МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)

Д.А. ПОПОВА-КОВАРЦЕВА, Е.В. СОПЧЕНКО

ОСНОВЫ СОВРЕМЕННЫХ ТЕХНОЛОГИЙ БАЗ ДАННЫХ

Рекомендовано редакционно-издательским советом федерального государственного автономного образовательного учреждения высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева» в качестве учебного пособия для обучающихся по основной образовательной программе высшего образования по направлению подготовки 02.03.02 Фундаментальная информатика и информационные технологии

САМАРА
Издательство Самарского университета
2019

УДК 004(075)
ББК 32.97я7
П58

Рецензенты: д-р техн. наук, проф. С.А. П р о х о р о в ;
д-р техн. наук, проф. А.А. Т ю г а ш е в

Попова-Коварцева, Дарья Александровна

П58 **Основы современных технологий баз данных:** учеб. пособие / *Д.А. Попова-Коварцева, Е.В. Сопченко*. – Самара: Изд-во Самарского университета, 2019. – 92 с.: ил.

ISBN 978-5-7883-1453-2

В пособии рассмотрены практические вопросы проектирования и реализации баз данных, а именно:

- принципы проектирования модели базы данных;
- создание логической и физической моделей базы данных с помощью программного средства автоматизации процесса моделирования БД;
- создание базы данных с помощью структурированного языка запросов SQL, принципы использования языка описания данных, языка манипулирования данными, языка запросов.

Теоретические сведения сопровождаются многочисленными примерами. Для закрепления теоретического материала приводится описание лабораторных работ по темам, рассмотренным в пособии.

Подготовлено на кафедре программных систем Самарского университета и предназначено для студентов дневного отделения, обучающихся по направлению подготовки бакалавров «Фундаментальная информатика и информационные технологии», изучающих курсы «Базы данных» и «Проектирование баз данных». Может быть полезным для студентов других направлений подготовки, изучающих технологии баз данных.

УДК 004(075)
ББК 32.97я7

ISBN 978-5-7883-1453-2 © Самарский университет, 2019

ОГЛАВЛЕНИЕ

СПИСОК ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ

БД – база данных

СУБД – система управления базами данных

CASE (от англ. computer-aided software engineering) – набор инструментов и методов программной инженерии для проектирования программного обеспечения

SQL (от англ. structured query language) – структурированный язык запросов

DDL (data definition language) – язык описания данных, используемый для описания структуры баз данных

DML (data manipulation language) – язык управления (манипулирования) данными, используемый для получения, вставки, удаления или изменения данных в базах данных

IDEF1X – методология моделирования баз данных на основе модели «сущность-связь»

ВВЕДЕНИЕ

Прогресс носителей информации привел к тому, что базы данных стали обычным компонентом вычислительных систем любой мощности. Современный специалист по информационным технологиям просто обязан иметь познания в этой области.

В учебном пособии изложены основы современных технологий баз данных. Рассмотрены вопросы проектирования реляционной базы данных, создания структуры БД и обработки данных в ней средствами структурированного языка запросов SQL. Книга ориентирована на использование в учебном процессе, поэтому теоретический материал дополнен примерами по каждой теме. Пособие состоит из двух глав, каждая из которых соответствует одной теме. В конце каждой главы приводятся контрольные вопросы.

В главе 1 приводятся основные сведения о CASE-технологиях, применяемых в проектировании баз данных. Описаны основная цель, характеристики CASE-средств и их классификация. Рассмотрен пример проектирования реляционной базы данных с помощью CASE-средства CA ERwin Data Modeler, описаны преимущества от использования данного графического средства моделирования данных и его функциональные возможности.

Глава 2 посвящена описанию структурированного языка запросов SQL. Описаны составляющие языка SQL: язык определения данных DDL и язык манипулирования данными DML. Описан синтаксис и примеры использования операторов языка DDL: создание, удаление, изменение таблиц и представлений. Описан синтаксис операторов языка манипулирования данными DML: добав-

ление, удаление и изменение записей таблицы. Большое внимание уделено оператору выборки данных SELECT. На примере небольшой базы данных некоторой авиакомпании приводятся варианты использования всевозможных конструкций команды SELECT, описаны операторы, предикаты и агрегатные функции, используемые в ней. Рассмотрен механизм представлений.

В приложении приводятся описания лабораторных работ по темам, рассмотренным в пособии. Каждая лабораторная работа содержит перечень заданий, необходимых для ее выполнения. Приводятся варианты предметных областей с описанием функциональных требований для моделирования их в ходе выполнения лабораторного практикума.

Глава 1. CASE–ТЕХНОЛОГИИ В ПРОЕКТИРОВАНИИ БАЗ ДАННЫХ

1.1. Основные сведения о CASE–средствах

Термин CASE (Computer Aided Software Engineering) дословно переводится как разработка программного обеспечения с помощью компьютера. В настоящее время этот термин получил более широкий смысл, означающий автоматизацию разработки информационных систем [1].

Современные CASE–средства охватывают обширную область поддержки многочисленных технологий проектирования информационных систем: от простых средств анализа и документирования до полномасштабных средств автоматизации, покрывающих весь жизненный цикл программного обеспечения (ПО) [2].

Основная цель CASE–систем и средств состоит в том, чтобы отделить проектирование программного обеспечения от его кодирования и последующих этапов разработки (тестирование, документирование и т.п.), а также автоматизировать весь процесс создания программных систем. Многие современные CASE–средства предоставляют возможности для моделирования практически всех предметных областей деятельности организаций. В составе этих средств существуют инструменты для описания моделей бизнес–процессов за счет различных диаграмм, схем, графов и таблиц [1].

Основными характеристиками CASE–средств, важными с точки зрения моделирования и оптимизации бизнес процессов, являются следующие[9]:

- *наличие графического интерфейса*. Для представления моделей процессов CASE–средства должны обладать возможностью отображать процессы в виде схем. Схемы много проще в использовании, чем различные текстовые и числовые описания. Это позволяет получать легко управляемые компоненты модели, обладающие простой и ясной структурой;
- *наличие репозитория*. Репозиторий– это общая база данных, которая содержит описание элементов процессов и отношений между ними. Каждый объект репозитория должен обладать перечнем свойств, характерных только для этого объекта;
- *гибкость применения*. Эта характеристика дает возможность представлять бизнес процессы в различных вариантах, важных с точки зрения анализа. CASE–средства должны позволять проводить анализ процессов и создавать модели, сфокусированные на различных аспектах деятельности предприятия;
- *возможность коллективной работы*. Анализ и моделирование процессов может требовать совместной работы нескольких человек. Для одновременной работы над моделями процессов CASE–средства должны обеспечивать управление изменениями любых фрагментов моделей и их модификацией при коллективном доступе;
- *построение прототипов*. Прототипы процессов необходимы для того, чтобы на ранних стадиях изменения процессов можно было понять, насколько процесс будет соответствовать требованиям;
- *построение отчетов*. CASE–средства должны обеспечивать построение отчетов по всем моделям процессов с учетом взаимосвязи элементов. Такие отчеты необходи-

мы для анализа моделей и определения возможностей оптимизации. За счет отчетов обеспечивается контроль полноты и достаточности моделей, уровень декомпозиции процессов, правильность синтаксиса диаграмм и типов применяемых элементов.

1.2. Классификация CASE–средств

Все современные CASE–средства могут быть классифицированы, в основном, по *типам* и *категориям*. Классификация по *типам* отражает функциональную ориентацию CASE–средств на те или иные процессы жизненного цикла (ЖЦ). Классификация по *категориям* определяет степень интегрированности по выполняемым функциям и включает отдельные локальные средства, решающие небольшие автономные задачи, набор частично интегрированных средств, охватывающих большинство этапов жизненного цикла ИС, и полностью интегрированные средства, поддерживающие весь ЖЦИС и связанные общим репозиторием [1]. Помимо этого, CASE–средства можно классифицировать по типу используемых моделей и по степени независимости от СУБД.

По *ориентации на этапы жизненного цикла* выделяют следующие основные типы [1]:

- *средства анализа*, предназначенные для построения и анализа моделей предметной области. Примерами программ этого типа являются: Design/IDEF, разработки фирмы MetaSoftware и BPwin (разработка LogicWorks);
- *средства анализа и проектирования*, обеспечивающие создание проектных спецификаций. Примеры: Vantage Team Builder (Cayenne), Designer/2000 (ORACLE), Silverrun (CSA), PRO–IV (McDonnell Douglas), CASE.Аналитик (МакроПроджект);

- *средства проектирования баз данных*, обеспечивающие моделирование данных и генерацию схем баз данных (как правило, на языке SQL) для наиболее распространенных СУБД. К ним относятся ERwin (Logic Works), S–Designor (SDP) и DateBase Designer (ORACLE);
- *средства разработки приложений*. К ним относятся средства JAM (JYACC), PowerBuilder (Sybase), Developer/2000 (ORACLE), New Era (Informix), SQL Windows (Gupta), Delphi (Borland) и др.

