16**), на которые система должна давать ответ с позиции изучаемой предметной областью. Список требований и вопросов согласовать с руководителем проекта.

3. **Проанализировать** требования к системе и **построить** инфологическую модель данных с использованием CA ERWin DataModeler или другим подобным инструментом.

4. **Создать** в среде MS SQL сервер (версия определяется преподавателем) **базу данных**, состоящую из необходимого количества таблиц (**количество таблиц определяется на стадии моделирования в соответствии с заданием и потребностями нормализации исходных таблиц**). Обязательно предоставить SQL-скрипт для создания базы данных.

5. **Заполнить** таблицы данными. **Каждая** таблица должна содержать не менее **10 – 15 записей**.

6. **Создать комплекс** запросов, форм, макросов, модулей и отчетов, позволяющих пользователю выполнить действия, указанные в индивидуальном варианте задания и определенные требованиями. Допускается создание вспомогательных запросов. Обязательным является создание сложных форм типа master-detail (т.е. отображение связи между родительской и дочерней таблицей «один-ко-многим»). **Предусмотреть создание хотя бы одного отчета средствами графического представления.**

7. **Создать главную форму приложения**, содержащую меню пользователя, соответствующее пунктам индивидуального варианта задания. Главная форма должна отображаться при запуске приложения. (Рекомендовано использование мультидокументного интерфейса MDI).

8. **Представить пояснительную** записку к курсовому проекту, все необходимые и используемые в ходе работы модели, приложение в виде исходных файлов проекта и в скомпилированном виде, базу данных и, при необходимости, инструкцию по использованию на CD или DVD. Диск должен быть подписан.

При защите курсового проекта студент должен продемонстрировать знания по теме выполненного курсового проекта, уметь пояснить выбранные решения и полученные результаты.

Пояснительная записка должна содержать:

1. Титульный лист.
2. Оглавление.
3. Задание на курсовой проект.
4. Введение (цели и назначение выполняемого курсового проекта). Требования к системе (список функций или возможностей системы). Список обязательных вопросов, на которые необходимо получить ответы (см. пункт 7).
5. Системный анализ предметной области с построением инфологической модели данных и модели функционирования приложения (приветствуется использование знаний, полученных в курсе ТИП и С, т.е. диаграммы DFD, IDEF0, IDEF3).
6. Описание структуры записей таблиц в виде SQL-описаний (т.е. DDL-инструкции для создания БД).
7. Не менее 10-15 записей по каждой таблице проекта.
8. По каждому запросу должно быть представлено SQL-описание, а также результат его выполнения.
9. Распечатку формы, содержащей меню курсового проекта, или главную форму.
10. Описание способов реализации каждого пункта задания и его вызова из меню.
11. Для каждой формы должно быть представлено ее изображение с описанием управляющих элементов. Для управляющих элементов должны быть представлены значения их свойств. Тексты созданных автором процедур для каждой формы и управляющих элементов. (Рекомендуется прилагать только собственные процедуры, сгенерированные автоматически не отображать).
12. Распечатку отчета и изображение отчета в режиме конструктора.
13. Тексты созданных автором процедур, используемых приложением в целом.
14. Заключение.
15. Список используемых источников.

Во введении нужно указать, какая среда выбрана для выполнения курсового проекта и почему. Студент должен также указать, какие дополнительные уточняющие предположения о характере данных он принял

при реализации своей работы. Оформление пояснительной записки производится в соответствии с требованиями, изложенными в работе [8].

Этапы выполнения курсового проекта

1. Системный анализ:

- изучить предметную область задания;
- составить словарь предметной области;
- сформулировать требования к проекту;
- сформулировать минимальный список вопросов, на которые нужно получить ответы;
- создать инфологическую модель;
- **отчитаться по этапу до 1 контрольной точки.**

2. Проектирование базы данных и приложения:

- спроектировать базу данных;
- создать базу данных, подготовить SQL-скрипт на создание БД;
- заполнить базу данных первичными сведениями;
- подготовить запросы, позволяющие получить ответы на сформулированные на 1 этапе вопросы;
- спроектировать приложение:
 - подготовить описание алгоритмов основной функциональности приложения;
 - предоставить структуру приложения;
- **Отчитаться по этапу до 2 контрольной точки.**

3. Реализация проекта и подготовка окончательного варианта записки:

- реализовать приложение проекта в соответствии с требованиями и заданием;
- подготовить пояснительную записку;
- **сдать готовый проект руководителю до 3 контрольной точки, т.е. до завершения семестра.**

Варианты заданий на курсовое проектирование

Варианты заданий взяты из работы [9].

1. Страховая компания

Описание предметной области

Вы работаете в страховой компании. Вашей задачей является отслеживание ее финансовой деятельности.

Компания имеет различные филиалы по всей стране. Каждый филиал характеризуется названием, адресом и телефоном. Деятельность компании организована следующим образом: к вам обращаются различные лица с целью заключения договора о страховании. В зависимости от принимаемых на страхование объектов и страхуемых рисков договор заключается по определенному виду страхования (например, страхование автотранспорта от угона, страхование домашнего имущества, добровольное медицинское страхование). При заключении договора вы фиксируете дату заключения, страховую сумму, вид страхования, тарифную ставку и филиал, в котором заключался договор.

Таблицы

Договоры (Номер договора, Дата заключения, Страховая сумма, Тарифная ставка, Код филиала, Код вида страхования).

Вид страхования (Код вида страхования, Наименование).

Филиал (Код филиала, Наименование филиала, Адрес, Телефон).