По *функциональной полноте* CASE–системы и средства можно условно разделить на следующие типы:

- системы, предназначенные для решения частных задач на одном или нескольких этапах жизненного цикла;
- интегрированные системы, поддерживающие весь жизненный цикл ИС и связанные с общим репозиторием.

По *поддерживаемым методологиям проектирования* CASE–системы можно разделить на три основные разновидности: *функционально (структурно)–ориентированные, объектно–ориентированные и комплексно–ориентированные (набор методологий проектирования)*.

По *степени независимости от СУБД* CASE–системы можно разделить на две группы:

- независимые системы. Такие CASE–системы поставляются в виде автономных систем, не входящих в состав конкретной СУБД;
- встроенные в СУБД. Такие CASE–системы поддерживают главным образом формат баз данных СУБД, в состав которой они входят. При этом возможна поддержка и других форматов баз данных.

1.3. Пример проектирования БД с помощью CASE–средства ERwin

Одним из CASE–средств, наиболее удачных с точки зрения соотношения цены, простоты использования и возможностей, является AllFusionERwinDateModeler (ранее ERwin)фирмы ComputerAssociates.

ERwin – это графический инструментарий для моделирования данных, основной целью которого является помощь аналитику в использовании правил и требований к информации при создании логических и физических моделей данных. Моделирование данных представляет собой деятельность по формированию и документированию требований к информации. Требования к информации описывают данные и правила, необходимые для поддержки профессиональной деятельности. Модель данных может выражать как сложные информационные потребности целой корпорации, так и конкретные информационные потребности одной единственной программы [6].

Преимущества от использования CASE–средства ERwin:

1. Существенное повышение скорости разработки за счет мощного редактора моделей, автоматической генерации базы данных, автоматической подготовки документации.

2. Возможность легко вносить изменения в модель при разработке и расширении системы.

3. Нет необходимости ручной подготовки SQL–предложений для создания базы данных.

4. Возможность создания моделей БД, позволяющих автоматически решать вопросы, связанные с сохранением ее целостности.

5. Независимость логической модели от используемой системы управления базой данных (СУБД), что позволяет применять универсальные методы для ее экспорта в конкретные СУБД.

6. Возможность автоматического формирования большого числа отчётов, отражающих текущее состояние процесса проектирования БД. Важно, что эти отчеты всегда в точности соответствуют реальной структуре БД.

7. Обратное проектирование позволяет документировать и вносить изменения в существующие информационные системы.

Существуют две точки зрения на информационную модель и, соответственно, два разных способа мышления и моделирования – *логический уровень* и *физический уровень*.

Логический уровень – это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире. Объекты модели, представляемые на логическом уровне, называются *сущностями* и *атрибутами*. Логическая модель данных может быть построена на основе другой логической модели, например, на основе модели процессов. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

Физическая модель данных, напротив, зависит от конкретной СУБД. В физической модели должна содержаться информация обо всех объектах БД. Если в логической модели не имеет значения, какой конкретно тип данных имеет атрибут, то в физической модели важно описать всю информацию о конкретных физических объектах – таблицах, колонках, индексах, процедурах и т. д. Одной и той же логической модели может соответствовать несколько разных физических моделей.

ERwin поддерживает построение и организацию работы с этими двумя различными уровнями представления одной модели и позволяет создавать множество вариантов отображения модели на каждом уровне.

Формально в ERwin поддерживается три уровня отображения моделей:

- Logical – логический уровень (соответствует логической модели);
- Physical – физический уровень (соответствует физической модели);
- Logical/Physical – стандартный уровень (включает логическую и физическую модели).

Существует большое количество литературы, ориентированной на функциональные возможности пакета AllFusion-ERwinDateModeler, такие как [2, 6, 8-11]. В данном пособии рассмотрим только основные функции ERwin, пользуясь примером, описанным ниже.

Создание модели данных, как правило, начинается с создания логической модели. После описания логической модели проектировщик выберет необходимую СУБД и ERwin автоматически создаст соответствующую физическую модель. На основе физической модели ERwin может сгенерировать системный каталог СУБД или соответствующий SQL–скрипт [8].

Обычно выбирают оба уровня представления модели данных: Logical/Physical (рис. 1.1). Если будет использоваться модель данных на физическом уровне, то необходимо указать СУБД (Database) и, если требуется, версию (Version).

В качестве примера рассмотрим создание логической модели базы данных для автоматизации работы Государственной аттестационной комиссии (ГАК). Ежегодно в университете проводится защита выпускных квалификационных работ (ВКР) студентов по двум направлениям: бакалавриат, магистратура. В состав аттестационной комиссии (5 человек) могут входить как сотрудники университета, так и члены сторонних организаций. В результате защиты ВКР комиссия ставит студенту оценку (отлично, хорошо, удовлетворительно, неудовлетворительно) и выносит решение о присвоении выпускнику квалификации, соответствующей его нап-

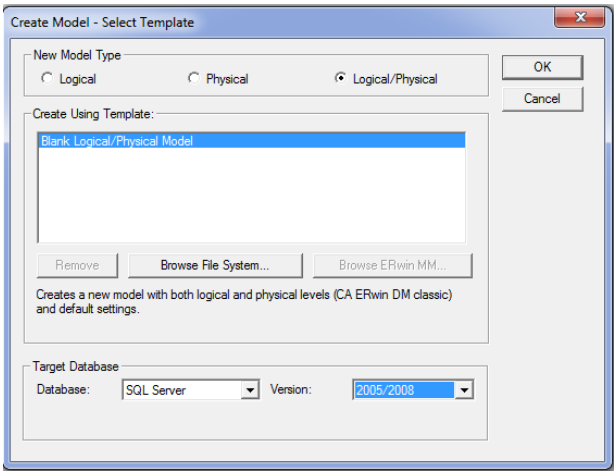

Рис. 1.1. Создание новой модели. Диалоговое окно CreateModel – SelectTemplate

равлению подготовки. База данных позволит автоматически создавать и выводить на печать необходимые для проведения ГАК отчеты.

1.3.1. Создание новой модели данных

Чтобы создать модель БД, необходимо выбрать пункт главного меню File/New … и указать, что необходимо создать логическую и физическую модель (Logical/Physical), а в качестве целевой СУБД (Target Datebase) выбрать например SQLServer (рис. 1.1).

Для создаваемой модели данных необходимо заполнить ее свойства в соответствующем диалоговом окне (Model Properties). В пункте меню Model/Model Properties … надо заполнить поля вкладок.

На вкладке General вводятся сведения о названии модели данных и ее авторе (рис. 1.2).

На вкладке Definition можно описать предназначение и выполняемые базой данных функции (рис. 1.3).

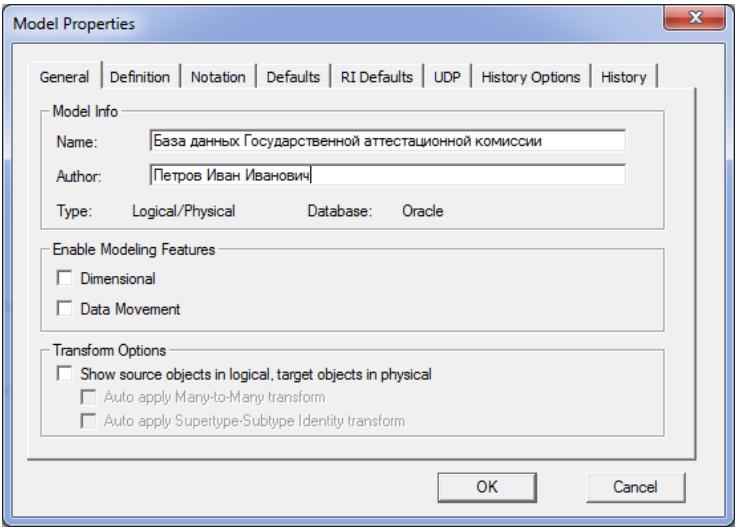

Рис. 1.2. Описание свойств создаваемой модели, вкладка General на форме ModelProperties

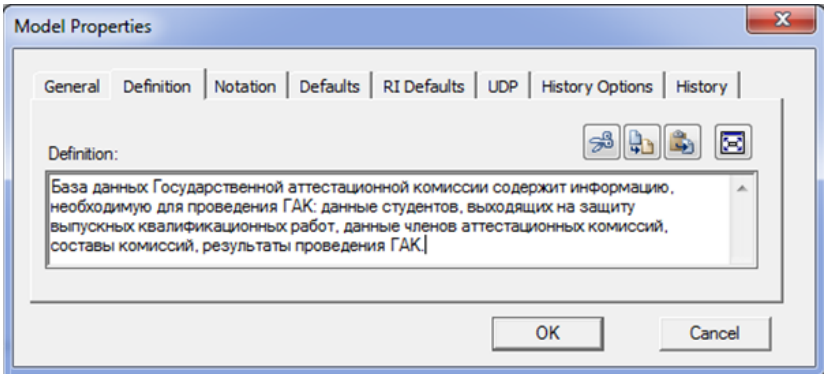

Рис. 1.3. Описание свойств создаваемой модели, вкладка Definition на форме ModelProperties

Вкладка Notation позволяет выбрать нотацию для логической и/или физической модели данных (рис. 1.4). Описывать модель данных можно в трех международных признанных системах обозначений (нотациях):

1. Integration DEFinition for Information Modeling (IDEF1X).
2. Information Engenering (IE).
3. Dimensional Modeling (DM).

IDEF1X была разработана для армии США и является федеральным стандартом США. Кроме того, она является стандартом в ряде международных организаций.

Нотация IE – Information Engineering (информационное проектирование) разработана Клайвом Финклештейном и Джеймсом Мартином. Нотация использовалась сначала преимущественно в промышленности, а в настоящее время широко применяется в различных областях. Нотация IE во многом похожа на IDEF1X.

Нотация DM – Dimensional Modeling (многомерное моделирование данных). Специализированная нотация, предназначенная для разработки хранилищ данных, применяемая только на уровне физического моделирования.

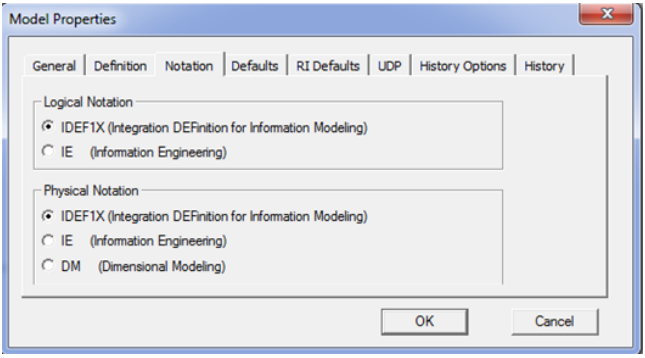

Рис. 1.4. Описание свойств создаваемой модели,
вкладка Notation на форме ModelProperties

1.3.2. Описание сущностей и их атрибутов

Основными компонентами диаграммы логического уровня модели в ERwin являются сущности, атрибуты и связи. Построение модели данных предполагает определение сущностей и атрибутов: необходимо определить, какая информация будет храниться в конкретной сущности и в конкретном атрибуте. Связь является логическим отношением между сущностями.

Проанализировав рассматриваемую предметную область, можно выделить следующие основные сущности:

— Сущность СТУДЕНТ должна содержать всю информацию о студенте.

В данной сущности можно выделить следующие атрибуты:

- Номер студенческого билета;
- ФИО студента;
- Тема ВКР;
- Средний балл;
- Номер группы;
- Научный руководитель;
- Оценка за ВКР.

— Сущность СОТРУДНИК должна содержать информацию о всех сотрудниках, которые могут принимать участие в проведении Государственной аттестационной комиссии. Это может быть научный руководитель или член комиссии ГАК, причем сотрудник может работать как в университете, так и в сторонней организации.

В данной сущности выделяются следующие атрибуты:

- ФИО сотрудника;
- Место работы;
- Должность;
- Ученая степень;
- Ученое звание.

— Сущность КОМИССИЯ необходима для того, чтобы была возможность ежегодно формировать составы комиссий. Для каждого направления (бакалавриат или магистратура) состав комиссий различен. Также известно, что комиссия может меняться каждый год. В состав ГАК входят пять человек, информация о них должна отражаться в данной сущности.

Исходя из вышеизложенных соображений, можно сделать вывод о том, что сущность КОМИССИЯ должна иметь атрибуты:

- Год защиты;
- Направление подготовки;
- 1 член комиссии;
- 2 член комиссии;
- 3 член комиссии;
- 4 член комиссии;
- 5 член комиссии.

— Сущность ДИСЦИПЛИНА необходима для того, чтобы у секретаря ГАК была полная информация об успеваемости студента, а также для подсчета среднего балла. В сущности будет хранится информация о дисциплинах, она будет содержать следующие атрибуты:

- Название дисциплины;
- Количество часов.

Если рассмотреть описанные выше сущности еще раз, то можно заметить атрибуты, которые можно вынести в отдельные сущности. Так, например, атрибут Ученая Степень имеет определенный список ученых степеней (д.т.н., к.т.н., к.ф.–м.н., и т.д.), то же самое можно сказать и про атрибут Ученое Звание (доцент, профессор и т.д.). Так же в отдельную сущность вынесем атрибут Место Работы. У сотрудников одного университета место работы заведомо совпадает, поэтому будет удобнее указывать место работы, выбирая его из сущности–справочника, вместо того, чтобы

вбивать каждый раз одну и ту же информацию. В сущности КОМИССИЯ есть атрибут Направление Подготовки, его мы также вынесем в отдельную сущность–справочник.

В результате получим еще несколько сущностей:

- Сущность УЧЕНАЯ СТЕПЕНЬ;
- Сущность УЧЕНОЕ ЗВАНИЕ;
- Сущность МЕСТО РАБОТЫ;
- Сущность НАПРАВЛЕНИЕ ПОДГОТОВКИ.

Далее необходимо создать сущности рассматриваемой предметной области. Для внесения сущности в модель нужно воспользоваться кнопкой  на панели инструментов ERwinToolbox. Затем необходимо щелкнуть по тому месту диаграммы, где планируется расположить новую сущность. Щелкнув правой кнопкой мыши по сущности и выбрав из контекстного меню пункт EntityProperties, можно вызвать диалог Entities, в котором определяются имя, описание и комментарии сущности. Закладка Definition используется для ввода определения сущности. Закладка Volumetrics позволяет на логическом уровне вводить информацию о приблизительном размере соответствующих таблиц. А с помощью закладок Note, Note 2, Note 3, UDP можно внести дополнительные комментарии и определения к сущности. Во вкладке Icon каждой сущности можно поставить в соответствие изображение, которое будет отображаться в режиме просмотра модели на уровне иконок.

На рис. 1.5 изображен пример заполнения вкладки Definition для сущности Студент. Затем следует указать атрибуты созданной сущности. Для этого в диалоговом окне Attributes (пункт меню Model/Attributes …) необходимо создать атрибуты и заполнить свойства каждого из них (рис. 1.6). Аналогичным образом следует создать описанные выше сущности и их атрибуты.

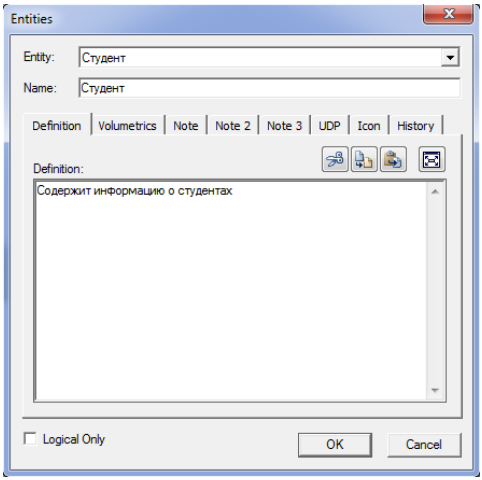

Рис. 1.5. Описание свойств сущности, вкладка Definition на форме Entities

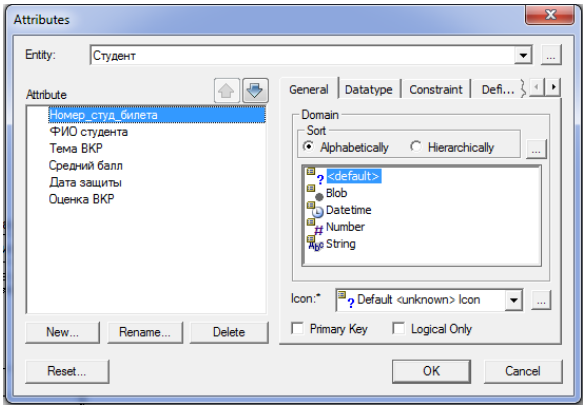

Рис. 1.6. Описание атрибутов сущности, форма Attributes

Затем для каждой сущности *необходимо указать первичный ключ* – набор атрибутов, значения которых однозначно определяют каждый экземпляр сущности. Для задания первичных ключей и атрибутов используется редактор атрибутов, на вкладке General

диалога Attributes необходимо сделать пометку в окне выбора PrimaryKey.

В качестве первичных ключей можно использовать несколько атрибутов или групп атрибутов. Атрибуты, которые могут быть выбраны первичными ключами, называются кандидатами в ключевые атрибуты (потенциальные атрибуты).

Первичный ключ должен быть подобран таким образом, чтобы по значениям атрибутов, в него включенных, можно было точно идентифицировать экземпляр сущности. Никакой из атрибутов первичного ключа не должен иметь нулевое значение. Значения атрибутов первичного ключа не должны меняться. Если значение изменилось, значит, это уже другой экземпляр сущности.