Развитие постановки задачи

Нужно учесть, что договоры заключают страховые агенты. Помимо информации об агентах (фамилия, имя, отчество, адрес, телефон), нужно еще хранить филиал, в котором работают агенты. Кроме того, исходя из базы данных, нужно иметь возможность рассчитывать заработную плату агентам. Заработная плата составляет некоторый процент от страхового платежа (страховой платеж – это страховая сумма, умноженная на тарифную ставку). Процент зависит от вида страхования, по которому заключен договор.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

2. Гостиница

Описание предметной области

Вы работаете в гостинице. Вашей задачей является отслеживание финансовой стороны ее работы.

Ваша деятельность организована следующим образом: гостиница предоставляет номера клиентам на определенный срок. Каждый номер характеризуется вместимостью, комфортностью (люкс, полуплюкс, обычный) и ценой. Вашими клиентами являются различные лица, о которых вы собираете определенную информацию (фамилия, имя, отчество и некоторый

комментарий). Сдача номера клиенту производится при наличии свободных мест в номерах, подходящих клиенту по указанным выше параметрам. При поселении фиксируется дата поселения. При выезде из гостиницы для каждого места запоминается дата освобождения.

Таблицы

Клиенты (Код клиента, Фамилия, Имя, Отчество, Паспортные данные, Комментарий).

Номера (Код номера, Номер, Количество человек, Комфортность, Цена).

Поселение (Код поселения, Код клиента, Код номера, Дата поселения, Дата освобождения, Примечание).

Развитие постановки задачи

Необходимо не только хранить информацию по факту сдачи номера клиенту, но и осуществлять бронирование номеров. Кроме того, для постоянных клиентов, а также для определенных категорий клиентов предусмотрена система скидок. Скидки могут суммироваться.

Внести в структуру таблиц изменения, учитывающие этот факт, и изменить существующие запросы. Добавить новые запросы.

3. Ломбард

Описание предметной области

Вы работаете в ломбарде. Вашей задачей является отслеживание финансовой стороны его работы.

Деятельность компании организована следующим образом: к вам обращаются различные лица с целью получения денежных средств под залог определенных товаров. У каждого из приходящих к вам клиентов вы запрашиваете фамилию, имя, отчество и другие паспортные данные. После оценивания стоимости принесенного в качестве залога товара вы определяете сумму, которую готовы выдать на руки клиенту, а также свои комиссионные. Кроме того, определяете срок возврата денег. Если клиент согласен, то ваши договоренности фиксируются в виде документа, деньги выдаются клиенту, а товар остается у вас. В случае, если в указанный срок не происходит возврата денег, товар переходит в вашу собственность.

Таблицы

Клиенты (Код клиента, Фамилия, Имя, Отчество, Номер паспорта, Серия паспорта, Дата выдачи паспорта).

Категории товаров (Код категории товаров, Название, Примечание).

Сдача в ломбард (Код, Код категории товаров, Код клиента, Описание товара, Дата сдачи, Дата возврата, Сумма, Комиссионные).

Развитие постановки задачи

После перехода прав собственности на товар ломбард может продавать товары по цене, меньшей или большей, чем была заявлена при сдаче. Цена может меняться несколько раз, в зависимости от ситуации на рынке. (Например, владелец ломбарда может устроить распродажу зимних вещей в конце зимы.) Помимо текущей цены, нужно хранить все возможные значения цены для данного товара.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

4. Реализация готовой продукции

Описание предметной области

Вы работаете в компании, занимающейся оптово-розничной продажей различных товаров. Вашей задачей является отслеживание финансовой стороны ее работы. Деятельность компании организована следующим образом: компания торгует товарами из определенного спектра. Каждый из этих товаров характеризуется наименованием, оптовой ценой, розничной ценой и справочной информацией. В вашу компанию обращаются покупатели. Для каждого из них вы запоминаете в базе данных стандартные данные (наименование, адрес, телефон, контактное лицо) и составляете по каждой сделке документ, запоминая наряду с покупателем количество купленного им товара и дату покупки.

Таблицы

Товары (Код товара, Наименование, Оптовая цена, Розничная цена, Описание).

Покупатели (Код покупателя, Телефон, Контактное лицо, Адрес).

Сделки (Код сделки, Дата сделки, Код товара, Количество, Код покупателя, Признак оптовой продажи).

Развитие постановки задачи

Теперь ситуация изменилась. Выяснилось, что обычно покупатели в рамках одной сделки покупают не один товар, а сразу несколько. Также компания решила предоставлять скидки в зависимости от количества закупленных товаров и их общей стоимости.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

5. Ведение заказов

Описание предметной области

Вы работаете в компании, занимающейся оптовой продажей различных

товаров. Вашей задачей является отслеживание финансовой стороны ее работы.

Деятельность компании организована следующим образом: компания торгует товарами из определенного спектра. Каждый из этих товаров характеризуется ценой, справочной информацией и признаком наличия или отсутствия доставки. В вашу компанию обращаются заказчики. Для каждого из них вы запоминаете в базе данных стандартные данные (наименование, адрес, телефон, контактное лицо) и составляете по каждой сделке документ, запоминая наряду с заказчиком количество купленного им товара и дату покупки.

Таблицы

Товары (Код товара, Цена, Доставка, Описание).

Заказчики (Код заказчика, Наименование, Адрес, Телефон, Контактное лицо).

Заказы (Код заказа, Код заказчика, Код товара, Количество, Дата).

Развитие постановки задачи.