При выборе первичного ключа можно внести в сущность дополнительный атрибут и сделать его ключом. Так, для определения первичного ключа часто используют уникальные номера, которые могут автоматически генерироваться системой при добавлении экземпляра сущности в БД.

Полезным источником информации при выборе первичного ключа может стать перечень требований к хранимой информации, приведенный в задании. Рассмотрим по очереди каждую из сущностей.

Сущность СТУДЕНТ содержит разнообразную информацию о студенте и выполняемой им работе. Среди выделенных выше атрибутов данной сущности на роль первичного ключа хорошо подходит атрибут Номер Студенческого Билета. Данный атрибут является уникальным, содержит только цифры, и поэтому его удобно использовать в качестве первичного ключа.

Поскольку атрибут ФИО сотрудника сущности СОТРУДНИК нельзя считать уникальным, потребуется ввести дополнительный атрибут Код сотрудника, который в данной сущности будем использовать как первичный ключ.

Из описания предметной области известно, что состав комиссии может изменяться каждый год, а также состав комиссий по разным направлениям различен. Поэтому в сущности КОМИССИЯ целесообразно ввести дополнительный атрибут Код комиссии и использовать его в качестве первичного ключа.

В сущностях–справочниках, таких как ДИСЦИПЛИНА, УЧЕНОЕ ЗВАНИЕ, УЧЕНАЯ СТЕПЕНЬ, НАПРАВЛЕНИЕ ПОДГОТОВКИ необходимо ввести суррогатные ключи (произвольный номер, который уникальным образом определяет запись в сущности). Для перечисленных сущностей это будут атрибуты Код дисциплины, Код звания, Код степени, Код направления соответственно.

На диаграмме сущность изображается прямоугольником. Название сущности помещается над изображением прямоугольника. Первичные ключи попадают в область прямоугольника, расположенную над чертой. Остальные атрибуты помещаются под черту.

В результате проделанной работы на диаграмме появятся созданные сущности с атрибутами (рис. 1.7).

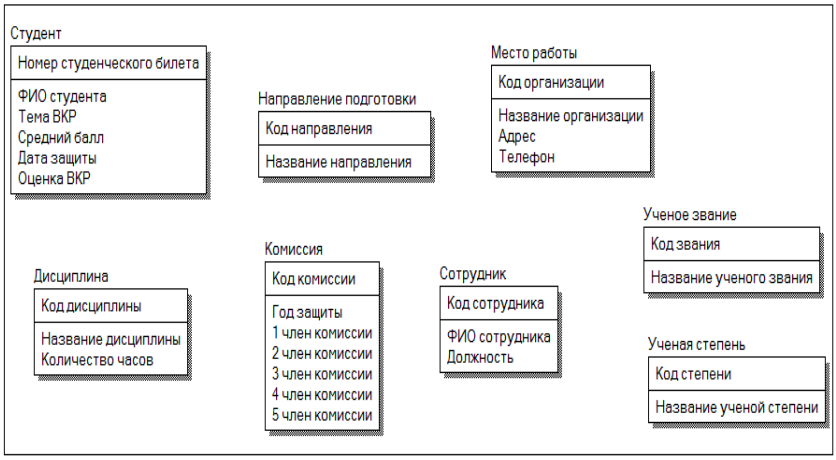

Рис. 1.7. Внесение сущностей и их атрибутов на диаграмму

1.3.3. Определение связей между сущностями

Следующим шагом в создании диаграммы является определение связей между сущностями. Следует напомнить, что для задания связи между двумя сущностями необходимо указать имя связи, тип связи, мощность связи, допустимость пустых (null) значений, требования по обеспечению ссылочной целостности, а в некоторых случаях и роль. ERwin поддерживает следующие типы связей: идентифицирующая/неидентифицирующая, полная/неполная категория, связи «один ко многим» и «многие ко многим».

Имя связи (VerbPhrase) – фраза, характеризующая отношение между родительской и дочерней сущностями. Для связи «один ко многим» достаточно указать имя, характеризующее отношение родительской сущности к дочерней (Parent–to–Child) (рис. 1.8). Для связи «многие ко многим» следует указывать имя как Parent–to–Child, так и Child–to–Parent (рис. 1.9). Это облегчает чтение диаграммы.

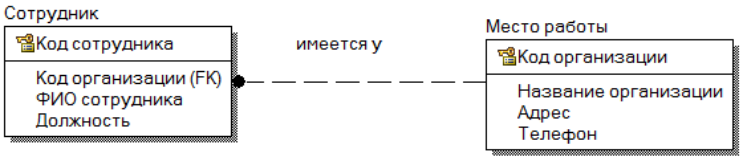

Рис. 1.8. Имя связи между сущностями для примера связи «один ко многим»

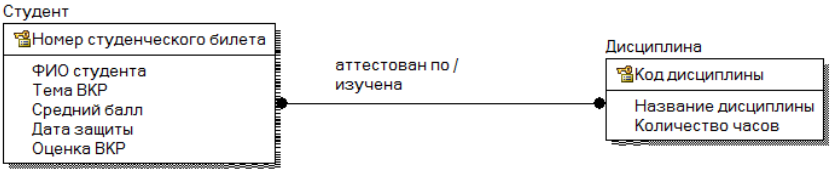

Рис. 1.9. Имя связи между сущностями для примера связи «многие ко многим»

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

Для рассматриваемой предметной области можно выделить следующие связи между сущностями:

- СОТРУДНИК *руководит* СТУДЕНТОМ.
- СОТРУДНИК *участвует* в КОМИССИИ.
- МЕСТО РАБОТЫ *имеется* у СОТРУДНИКА.
- УЧЕНОЕ ЗВАНИЕ *присвоено* СОТРУДНИКУ.
- УЧЕНАЯ СТЕПЕНЬ *присвоена* СОТРУДНИКУ.
- НАПРАВЛЕНИЕ ПОДГОТОВКИ *определяет* КОМИССИЮ.
- СТУДЕНТ *аттестован* по ДИСЦИПЛИНЕ.

В IDEF1X концепция зависимых и независимых сущностей усиливается типом взаимосвязей между двумя сущностями. Если нужно, чтобы внешний ключ передавался в дочернюю сущность (и, в результате, создавал зависимую сущность), то необходимо создать *идентифицирующую связь* между родительской и дочерней сущностями. Идентифицирующие взаимосвязи обозначаются сплошной линией между сущностями. Зависимая сущность изображается прямоугольником со скругленными углами. В дочерней сущности новые атрибуты помечаются как внешний ключ – (ForeignKey –FK) [10].

При установлении *неидентифицирующей связи* дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов дочерней сущности. Неидентифицирующая связь служит для связывания независимых сущностей [11].

Для создания новой связи можно воспользоваться палитрой инструментов ERwin  . На панели следует

выбрать требуемую связь, это может быть связь типа «один ко многим» идентифицирующая и неидентифицирующая , либо связь «многие ко многим» . Сначала нужно щелкнуть по родительской сущности, а потом – по дочерней.

Для редактирования свойств связи следует щелкнуть правой кнопкой мыши по линии связи и выбрать в контекстном меню пункт RelationshipProperties. В появившемся диалоге Relationships в закладке General можно задать имя связи (раздел VerbPhrase), мощность (раздел Cardinality), тип связи (раздел RelationshipType).

На рис. 1.10 изображено задание имени связи между сущностями СОТРУДНИК и КОМИССИЯ на вкладке General.

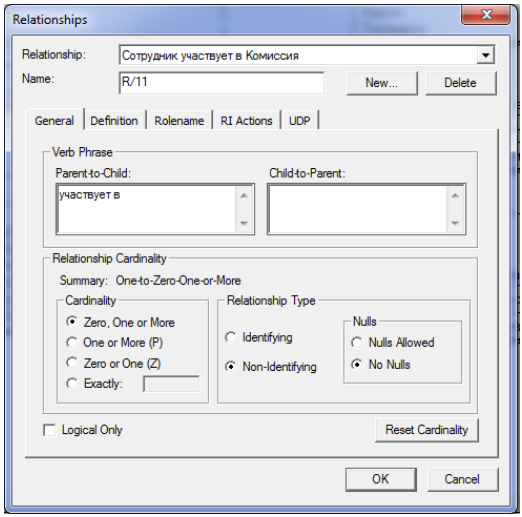

Рис. 1.10. Описание связи между сущностями, форма Relationships

В закладке Definition диалога Relationships можно дать более полное определение связи. В закладке RIActions определяют пра-

вила ссылочной целостности, закладка UDP служит для задания значений свойств, определяемых пользователем. В закладке Rolename можно задать *имя роли*.

Далее необходимо задать *имя роли (функциональное имя)*.

В нашем примере в сущности КОМИССИЯ внешний ключ Код сотрудника имеет функциональное имя 1 член комиссии, которое показывает, какую роль играет этот атрибут в сущности. Таким же образом назначаем имя роли для второго, третьего, четвертого и пятого членов комиссии. По умолчанию в списке атрибутов показывается только имя роли. Для отображения полного имени атрибута следует в контекстном меню, которое появляется, если щелкнуть правой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт EntitiyDisplay и затем включить опцию Rolename/Attribute. Полное имя показывается как функциональное имя и базовое имя, разделенные точкой (рис. 1.11) [11].

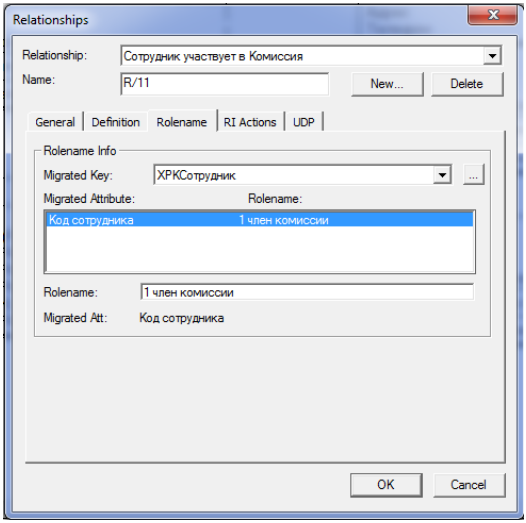

Рис. 1.11. Задание имени роли на вкладке Rolename, форма Relationships

Стоит отметить, что связь типа «многие ко многим» может быть создана только на логическом уровне. На физическом уровне такая связь должна быть преобразована. По умолчанию при переходе к физическому уровню ERwin автоматически не преобразует связь «многие ко многим», и на физическом уровне диаграмма выглядит так же, как и на логическом. Однако при генерации схемы такая связь игнорируется. Связь «многие ко многим» может быть преобразована к типу «один ко многим» либо вручную, либо автоматически.

Для принудительного преобразования связи «многие ко многим» на логическом уровне необходимо щелкнуть по связи правой кнопкой мыши и выбрать пункт меню Create Association Table. Далее необходимо выполнить все предлагаемые шаги для преобразования связи.

Автоматическое преобразование может быть выполнено при переходе на физический уровень. Преобразование связи включает создание новой таблицы и двух новых связей «один ко многим» от старых к новой таблице. Однако автоматического решения проблемы связи «многие ко многим» не всегда оказывается достаточно.

В рассматриваемом примере связь «многие ко многим» существует между сущностями СТУДЕНТ и ДИСЦИПЛИНА. Действительно, один студент в ходе учебного процесса изучает множество дисциплин и одна и та же дисциплина изучается многими студентами. Для преобразования связи введем дополнительную сущность СТУДЕНТ_ДИСЦИПЛИНА. Поскольку для секретаря ГАК существенным является не только изученные студентом дисциплины, но и оценки, полученные по предметам (они нужны для определения среднего балла студента), введем в новую сущность дополнительный атрибут «Оценка».

В результате определения всех связей между сущностями диаграмма примет вид, изображенный на рис. 1.12.

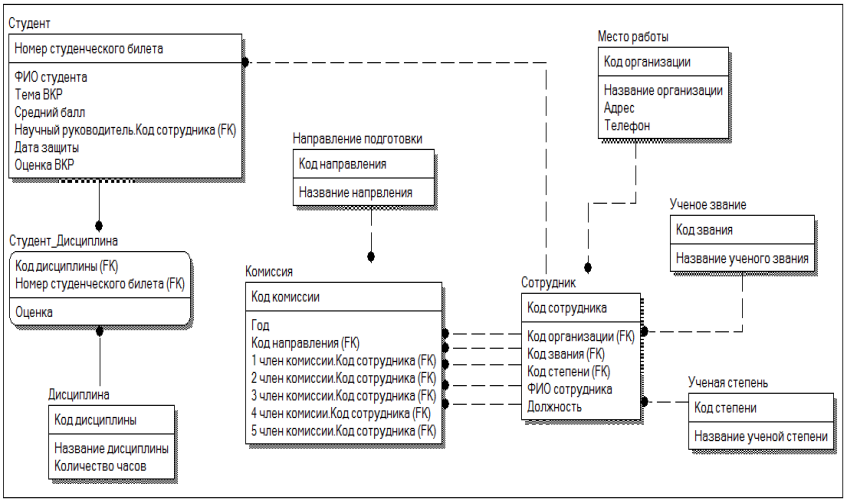

Рис. 1.12. Логическая модель предметной области

1.3.4. Создание физического уровня модели

Поскольку модель физического уровня ориентируется на конкретную СУБД, необходимо выбрать требуемую СУБД с помощью редактора TargetServer в меню Datebase/ChooseDatebase (доступно только на физическом уровне) (рис. 1.13). ERwin поддерживает практически все распространенные СУБД.

Диалог Target Server позволяет задать для новых колонок тип данных по умолчанию и опцию NULL. Тип данных можно выбрать в раскрывающемся списке Default Datetype, который автоматически заполняется типами данных, поддерживаемыми выбранной СУБД.

В результате перехода модели с логического уровня на физический сущности становятся таблицами, а атрибуты – столбцами.

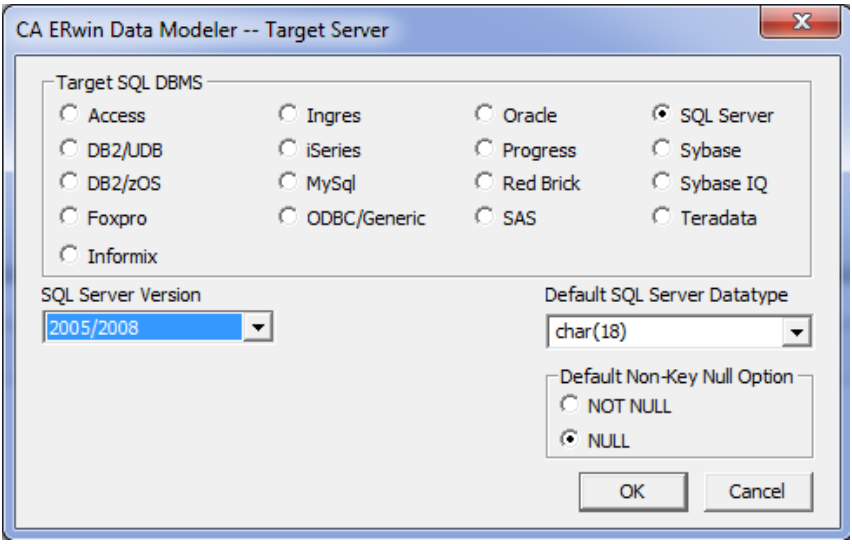

Рис. 1.13. Выбор СУБД на физическом уровне

При смене СУБД Erwin позволяет автоматически преобразовать тип данных полей таблиц. Добавить таблицы, поля или создать связи можно так же, как и на логическом уровне. Имена таблиц и полей в физической модели создаются на основе имен соответствующих сущностей и атрибутов, учитывая максимальную длину и другие синтаксические ограничения выбранной СУБД [6].

На уровне физической модели данных можно задать дополнительные параметры, зависящие от выбранной СУБД, например, параметры таблиц и табличных пространств, сегменты отката, триггеры и хранимые процедуры, ограничения целостности и значения по умолчанию, индексы и т.д. [6].

1.4. Контрольные вопросы к главе 1

1. Какова основная цель CASE–средств?
2. Перечислите основные характеристики CASE–средств.
3. Как можно классифицировать CASE–средства?
4. Перечислите основные преимущества от использования CASE–средства ERwin Data Modeler.
5. Сколько уровней отображения модели поддерживает ERwin Data Modeler?
6. Перечислите основные компоненты диаграммы логического уровня модели в ERwin Data Modeler.

Глава 2. ЯЗЫК SQL

Рост количества данных, необходимость их хранения и обработки привела к тому, что возникла потребность в создании стандартного языка БД.

SQL (Structured Query Language) – структурированный язык запросов, предоставляющий средства создания и обработки данных в реляционных БД. Язык запросов к базам данных SQL появился в 70–е гг. Его прототип был разработан фирмой IBM и известен под названием SEQUEL (Structured English QUEry Language). SQL вобрал в себя достоинства реляционной модели (в частности то, что в ее основе лежит математический аппарат реляционной алгебры и реляционного исчисления), используя при этом сравнительно небольшое число операторов и простой синтаксис.

В SQL существуют:

- предложения определения данных – создание БД, а также определение и уничтожение таблиц и индексов;
- запросы на выбор данных – предложение SELECT;
- предложения модификации данных – добавление, удаление и изменение данных;
- предложения управления данными – предоставление и отмена привилегий на доступ к данным, управление транзакциями и другие.

Язык SQL считают *декларативным* (описательным) языком, в отличие от языков, на которых пишутся программы. Это означа-

ет, что выражения на языке SQL описывают что требуется сделать, а не каким образом.

SQL является непроцедурным языком и не содержит операторов управления, организации подпрограмм, ввода–вывода и т. п. В связи с этим SQL автономно не используется, обычно он погружен в среду встроенного языка программирования СУБД. В различных СУБД состав операторов SQL может несколько отличаться. В этом случае его называют встроенным SQL.