Теперь ситуация изменилась. Выяснилось, что доставка разных товаров может производиться способами, различными по цене и скорости. Нужно хранить информацию о том, какими способами может осуществляться доставка каждого товара, и о том, какой вид доставки (а соответственно, и какую стоимость доставки) выбрал клиент при заключении сделки.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

6. Бюро по трудоустройству

Описание предметной области

Вы работаете в бюро по трудоустройству. Вашей задачей является отслеживание финансовой стороны работы компании.

Деятельность бюро организована следующим образом: бюро готово искать работников для различных работодателей и вакансии для ищущих работу специалистов различного профиля. При обращении к вам клиента-работодателя его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в базе данных. При обращении к вам клиента-соискателя его стандартные данные (фамилия, имя, отчество, квалификация, профессия, иные данные) также фиксируются в базе данных. По каждому факту удовлетворения интересов обеих сторон составляется документ. В документе указываются соискатель, работодатель, должность и комиссионные (доход бюро).

Таблицы

Работодатели (Код работодателя, Название, Вид деятельности, Адрес, Телефон).

Соискатели (Код соискателя, Фамилия, Имя, Отчество, Квалификация, Вид деятельности, Иные данные, Предполагаемый размер заработной платы).

Сделки (Код соискателя, Код работодателя, Должность, Комиссионные).

Развитие постановки задачи

Оказалось, что база данных не совсем точно описывает работу бюро. В базе фиксируется только сделка, а информация по открытым вакансиям не хранится. Кроме того, для автоматического поиска вариантов необходимо вести справочник «Виды деятельности».

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

7. Нотариальная контора

Описание предметной области

Вы работаете в нотариальной конторе. Вашей задачей является отслеживание финансовой стороны работы компании.

Деятельность нотариальной конторы организована следующим образом: фирма готова предоставить клиенту определенный комплекс услуг. Для наведения порядка вы формализовали эти услуги, составив их список с описанием каждой услуги. При обращении к вам клиента его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в базе данных. По каждому факту оказания услуги клиенту составляется документ. В документе указываются услуга, сумма сделки, комиссионные (доход конторы), описание сделки.

Таблицы

Клиенты (Код клиента, Название, Вид деятельности, Адрес, Телефон).

Сделки (Код сделки, Код клиента, Код услуги, Сумма, Комиссионные, Описание).

Услуги (Код услуги, Название, Описание).

Развитие постановки задачи

Теперь ситуация изменилась. В рамках одной сделки клиенту может быть оказано несколько услуг. Стоимость каждой услуги фиксирована. Кроме того, компания предоставляет в рамках одной сделки различные виды скидок. Скидки могут суммироваться.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

8. Фирма по продаже запчастей

Описание предметной области

Вы работаете в фирме, занимающейся продажей запасных частей для автомобилей. Вашей задачей является отслеживание финансовой стороны работы компании.

Основная часть деятельности, находящейся в вашем ведении, связана с работой с поставщиками. Фирма имеет определенный набор поставщиков, по каждому из которых известны название, адрес и телефон. У этих поставщиков вы приобретаете детали. Каждая деталь наряду с названием характеризуется артикулом и ценой (считаем цену постоянной). Некоторые из поставщиков могут поставлять одинаковые детали (один и тот же артикул). Каждый факт покупки запчастей у поставщика фиксируется в базе данных, причем обязательными для запоминания являются дата покупки и количество приобретенных деталей.

Таблицы

Поставщики (Код поставщика, Название, Адрес, Телефон).

Детали (Код детали, Название, Артикул, Цена, Примечание).

Поставки (Код поставщика, Код детали, Количество, Дата).

Развитие постановки задачи

Теперь ситуация изменилась. Выяснилось, что цена детали может меняться от поставки к поставке. Поставщики заранее ставят вас в известность о дате изменения цены и о ее новом значении. Нужно хранить не только текущее значение цены, но и всю историю изменения цен.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

9. Курсы повышения квалификации

Описание предметной области

Вы работаете в учебном заведении и занимаетесь организацией курсов повышения квалификации.

В вашем распоряжении имеются сведения о сформированных группах студентов. Группы формируются в зависимости от специальности и отделения. В каждую из них включено определенное количество студентов. Проведение занятий обеспечивает штат преподавателей. Для каждого из них у вас в базе данных зарегистрированы стандартные анкетные данные (фамилия, имя, отчество, телефон) и стаж работы. В результате распределения нагрузки вы получаете информацию о том, сколько часов занятий проводит каждый преподаватель с соответствующими группами. Кроме того, хранятся сведения о типе проводимых занятий (лекции, практика), предмете и оплате за 1 час.

Таблицы

Группы (Номер группы, Специальность, Отделение, Количество студентов).

Преподаватели (Код преподавателя, Фамилия, Имя, Отчество, Телефон, Стаж).

Нагрузка (Код преподавателя, Номер группы, Количество часов, Предмет, Тип занятия, Оплата).

Развитие постановки задачи

В результате работы с базой данных выяснилось, что размер почасовой оплаты зависит от предмета и типа занятия. Кроме того, каждый преподаватель может вести не все предметы, а только некоторые.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

10. Определение факультативов для студентов

Описание предметной области

Вы работаете в высшем учебном заведении и занимаетесь организацией факультативов.

В вашем распоряжении имеются сведения о студентах, включающие стандартные анкетные данные (фамилия, имя, отчество, адрес, телефон). Преподаватели вашей кафедры должны обеспечить проведение факультативных занятий по некоторым предметам. По каждому факультативу установлено определенное количество часов и вид проводимых занятий (лекции, практика, лабораторные работы). В результате работы со студентами у вас появляется информация о том, на какие факультативы записался каждый из них. Существует некоторый минимальный объем факультативных предметов, которые должен прослушать каждый студент. По окончании семестра вы заносите информацию об оценках, полученных студентами на экзаменах.