Основным назначением языка SQL (как и других языков для работы с базами данных) является подготовка и выполнение запросов. В результате выборки данных из одной или нескольких таблиц может быть получено множество записей, называемое представлением.

Для удобства работы с представлениями в язык SQL введено понятие курсора. Курсор представляет собой своеобразный указатель, используемый для перемещения по наборам записей при их обработке.

Операторы языка SQL условно можно разделить на подъязыки:

- DDL (Date Definition Language) – язык определения данных;
- DML (Date Manipulation Language) – язык манипулирования данными.

Языку SQL посвящено большое число книг и учебных пособий, некоторые из них приведены в списке литературы данного пособия [3-5, 7, 12]. В связи с этим рассмотрим лишь важные общие особенности языка.

В табл. 2.1 приведены команды, относящиеся к языкам определения и манипулирования данными.

Таблица 2.1. **Операторы языка SQL**

Вид	Название	Назначение
DDL	CREATE TABLE	создание таблицы
	DROP TABLE	удаление таблицы
	ALTER TABLE	изменение структуры таблицы
	CREATE INDEX	создание индекса
	DROP INDEX	удаление индекса
	CREATE VIEW	создание представления
	DROP VIEW	удаление представления
	GRAND	назначение привилегий
	REVOKE	удаление привилегий
DML	SELECT	выборка записей
	UPDATE	изменение записей
	INSERT	вставка новых записей
	DELETE	удаление записей

2.1. Типы данных SQL

В реляционной модели данных каждый атрибут определен на некотором домене. В языке SQL для определения домена служит понятие *типа данных*.

Стандартные типы данных делятся на следующие категории:

- *Текст*. Это могут быть отдельные символы, строки фиксированной длины и строки переменной длины. Любой текст заключается в одинарные кавычки (апострофы).
- *Точные числа* (целые и десятичные).
- *Приближенные числа*. Числа с плавающей точкой и с основанием 10, используемые в научной нотации.
- *Дата и время*. Это может быть только дата, только время или их комбинация.
- *Двоичные значения*.

К наиболее используемым типам данных относят:

- *CHAR(n)* – текстовая строка фиксированной длины, содержащая n символов. Если параметр n опущен, он счита-

ется равным 1. При использовании строк фиксированной длины для заполнения незанятых позиций обычно используются пробелы. Не рекомендуется использовать тип строк фиксированной длины для полей, в которых предполагается хранить данные различной длины, например имена. Из-за нерационального использования имеющегося пространства могут возникнуть проблемы с организацией сравнения содержащихся в соответствующих полях данных.

- *VARCHAR(n)* – текстовая строка переменной длины, содержащая не более n символов. Если параметр n опущен, значение по умолчанию определяется реализацией. Для данных в виде строк различной длины рекомендуется всегда использовать тип данных, допускающий строки переменной длины.
- *INTEGER* – точный числовой тип без десятичной точки. Записывается так же как *INT*.
- *DATE* – тип, содержащий дату с полями для года, месяца и дня.

Точное название и описание типов данных зависит от конкретной СУБД.

2.2. Создание, удаление и изменение таблиц

Таблицы создаются командой CREATE TABLE. Эта команда создает пустую таблицу, не содержащую записи. Главное в этой команде – это определение имени таблицы и описания набора имен полей, которые указываются в соответствующем порядке. Кроме того, этой командой оговариваются типы данных и размеры полей таблицы.

Упрощенный синтаксис этой команды:

CREATE TABLE <имя_таблицы>

({<имя_столбца><тип_данных>[<размер>]

[<ограничение_целостности_столбца>],

…

[<ограничение_целостности_таблицы>]);

В квадратных скобках указаны элементы, которые могут в запросе отсутствовать.

Пример 1.1. Создание, изменение и удаление таблиц

1. Создать таблицу САМОЛЕТ, содержащую поля код самолета и модель:

```
CREATE TABLE Airplane
   (IdAirplane int,
   Model varchar(20));
```

Для расширения возможностей работы с данными, вводимыми в таблицу, необходим механизм проверки значений (*ограничения данных*).

В примере, описанном выше, в качестве ограничений использовались только тип и размер вводимых данных. В результате SQL будет отбрасывать любые значения, не соответствующие заданному критерию.

Команду CREATE TABLE часто используют для того, чтобы предохранить поле от попадания в него NULL значений (специальное обозначение того, что значение поля неизвестно). Для этого применяется ограничение NOT NULL, которое применимо только к отдельным столбцам.

Для определения группы из одного или нескольких столбцов в качестве первичного ключа используется ограничение PRIMARY KEY. Оно может применяться как на уровне отдельных столбцов,

так и на уровне таблицы. Для первичных ключей недопустимы NULL значения.

Если необходимо быть уверенным в том, что все значения, введенные в поле, отличаются друг от друга, то в ограничение поля при создании таблицы помещают ключевое слово UNIQUE. Данное ограничение следует применять там, где значения должны различаться по логическим соображениям, не связанным со структурой базы данных. Для ограничения UNIQUE допустимы NULL значения.

Ограничение CHECK можно использовать для задания набора допустимых для столбца значений. Это ограничение состоит из ключевого слова CHECK и заключенного в скобки предиката, проверяющего столбец. Попытка занести в такой столбец значение, которое делает предикат ложным, будет отклонена.

DEFAULT – выражение, используемое в столбце по умолчанию. С помощью этого средства пользователь может указать требуемые для работы ограничения или условия заполнения таблиц.

2. Создать таблицу СОТРУДНИК с полями: код сотрудника (первичный ключ), ФИО, должность, стаж, дата рождения, зарплата, пол:

```
CREATE TABLE Worker
    (IdWorker int NOT NULL PRIMARY KEY,
    Name varchar(30) NOT NULL,
    Position varchar(30) NOT NULL,
    Seniority int  NULL
    BirthDate smalldatetime NULL,
    Salary int  NOT NULL
    Gender char(1) CHECK (Gender IN ('м', 'ж'));
```

3. Создать таблицу СЛУЖАЩИЙ с полями: фамилия, имя, отчество, номер страхового полиса, оклад, телефон, номер отдела. В данной таблице должен быть составной первичный ключ, вклю-

чающий в себя поля: фамилия, имя, отчество. Значение поля «Оклад» должно удовлетворять условию – оклад должен быть более 30 тыс. руб. Значение по умолчанию для поля «Отдел» – 1.

```
CREATE TABLE Employee
    (Surname varchar(30) NOT NULL,
    Name varchar(20) NOT NULL,
    Patronymic varchar(30) NOT NULL,
    Genderis char(15)  NOT NULL UNIQUE,
    Salary money  NOT NULL CHECK(Oklad>30000)
    Phone char(11) NULL,
    Dept smallint NOT NULL DEFAULT 1
    PRIMARY KEY (Surname, Name, Patronymic));
```

После того как таблица была создана, ее можно изменить. Команда ALTER TABLE является широко доступным средством для того, чтобы изменить определение существующей таблицы. Чаще всего с ее помощью добавляют поля к таблице, хотя она может удалять или изменять их размеры. Типичный синтаксис этой команды:

```
ALTER TABLE <имя_таблицы>
  ADD <имя_столбца> <тип_данных>;
```

Добавленное поле станет последним по порядку в таблице. Допускается добавление сразу нескольких новых полей.

Синтаксис команды DROP TABLE для удаления таблицы:

```
DROP TABLE <имя_таблицы>;
```

4. Добавить в таблицу Airplane поля: вместимость (Capacity) и дальность (Distance).

```
ALTER TABLE Airplane
  ADD Capacity smallint,
    Distance int;
```

5. Удалить таблицу Airplane:

```
DROP TABLE Airplane;
```

2.3. Команды модификации данных

Оператор INSERT для добавления записей в таблицу имеет формат:

INSERT INTO <имя_таблицы> [<имя_столбца>]

VALUES(<значение1>,<значение2>,...);

Список столбцов указывает на те столбцы, которым бу-дут присвоены значения в добавляемых записях. Список может быть опущен, тогда подразумеваются все столбцы таблицы (кроме объявленных как счетчик), причем в порядке, установленном при создании таблицы. Если в операторе INSERT указывается конкретный список имен полей, то любые пропущенные в нем столбцы должны быть объявлены при создании таблицы как допускающие значение NULL, за исключением тех случаев, когда при описании столбца использовался параметр DEFAULT.

Пример 2.2. Команды модификации данных

1. Добавим в таблицу Airplane новую запись:

INSERT INTO Airplane

VALUES(101, 'Ту–204–100', 210, 4500);

Можно использовать команду INSERT для того, чтобы получать или выбирать значения из одной таблицы и помещать их в другую. Для этого предложение VALUES заменяется на соответствующий запрос. Так же можно использовать подзапросы внутри любого запроса в команде INSERT.

2. Допустим существует таблица, содержащая информацию о самолетах фирмы «BOEING» (табл. 2.2):

Требуется поместить информацию о самолетах фирмы «BOEING» вместимостью более 200 человек из таблицы Boeing в таблицу Airplane:

Таблица 2.2. **Таблица Boeing**

IdAirplane	Model	Capacity	Distance
106	Boeing 717	90	2000
107	Boeing 787–8	250	13500
108	Boeing 787–10	440	12000

```
INSERT INTO Airplane
 SELECT *
 FROM Boeing
 WHERE Capacity > 200;
```

В результате данной вставки в таблице Airplane появятся три записи: одна, которую мы поместили туда ранее (пример 2.2(1)), и две – из таблицы Boeing, удовлетворяющие условию выборки:

IdAirplane	Model	Capacity	Distance
101	Ty–204–100	210	4500
107	Boeing 787–8	250	13500
108	Boeing 787–10	440	12000

Более детально синтаксис команды SELECT будет рассмотрен ниже.

Оператор DELETE предназначен для удаления группы записей из таблицы.

Формат оператора:

DELETE FROM <имя_таблицы> [WHERE <критерий>];

Если предложение WHERE присутствует, удаляются записи из таблицы, удовлетворяющие условию отбора. Если опустить предложение WHERE, из таблицы будут удалены все записи, однако сама таблица сохранится.

3. Удалить из таблицы Airplane самолеты с вместимостью более 300 человек:

DELETE FROM Airplane

WHERE Capacity > 300;

В результате таблица Airplane буде выглядеть следующим образом:

IdAirplane	Model	Capacity	Distance
101	Ту–204–100	210	4500
107	Boeing 787–8	250	13500

Оператор UPDATE применяется для изменения значений в группе записей или в одной записи указанной таблицы.

Формат оператора:

UPDATE <имя_таблицы>

SET <имя_столбца>=<выражение>[,...n][WHERE <условие_отбора>];

В предложении SET указываются имена одного или более столбцов, данные в которых необходимо изменить. Предложение WHERE является необязательным. Если оно опущено, значения указанных столбцов будут изменены во всех строках таблицы. Если предложение WHERE присутствует, то обновлены будут только строки, удовлетворяющие условию отбора. Выражение – это новое значение соответствующего столбца, оно должно быть совместимо со столбцом по типу данных.

4. В таблице Airplane изменить вместимость и дальность полета для самолета с кодом 101:

UPDATE Airplane SET Capacity =280, Distance =5000

WHERE IdAirplane =101;

Таблица Airplane после внесенного изменения:

IdAirplane	**Model**	**Capacity**	**Distance**
101	Ту–204–100	280	5000
107	Boeing 787–8	250	13500

2.4. Выборка данных. Оператор SELECT

Оператор SELECT языка SQL является самым часто используемым оператором и предназначен для выборки информации из таблиц БД. Результатом выполнения команды SELECT является временная таблица, которая помещается в курсор (специальную область памяти СУБД) и обычно выводится на экран.

2.4.1. Синтаксис команды SELECT

Упрощенный формат оператора:

SELECT [ALL|DISTINCT] <список атрибутов> FROM <список таблиц>

[WHERE <условие >]

[ORDER BY <список атрибутов>]

[GROUP BY <список атрибутов>]

[HAVING <условие>]

[UNION <выражение с оператором SELECT>];

В общем случае запрос начинается с ключевого слова SELECT, сопровождаемого пробелом. После этого должен следовать список разделенных запятыми имен полей, которые необходимо вывести. Имени поля или выражению может быть присвоен временный синоним (*алиас*), который будет названием поля в результирующей таблице:

<название атрибута> AS <алиас>

Ключевое слово FROM, следующее далее, сопровождается пробелом и именем таблицы, запрос к которой делается.

Если необходимо получить все поля таблицы, к которым обращается данный запрос, можно указать символ *. Поля будут выводиться в том порядке, в котором они были указаны при создании таблицы.

В дальнейшем будем использовать в качестве примера небольшую базу данных авиакомпании (табл. 2.3–2.6), которая содержит информацию о сотрудниках, имеющихся самолетах, составе экипажей и выполняемых рейсах.

В таблице СОТРУДНИК (Worker) содержится информация о сотрудниках авиакомпании. Данная таблица имеет поля:

- IdWorker – уникальный код сотрудника;
- Name – Фамилия, инициалы сотрудника;
- Position – название занимаемой должности;
- Seniority – стаж сотрудника;
- BirthDate – дата рождения;
- Salary – зарплата сотрудника;
- Gender – пол.

Таблица 2.3. **Таблица Worker**

Id Worker	Name	Position	Seniority	BirthDate	Salary	Gender
1	Петров И.А.	пилот	25	25.09.1968	45600	м
2	Иванов М.Г.	пилот	20	13.01.1958	45600	м
3	Васильева Б.Ю.	бортпроводник	5	12.12.1994	26000	ж
4	Котова И.Т.	бортпроводник	12	03.09.1978	26000	ж
5	Аносов Т.Ю.	бортинженер	6	12.07.1985	34300	м
6	Киселев А.А.	пилот	30	03.03.1956	45600	м
7	Комаров Т.Н.	пилот	23	15.09.1971	45600	м
8	Фаизов М.Н.	бортинженер	7	12.11.1981	34300	м
9	Рубцов Л.И.	бортинженер	15	12.08.1978	34300	м
10	Кольцова О.М.	бортпроводник	20	03.05.1974	26000	ж
11	Кузьмин Е.С.	бортпроводник	17	07.04.1973	26000	м
12	Яновский И.Р.	бортпроводник	12	14.09.1976	26000	м
13	Гостева С.А.	бортпроводник	3	17.03.1995	26000	ж
14	Понкратов А.А.	пилот	17	02.03.1961	45600	м
15	Фадеев П.В.	пилот	13	07.08.1967	45600	м
16	Наумов К.Ю.	пилот	32	19.02.1956	45600	м

Таблица САМОЛЕТ (Airplane) содержит информацию о имеющихся в авиакомпании самолетах, таблица имеет поля:

- IdAirplane – уникальный код самолета;
- Model – модель самолета;
- Capacity – вместимость (количество посадочных мест);
- Distance – максимальная дальность полета.

Таблица РЕЙС (Route, маршрут) содержит информацию о рейсах, выполняемых данной авиакомпанией. Назначение полей таблицы следующее:

- IdRoute – уникальный код рейса;
- IdAirplane – код самолета;
- IdAirCrew – код экипажа;
- PointOfDept – город вылета;
- PointOfDest – город назначения;
- Date – дата вылета.

Таблица 2.4. **Таблица Airplane**

IdAirplane	**Model**	**Capacity**	**Distance**
101	Ту–204–100	210	4500
102	Ту–204–300	164	9200
103	SSJ–95	95	3050
104	A300	254	5300
105	A350–900	420	15000
106	Boeing 717	90	2000
107	Boeing 787–8	250	13500
108	Boeing 787–10	440	12000
109	ATR 42	50	1500
110	Ил114	64	1500

Таблица 2.5. **Таблица Route**

IdRoute	IdAirplane	IdAirCrew	Point OfDept	Point OfDest	Date
98	105	901	Самара	Иркутск	2018–01–01
112	107	909	Самара	Берлин	2018–01–01
113	105	902	Казань	Милан	2018–01–02
IdRoute	**IdAirplane**	**IdAirCrew**	**Point OfDept**	**Point OfDest**	**Date**
116	104	905	Волгоград	Челябинск	2018–01–02
117	104	909	Москва	Казань	2018–01–02
120	108	902	Самара	Милан	2018–01–04
121	108	905	Казань	Москва	2018–01–04
201	101	910	Самара	Москва	2018–01–05
321	107	902	Казань	Челябинск	2018–01–05
421	109	909	Самара	Саратов	2018–01–05
455	103	904	Уфа	Самара	2018–01–08

Таблица ЭКИПАЖ (AirCrew) содержит информацию о составе экипажей. Причем в каждой строке таблицы содержатся номер экипажа и соответствующие этому номеру коды сотрудников, назначенных на должности пилотов, бортинженеров и бортпроводников. Состав экипажа может быть неполным, в этом случае соответствующие поля будут пустыми. Назначение полей таблицы:

- IdAirCrew – уникальный код экипажа;
- IdPilot1 – код сотрудника первого пилота;
- IdPilot2 – код сотрудника второго пилота;
- IdFlightEng1, IdFlightEng2 – коды сотрудников, назначенных на должности бортинженеров;
- IdSteward1, IdSteward2, IdSteward3– коды сотрудников, назначенных на должности бортпроводников.