Таблицы

Студенты (Код студента, Фамилия, Имя, Отчество, Адрес, Телефон).

Предметы (Код предмета, Название, Объем лекций, Объем практик, Объем лабораторных работ).

Учебный план (Код студента, Код предмета, Оценка).

Развитие постановки задачи

Теперь ситуация изменилась. Выяснилось, что некоторые из факультативов могут длиться более одного семестра. В каждом семестре для предмета устанавливается объем лекций, практик и лабораторных работ в часах. В качестве итоговой оценки за предмет берется последняя оценка, полученная студентом. Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

11. Распределение учебной нагрузки

Описание предметной области

Вы работаете в высшем учебном заведении и занимаетесь распределением нагрузки между преподавателями кафедры.

В вашем распоряжении имеются сведения о преподавателях кафедры, включающие наряду с анкетными данными информацию об их ученой степени, занимаемой административной должности и стаже работы. Преподаватели вашей кафедры должны обеспечить проведение занятий по некоторым предметам. По каждому из них установлено определенное количество часов. В результате распределения нагрузки у вас должна получиться информация следующего рода: «Такой-то преподаватель проводит занятия по такому-то предмету с такой-то группой».

Таблицы

Преподаватели (Код преподавателя, Фамилия, Имя, Отчество, Ученая степень, Должность, Стаж).

Предметы (Код предмета, Название, Количество часов).

Нагрузка (Код преподавателя, Код предмета, Номер группы).

Развитие постановки задачи

Теперь ситуация изменилась. Выяснилось, что все проводимые занятия делятся на лекционные и практические. По каждому виду занятий устанавливается свое количество часов. Кроме того, данные о нагрузке нужно хранить несколько лет.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

12. Распределение дополнительных обязанностей

Описание предметной области

Вы работаете в коммерческой компании и занимаетесь распределением дополнительных разовых работ. Вашей задачей является отслеживание хода их выполнения.

Компания имеет штат сотрудников, каждый из которых получает определенный оклад. Время от времени возникает потребность в выполнении некоторой дополнительной работы, не входящей в круг основных должностных обязанностей сотрудников. Для наведения порядка в этой сфере деятельности вы проклассифицировали все виды дополнительных работ, определив сумму оплаты по факту их выполнения. При возникновении дополнительной работы определенного вида вы назначаете ответственного, фиксируя дату начала. По факту окончания вы фиксируете дату и выплачиваете дополнительную сумму к зарплате с учетом вашей классификации.

Таблицы

Сотрудники (Код сотрудника, Фамилия, Имя, Отчество, Оклад).

Виды работ (Код вида, Описание, Оплата за день).

Работы (Код сотрудника, Код вида, Дата начала, Дата окончания).

Развитие постановки задачи

Теперь ситуация изменилась. Выяснилось, что некоторые из дополнительных работ являются достаточно трудоемкими и, в то же время, срочными, что требует привлечения к их выполнению нескольких сотрудников. Также оказалось, что длительность работ в каждом конкретном случае различна. Соответственно, нужно заранее планировать длительность работы и количество сотрудников, занятых ее выполнением.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

13. Техническое обслуживание станков

Описание предметной области

Ваше предприятие занимается ремонтом станков и другого промышленного оборудования. Вашей задачей является отслеживание финансовой стороны деятельности предприятия.

Клиентами вашей компании являются промышленные предприятия, оснащенные различным сложным оборудованием. В случае поломок оборудования они обращаются к вам. Ремонтные работы в вашей компании организованы следующим образом: все станки проклассифицированы по странам-производителям, годам выпуска и маркам. Все виды ремонта отличаются названием, продолжительностью в днях, стоимостью. Исходя из этих данных, по каждому факту ремонта вы фиксируете вид станка и дату начала ремонта.

Таблицы

Виды станков (Код вида станка, Страна, Год выпуска, Марка).

Виды ремонта (Код ремонта, Название, Продолжительность, Стоимость, Примечания).

Ремонт (Код вида станка, Код ремонта, Дата начала, Примечания).

Развитие постановки задачи

Теперь ситуация изменилась. Несложный анализ показал, что нужно не просто подразделять станки по видам, а иметь информацию о том, сколько раз ремонтировался тот или иной конкретный станок.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

14. Туристическая фирма

Описание предметной области

Вы работаете в туристической компании, продающей путевки клиентам. Вашей задачей является отслеживание финансовой стороны деятельности фирмы.

Работа с клиентами в вашей компании организована следующим образом: у каждого клиента, пришедшего к вам, собираются некоторые стандартные данные – фамилия, имя, отчество, адрес, телефон. После этого сотрудники выясняют у клиента, где он хотел бы отдыхать. При этом ему демонстрируются различные варианты, включающие страну проживания, особенности местного климата, имеющиеся отели разного класса. Наряду с этим обсуждается возможная длительность пребывания и стоимость путевки. В случае если удалось договориться и найти для клиента приемлемый вариант, вы регистрируете факт продажи путевки (или путевок, если клиент покупает сразу несколько путевок), фиксируя дату отправления. Иногда вы решаете предоставить клиенту некоторую скидку.

Таблицы

Маршруты (Код маршрута, Страна, Климат, Длительность, Отель, Стоимость).

Путевки (Код маршрута, Код клиента, Дата отправления, Количество, Скидка).

Клиенты (Код клиента, Фамилия, Имя, Отчество, Адрес, Телефон).

Развитие постановки задачи

Теперь ситуация изменилась. Фирма работает с несколькими отелями в нескольких странах. Путевки продаются на одну, две или четыре недели. Стоимость путевки зависит от длительности тура и отеля. Скидки, которые предоставляет фирма, фиксированы. Например, при покупке более одной путевки предоставляется скидка 5%. Скидки могут суммироваться.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

15. Грузовые перевозки

Описание предметной области

Вы работаете в компании, занимающейся перевозками грузов. Вашей задачей является отслеживание стоимости перевозок с учетом заработной платы водителей.

Компания осуществляет перевозки по различным маршрутам. Для каждого маршрута вы определили некоторое название, вычислили примерное расстояние и установили некоторую оплату для водителя. Информация о водителях включает фамилию, имя, отчество и стаж. Для проведения расчетов вы храните полную информацию о перевозках (маршрут, водитель, даты

отправки и прибытия). По факту некоторых перевозок водителям выплачивается премия.

Таблицы

Маршруты (Код маршрута, Название, Дальность, Количество дней в пути, Оплата).

Водители (Код водителя, Фамилия, Имя, Отчество, Стаж).

Проделанная работа (Код маршрута, Код водителя, Дата отправки, Дата возвращения, Премия).

Развитие постановки задачи

Теперь ситуация изменилась. Ваша фирма решила ввести гибкую систему оплаты. Так, оплата водителям теперь должна зависеть не только от маршрута, но и от стажа. Кроме того, нужно учесть, что перевозку могут осуществлять два водителя.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

16. Учет телефонных переговоров

Описание предметной области

Вы работаете в коммерческой службе телефонной компании. Компания предоставляет абонентам телефонные линии для междугородних переговоров. Вашей задачей является отслеживание стоимости междугородних телефонных переговоров. Абонентами компании являются юридические лица, имеющие телефонную точку, ИНН, расчетный счет в банке. Стоимость переговоров зависит от города, в который осуществляется звонок, и времени суток (день, ночь). Каждый звонок абонента автоматически фиксируется в базе данных. При этом запоминаются город, дата, длительность разговора и время суток.

Таблицы

Абоненты (Код абонента, Номер телефона, ИНН, Адрес).

Города (Код города, Название, Тариф дневной, Тариф ночной).

Переговоры (Код переговоров, Код абонента, Код города, Дата, Количество минут, Время суток).

Развитие постановки задачи

Теперь ситуация изменилась. Ваша фирма решила ввести гибкую систему скидок. Так, стоимость минуты теперь уменьшается в зависимости от длительности разговора. Размер скидки для каждого города разный.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

17. Учет внутриофисных расходов

Описание предметной области

Вы работаете в бухгалтерии частной фирмы. Сотрудники фирмы имеют возможность осуществлять мелкие покупки для нужд фирмы, предоставляя в бухгалтерию товарный чек. Вашей задачей является отслеживание внутриофисных расходов.

Фирма состоит из отделов. Каждый отдел имеет название. В каждом отделе работает определенное количество сотрудников. Сотрудники могут осуществлять покупки в соответствии с видами расходов. Каждый вид расходов имеет название, некоторое описание и предельную сумму средств, которые могут быть потрачены в месяц. При каждой покупке сотрудник оформляет документ, где указывает вид расхода, дату, сумму и отдел.

Таблицы

Отделы (Код отдела, Название, Количество сотрудников).

Виды расходов (Код вида, Название, Описание, Предельная норма).

Расходы (Код расхода, Код вида, Код отдела, Сумма, Дата).

Развитие постановки задачи

Теперь ситуация изменилась. Оказалось, что нужно хранить данные о расходах не только в целом по отделу, но и по отдельным сотрудникам. Нормативы по расходованию средств устанавливаются не в целом, а по каждому отделу за каждый месяц. Не использованные в текущем месяце деньги могут быть использованы позже.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

18. Библиотека

Описание предметной области

Вы являетесь руководителем библиотеки. Ваша библиотека решила зарабатывать деньги, выдавая напрокат некоторые книги, имеющиеся в небольшом количестве экземпляров. Вашей задачей является отслеживание финансовых показателей работы.

У каждой книги, выдаваемой в прокат, есть название, автор, жанр. В зависимости от ценности книги вы определили для каждой из них залоговую стоимость (сумма, вносимая клиентом при взятии книги напрокат) и стоимость проката (сумма, которую клиент платит при возврате книги, получая назад залог). В библиотеку обращаются читатели. Все читатели регистрируются в картотеке, которая содержит стандартные анкетные данные (фамилия, имя, отчество, адрес, телефон). Каждый читатель может обращаться в библиотеку

несколько раз. Все обращения читателей фиксируются, при этом по каждому факту выдачи книги запоминаются дата выдачи и ожидаемая дата возврата.

Таблицы

Книги (Код книги, Название, Автор, Залоговая стоимость, Стоимость проката, Жанр).

Читатели (Код читателя, Фамилия, Имя, Отчество, Адрес, Телефон).

Выданные книги (Код книги, Код читателя, Дата выдачи, Дата возврата).

Развитие постановки задачи

Теперь ситуация изменилась. Несложный анализ показал, что стоимость проката книги должна зависеть не только от самой книги, но и от срока ее проката. Кроме того, необходимо добавить систему штрафов за вред, нанесенный книге, и систему скидок для некоторых категорий читателей.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

19. Прокат автомобилей

Описание предметной области

Вы являетесь руководителем коммерческой службы в фирме, занимающейся прокатом автомобилей. Вашей задачей является отслеживание финансовых показателей работы пункта проката.

В автопарк входит некоторое количество автомобилей различных марок, стоимостей и типов. Каждый автомобиль имеет свою стоимость проката. В пункт проката обращаются клиенты. Все клиенты проходят обязательную регистрацию, при которой о них собирается стандартная информация (фамилия, имя, отчество, адрес, телефон). Каждый клиент может обращаться в пункт проката несколько раз. Все обращения клиентов фиксируются, при этом по каждой сделке запоминаются дата выдачи и ожидаемая дата возврата.

Таблицы

Автомобили (Код автомобиля, Марка, Стоимость, Стоимость проката, Тип).

Клиенты (Код клиента, Фамилия, Имя, Отчество, Адрес, Телефон).

Выданные автомобили (Код автомобиля, Код клиента, Дата выдачи, Дата возврата).

Развитие постановки задачи

Теперь ситуация изменилась. Несложный анализ показал, что стоимость проката автомобиля должна зависеть не только от самого автомобиля, но и от срока его проката и от года выпуска. Также нужно ввести систему штрафов за возвращение автомобиля в ненадлежащем виде и систему скидок для постоянных клиентов.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

20. Выдача банком кредитов

Описание предметной области

Вы являетесь руководителем информационно-аналитического центра коммерческого банка. Одним из существенных видов деятельности банка является выдача кредитов юридическим лицам. Вашей задачей является отслеживание динамики работы кредитного отдела.

В зависимости от условий получения кредита, процентной ставки и срока возврата все кредитные операции делятся на несколько основных видов. Каждый из этих видов имеет свое название. Кредит может получить клиент, при регистрации предоставивший следующие сведения: название, вид собственности, адрес, телефон, контактное лицо.

Каждый факт выдачи кредита регистрируется банком, при этом фиксируются сумма кредита, клиент и дата выдачи.

Таблицы

Виды кредитов (Код вида, Название, Условия получения, Ставка, Срок).

Клиенты (Код клиента, Название, Вид собственности, Адрес, Телефон, Контактное лицо).

Кредиты (Код вида, Код клиента, Сумма, Дата выдачи).

Развитие постановки задачи

Теперь ситуация изменилась. После проведения различных исследований выяснилось, что используемая система не позволяет отслеживать динамику возврата кредитов. Для устранения этого недостатка вы приняли решение учитывать в системе еще и дату фактического возврата денег. Нужно еще учесть, что кредит может гаситься частями, и за задержку возврата кредита начисляются штрафы.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

21. Инвестирование свободных средств

Описание предметной области

Вы являетесь руководителем аналитического центра инвестиционной компании, занимающейся вложением денежных средств в ценные бумаги.

Ваши клиенты – предприятия, которые доверяют управлять их свободными денежными средствами на определенный период. Вам необходимо выбрать вид ценных бумаг, которые позволят получить прибыль и компании, и

клиенту. При работе с клиентом для вас весьма существенной является информация о предприятии – название, вид собственности, адрес и телефон.

Таблицы

Ценные бумаги (Код ценной бумаги, Минимальная сумма сделки, Рейтинг, Доходность за прошлый год, Дополнительная информация).

Инвестиции (Код инвестиции, Код ценной бумаги, Код клиента, Котировка, Дата покупки, Дата продажи).

Клиенты (Код клиента, Название, Вид собственности, Адрес, Телефон).

Развитие постановки задачи

При эксплуатации базы данных стало понятно, что необходимо хранить историю котировок каждой ценной бумаги. Кроме того, помимо вложений в ценные бумаги, существует возможность вкладывать деньги в банковские депозиты.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

22. Занятость актеров театра

Описание предметной области

Вы являетесь коммерческим директором театра, и в ваши обязанности входит вся организационно-финансовая работа, связанная с привлечением актеров и заключением контрактов.

Вы организовали дело следующим образом: каждый год театр осуществляет постановку различных спектаклей. Каждый спектакль имеет определенный бюджет. Для участия в конкретных постановках в определенных ролях привлекаются актеры. С каждым из актеров вы заключаете персональный контракт на определенную сумму. Каждый из актеров имеет некоторый стаж работы, некоторые из них удостоены различных наград и званий.

Таблицы

Актеры (Код актера, Фамилия, Имя, Отчество, Звание, Стаж).

Спектакли (Код спектакля, Название, Год постановки, Бюджет).

Занятость актеров в спектакле (Код актера, Код спектакля, Роль, Стоимость годового контракта).

Развитие постановки задачи

В результате эксплуатации базы данных выяснилось, что в рамках одного спектакля на одну и ту же роль привлекается несколько актеров. Контракт определяет базовую зарплату актера, а по итогам реально отыгранных спектаклей актеру назначается премия. Кроме того, в базе данных нужно хранить информацию за несколько лет.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

23. Платная поликлиника

Описание предметной области

Вы являетесь руководителем службы планирования платной поликлиники. Вашей задачей является отслеживание финансовых показателей работы поликлиники.

В поликлинике работают врачи различных специальностей, имеющие разную квалификацию. Каждый день в поликлинику обращаются больные. Все они проходят обязательную регистрацию, при которой в базу данных заносятся стандартные анкетные данные (фамилия, имя, отчество, год рождения). Каждый больной может обращаться в поликлинику несколько раз, нуждаясь в различной медицинской помощи. Все обращения больных фиксируются, при этом устанавливается диагноз, определяется стоимость лечения, запоминается дата обращения.

Таблицы

Врачи (Код врача, Фамилия, Имя, Отчество, Специальность, Категория).

Пациенты (Код пациента, Фамилия, Имя, Отчество, Год рождения).

Обращения (Код обращения, Код врача, Код пациента, Дата обращения, Диагноз, Стоимость лечения).

Развитие постановки задачи

В результате эксплуатации базы данных выяснилось, что при обращении в поликлинику пациент обследуется и проходит лечение у разных специалистов. Общая стоимость лечения зависит от стоимости тех консультаций и процедур, которые назначены пациенту. Кроме того, для определенных категорий граждан предусмотрены скидки.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

24. Анализ динамики показателей финансовой отчетности различных предприятий

Описание предметной области

Вы являетесь руководителем информационно-аналитического центра крупного холдинга. Вашей задачей является отслеживание динамики показателей для предприятий холдинга.

В структуру холдинга входят несколько предприятий. Каждое предприятие имеет стандартные характеристики (название, реквизиты, телефон,

контактное лицо). Работа предприятия может быть оценена следующим образом: в начале каждого отчетного периода на основе финансовой отчетности вычисляется по неким формулам определенный набор показателей. Важность показателей характеризуется некоторыми числовыми константами. Значение каждого показателя измеряется в некоторой системе единиц.

Таблицы

Показатели (Код показателя, Название, Важность, Единица измерения).

Предприятия (Код предприятия, Название, Банковские реквизиты, Телефон, Контактное лицо).

Динамика показателей (Код показателя, Код предприятия, Дата, Значение).

Развитие постановки задачи

В результате эксплуатации базы данных выяснилось, что некоторые показатели считаются в рублях, некоторые в долларах, некоторые в евро. Для удобства работы с показателями нужно хранить изменения курсов валют относительно друг друга.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

25. Учет телекомпанией стоимости прошедшей в эфире рекламы

Описание предметной области

Вы являетесь руководителем коммерческой службы телевизионной компании. Вашей задачей является отслеживание расчетов, связанных с прохождением рекламы в телеэфире.

Работа построена следующим образом: заказчики просят поместить свою рекламу в определенной передаче в определенный день.

Каждый рекламный ролик имеет определенную продолжительность. Для каждой организации-заказчика известны банковские реквизиты, телефон и контактное лицо для проведения переговоров. Передачи имеют определенный рейтинг. Стоимость минуты рекламы в каждой конкретной передаче известна (определяется коммерческой службой, исходя из рейтинга передачи и прочих соображений).

Таблицы

Передачи (Код передачи, Название, Рейтинг, Стоимость минуты).

Реклама (Код рекламы, Код передачи, Код заказчика, Дата, Длительность в минутах).

Заказчики (Код заказчика, Название, Банковские реквизиты, Телефон, Контактное лицо).

Развитие постановки задачи

В результате эксплуатации базы данных выяснилось, что необходимо также хранить информацию об агентах, заключивших договоры на рекламу. Зарплата рекламных агентов составляет некоторый процент от общей стоимости рекламы, прошедшей в эфире.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

26. Интернет-магазин

Описание предметной области

Вы являетесь сотрудником коммерческого отдела компании, продающей различные товары через Интернет. Вашей задачей является отслеживание финансовой составляющей ее работы.

Работа компании организована следующим образом: на Интернет-сайте представлены (выставлены на продажу) некоторые товары. Каждый из них имеет некоторое название, цену и единицу измерения (штуки, килограммы, литры). Для проведения исследований и оптимизации работы магазина вы пытаетесь собирать данные с клиентов. При этом для вас определяющее значение имеют стандартные анкетные данные, а также телефон и адрес электронной почты для связи. В случае приобретения товаров на сумму свыше 5000 р. клиент переходит в категорию постоянных и получает скидку на каждую покупку в размере 2%. По каждому факту продажи вы автоматически фиксируете клиента, товары, количество, дату продажи, дату доставки.

Таблицы

Товары (Код товара, Название, Цена, Единица измерения).

Клиенты (Код клиента, Фамилия, Имя, Отчество, Адрес, Телефон, email, Признак постоянного клиента).

Продажи (Код продажи, Код товара, Код клиента, Дата продажи, Дата доставки, Количество).

Развитие постановки задачи

В результате эксплуатации базы данных выяснилось, что иногда возникают проблемы, связанные с нехваткой информации о наличии нужных товаров на складе в нужном количестве. Кроме того, обычно клиенты в рамках одного заказа покупают не один вид товара, а несколько видов. Исходя из суммарной стоимости заказа, компания предоставляет дополнительные скидки.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

27. Ювелирная мастерская

Описание предметной области

Вы работаете в ювелирной мастерской, осуществляющей изготовление ювелирных изделий для частных лиц на заказ. Вы работаете с определенными материалами (платина, золото, серебро, различные драгоценные камни и т.д.). При обращении потенциального клиента вы определяете, какое именно изделие ему необходимо. Все изготавливаемые изделия принадлежат к некоторому типу (серьги, кольца, броши, браслеты), выполнены из определенного материала, имеют некоторый вес и цену (включающую стоимость материалов и работы).

Таблицы

Изделия (Код изделия, Название, Тип, Код материала, Вес, Цена).

Материалы (Код материала, Название, Цена за грамм).

Продажи (Код изделия, Дата продажи, Фамилия покупателя, Имя покупателя, Отчество покупателя).

Развитие постановки задачи

В процессе опытной эксплуатации базы данных выяснилось, что ювелирное изделие может состоять из нескольких материалов. Кроме того, постоянным клиентам мастерская предоставляет скидки.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

28. Парикмахерская

Описание предметной области

Вы работаете в парикмахерской, обслуживающей клиентов в соответствии с их пожеланиями и некоторым каталогом различных видов стрижки. Так, для каждой стрижки определены название, принадлежность полу (мужская, женская), стоимость работы. Для наведения порядка вы, по мере возможности, составляете базу данных клиентов, запоминая их анкетные данные (фамилия, имя, отчество). Начиная с пятой стрижки, клиент переходит в категорию постоянных и получает скидку в 3% при каждой последующей стрижке. После того как закончена очередная работа, документом фиксируются стрижка, клиент и дата производства работ.

Таблицы

Стрижки (Код стрижки, Название, Пол, Стоимость).

Клиенты (Код клиента, Фамилия, Имя, Отчество, Пол, Признак постоянного клиента).

Работа (Код работы, Код стрижки, Код клиента, Дата).

Развитие постановки задачи

Теперь ситуация изменилась. У парикмахерской появился филиал, и вы хотели бы видеть, в том числе, и отдельную статистику по филиалам. Кроме того, стоимость стрижки может меняться с течением времени. Нужно хранить не только последнюю цену, но и все данные по изменению цены стрижки.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

29. Химчистка

Описание предметной области

Вы работаете в химчистке, осуществляющей прием у населения вещей для выведения пятен. Для наведения порядка вы, по мере возможности, составляете базу данных клиентов, запоминая их анкетные данные (фамилия, имя, отчество). Начиная с третьего обращения клиент переходит в категорию постоянных и получает скидку в 3% при чистке каждой последующей вещи. Все оказываемые услуги подразделяются на виды, имеющие название, тип и стоимость, зависящую от сложности работ. Работа с клиентом первоначально состоит в определении объема работ, вида услуги и, соответственно, ее стоимости. Если клиент согласен, он оставляет вещь (при этом фиксируются услуга, клиент и дата приема) и забирает ее после обработки (при этом фиксируется дата возврата).

Таблицы

Виды услуг (Код вида услуг, Название, Тип, Стоимость).

Клиенты (Код клиента, Фамилия, Имя, Отчество, Признак постоянного клиента).

Услуги (Код услуги, Код вида услуги, Код клиента, Дата приема, Дата возврата).

Развитие постановки задачи

Теперь ситуация изменилась. У химчистки появился филиал, и вы хотели бы видеть, в том числе, и отдельную статистику по филиалам. Кроме того, вы решили делать надбавки за срочность и сложность работ.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

30. Сдача в аренду торговых площадей

Описание предметной области

Вы работаете в крупном торговом центре, сдающем в аренду коммерсантам свои торговые площади.

Вашей задачей является наведение порядка в финансовой сфере работы торгового центра.

Работа торгового центра построена следующим образом: в результате планирования вы определили некоторое количество торговых точек в пределах здания, которые могут сдаваться в аренду. Для каждой из торговых точек важными данными являются этаж, площадь, наличие кондиционера и стоимость аренды в день. Со всех потенциальных клиентов вы собираете стандартные данные (название, адрес, телефон, реквизиты, контактное лицо). При появлении потенциального клиента вы показываете ему имеющиеся свободные площади. При достижении соглашения вы оформляете договор, фиксируя в базе данных торговую точку, клиента, период (срок) аренды.

Таблицы

Торговые точки (Код торговой точки, Этаж, Площадь, Наличие кондиционера, Стоимость аренды в день).

Клиенты (Код клиента, Название, Реквизиты, Адрес, Телефон, Контактное лицо).

Аренда (Код аренды, Код торговой точки, Код клиента, Дата начала, Дата окончания).

Развитие постановки задачи

В результате эксплуатации базы данных выяснилось, что некоторые клиенты арендуют сразу несколько торговых точек. Помимо этого, вам необходимо собирать информацию о ежемесячных платежах, поступающих от арендаторов.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Карпова, Т.С. Базы данных: модели, разработка, реализация / Т.С. Карпова. – СПб.: Питер, 2001.
2. Кренке, Д. Теория и практика построения баз данных / Д. Кренке; пер. с англ. – 9-е изд. – СПб.: Питер, 2005.
3. Маклаков, С.В. BPWin и ERWin. CASE-средства разработки информационных систем / С.В. Маклаков. – М.: Диалог-МИФИ, 2000.
4. Бурков, А.В. Проектирование информационных систем в Microsoft SQL Server 2008 и Visual Studio 2008 / А.В. Бурков. – <http://www.intuit.ru/department/se/pisqlvs2008/>.
5. Моисеенко, С.И. SQL задачи и решения / С.И. Моисеенко. – СПб.: Питер, 2006.
6. Моисеенко, С.И. SQL задачи и решения / С.И. Моисеенко. – <http://www.sql-tutorial.ru/>.
7. Рекомендуемые ссылки сайта «Упражнения на SQL». – <http://sql-ex.ru/links.php>.
8. Требования к оформлению курсовых и дипломных работ: метод. указания / сост. С. П. Бобков, Н. И. Терехин, О. Н. Ястребцев; Иван. гос. хим.-технол. ун-т. – Иваново, 2003.
9. Миндалев, И.В. Теория экономических информационных систем. Электронный учебно-методический комплекс / И.В. Миндалев – <http://www.kgau.ru/istiki/teis/index.html>.

Редактор О. А. Соловьева

Подписано в печать 9.06.2012. Формат 60x84 1/16. Бумага писчая.
Усл. п.л. 5,58. Уч.-изд. л. 6,19. Тираж 50 экз. Заказ

Отпечатано на полиграфическом оборудовании кафедры экономики и финансов ФБГОУ ВПО «ИГХТУ»

Ивановский государственный химико-технологический университет
153000, г. Иваново, пр. Ф. Энгельса, 7