Таблица 2.6. **Таблица AirCrew**

Id Air Crew	Id Pilot1	Id Pilot 2	Id Flight Eng1	Id Flight Eng2	Id Steward 1	Id Steward 2	Id Steward3
901	1	2	5	8	10	11	13
902	6	7	5	NULL	12	13	3
904	7	15	9	NULL	4	NULL	NULL
905	1	14	8	4	10	11	13
909	6	2	5	NULL	11	13	4
910	1	15	8	5	10	11	12

Пример 2.3. Команда SELECT

1. Вывести все строки из таблицы Самолет (Airplane):

SELECT * FROM Airplane;

IdAirplane	Model	Capacity	Distance
101	Ту–204–100	210	4500
102	Ту–204–300	164	9200
103	SSJ–95	95	3050
104	A300	254	5300
105	A350–900	420	15000
106	Boeing 717	90	2000
107	Boeing 787–8	250	13500
108	Boeing 787–10	440	12000
109	ATR 42	50	1500
110	Ил114	64	1500

При работе с данными очень часто возникает потребность в удалении избыточных данных. Это реализуется с помощью оператора DISTINCT**,** который обеспечивает возможность устранять повторяющиеся значения из предложения SELECT. Если вместо

DISTINCT указать ALL, то это будет иметь противоположный эффект и дублирование строк вывода сохранится.

2. Допустим, необходимо вывести список всех должностей с указанием оклада:

```
SELECT DISTINCT Position AS Должность, Salary AS Зарплата
  FROM Worker;
```

Должность	Зарплата
бортинженер	34300
бортпроводник	26000
пилот	45600

Если же в данном запросе не указать ключевое слово DISTINCT, то результирующая таблица будет содержать столько строк, сколько находится в таблице СОТРУДНИК, при этом должности и соответствующие им зарплаты будут иметь повторы.

Для упорядочения вывода полей таблицы используется команда ORDER BY, позволяющая сортировать вывод запроса согласно значениям в том или ином количестве выбранных столбцов. Если указывается несколько полей, то столбцы вывода упорядочиваются один внутри другого, при этом можно определять возрастание (ASC) или убывание (DESC) для каждого столбца. По умолчанию установлено возрастание.

ORDER BY можно использовать с GROUP BY для упорядочения групп, при этом ORDER BY должен быть последним.

3. Вывести список моделей самолетов и их вместимость в порядке убывания вместимости:

```
SELECT Model, Capacity
  FROM Airplane
  ORDER BY Capacity DESC;
```

Model	Capacity
Boeing 787–10	440
A350–900	420
A300	254
Boeing 787–8	250
Ty–204–100	210
Ty–204–300	164
SSJ–95	95
Boeing 717	90
Ил114	64
ATR 42	50

Ключевое слово WHERE позволяет использовать условие (предикат), принимающее значение истина или ложь для значений полей строк таблиц, к которым обращается оператор SELECT. Предложение WHERE определяет, какие строки указанных таблиц должны быть выбраны. В таблицу, являющуюся результатом запроса, включаются только те строки, для которых условие (предикат), указанное в предложении WHERE, принимает значение истина.

С помощью оператора GROUP BY можно объединить в одну группу все записи с одинаковым значением указанного поля (комбинации полей). Каждой такой группе в результирующей таблице соответствует одна запись. Допускается использование GROUP BY одновременно с несколькими полями.

Предложение HAVING позволяет указать условия выбора для групп записей.

Примеры использования WHERE, GROUP BY, HAVING будут рассмотрены ниже.

2.4.2. Операторы и предикаты

Существует несколько способов заставить таблицу предоставить ту информацию, которая необходима пользователю, а не просто выводить ее содержимое.

В условиях предложения WHERE могут использоваться операции сравнения, задаваемые следующими операторами: = (равно), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), <> (не равно).

Эти операторы имеют стандартные значения для числовых данных, а для символьных их определение зависит от кодов ASCII символов – они следуют в алфавитном порядке, причем заглавные буквы имеют меньший код, чем строчные, поэтому, например, 'Z' < 'a'.

Стандартными булевыми операторами, которые используются в SQL, являются AND, OR и NOT. Напомним, как они работают:

- AND использует два операнда в форме A AND B и оценивает их по отношению к истине, верны ли они оба;
- OR использует два операнда в форме A OR B и оценивает истинность: верен ли один из них;
- NOT использует один операнд в форме NOT A и заменяет его значение с ИСТИНА на ЛОЖЬ или наоборот.

Пример 2.4. Использование команды SELECT с предикатами

1. Вывести фамилии пилотов, у которых стаж больше 17 лет (указать стаж) и упорядочить по фамилии:

```
SELECT Name, Seniority
  FROM Worker
  WHERE Position='пилот' AND Seniority > 17
  ORDER BY Name;
```

Name	Seniority
Иванов М.Г.	20
Киселев А.А.	30
Комаров Т.Н.	23
Наумов К.Ю.	32
Петров И.А.	25

Для отбора необходимых данных можно использовать *предикаты*:

— IN - *предикат вхождения в список*

<выражение> IN (<список значений>)

– определяет множество значений, с которыми будет сравниваться значение <выражение>. Предикат считается истинным, если значение поля <выражение> равно хотя бы одному из элементов множества.

— BETWEEN- *предикат нахождения в диапазоне*

<выражение> BETWEEN <значение1> AND <значение2>

– определяет, входит ли значение поля <выражение> в указанные границы. Если значение поля меньше, чем значение1, или больше, чем значение2, предикат возвращает «ложь».

— LIKE – *предикат подобия*

<выражение> LIKE 'образец'

– используется для поиска подстрок, применяется только с полям типа CHAR, VARCHAR. Возможно использование шаблонов: '_' – один любой символ и '%' – произвольное количество символов (в т.ч., ни одного);

— IS NULL - *предикат неопределенного значения*.

<выражение> IS NULL

– определяет, установлено ли значение поля. Все другие предикаты и операторы сравнения возвращают неопреде-

лённый результат (null), если хотя бы один из операндов имеет значение null.

Все эти операции могут комбинироваться с операцией «не»: Not IN, Not Like, NOT BETWEEN, IS NOT NULL.

2. Вывести все имеющиеся самолеты марки 'Ту':

```
SELECT Model, Capacity, Distance
  FROM Airplane
  WHERE Model LIKE 'Ту%';
```

Model	Capacity	Distance
Ту–204–100	210	4500
Ту–204–300	164	9200

3. Вывести информацию о рейсах из Самары, на которые назначены самолеты с номерами 105 и 107:

```
SELECT IdRoute, IdAirplane, PointOfDept, PointOfDest, Date
  FROM Route
  WHERE IdAirplane IN (105, 107) AND PointOfDept='Самара';
```

IdRoute	IdAirplane	PointOfDept	PointOfDest	Date
98	105	Самара	Иркутск	01.01.2018
112	107	Самара	Берлин	01.01.2018

4. Вывести информацию о сотрудниках, у которых год рождения находится в диапазоне с 1971 по 1981:

```
SELECT IdWorker, Name, Position, Seniority, BirthDate
  FROM Worker
  WHERE BirthDate BETWEEN '01–01–1971' AND '31–12–1981'
  ORDER BY BirthDate;
```

IdWorker	Name	Position	Seniority	BirthDate
7	Комаров Т.Н.	пилот	23	15.09.1971
11	Кузьмина Е.С.	бортпроводник	17	07.04.1973
10	Кольцова О.М.	бортпроводник	20	03.05.1974
12	Яновский И.Р.	бортпроводник	12	14.09.1976
9	Рубцов Л.И.	бортинженер	15	12.08.1978
4	Котова И.Т.	бортпроводник	12	03.09.1978
8	Фаизов М.Н.	бортинженер	7	12.11.1981

5. Вывести номера экипажей, в которых не назначен второй бортинженер:

```
SELECT IdAirCrew
  FROM AirCrew
  WHERE IdFlightEng2 IS NULL;
```

IdAirCrew
902
904
909

2.4.3. Функции агрегирования

Запросы могут производить обобщенную групповую обработку значений полей, что реализуется с помощью агрегатных функций. Агрегатные функции производят одиночное значение для всей группы таблицы. В SQL допускаются следующие агрегатные функции.

— COUNT – определяет в результате количество строк (записей) или значений поля, не являющихся *NULL*-значениями.

COUNT (*) – подсчёт количества строк в результате.

COUNT (distinct <поле>) – подсчёт количества разных значений поля;

COUNT (<поле>) – подсчёт количества ненулевых значений поля;

— SUM – определяет арифметическую сумму значений указанного числового поля в результирующем множестве записей.

SUM (distinct <поле>) – суммирование различных значений поля;

— AVG – определяет среднее арифметическое значений указанного числового поля в результирующем множестве записей;

AVG (distinct <поле>) – среднее арифметическое разных значений поля;

— MAX, MIN – определяет максимальное (минимальное) значение указанного поля в результирующем множестве.

Агрегирующие функции можно комбинировать с фразой GROUP BY. В этом случае подсчет будет производиться для каждой группы записей с одинаковым значением комбинации полей, указанных в GROUP BY. При этом в списке выбора могут быть указаны только функции агрегирования, константы и поля, перечисленные в GROUP BY.

Пример 2.5. Использование функций агрегирования

1. Подсчитать количество сотрудников по должностям:

```
SELECT Position AS Должность, COUNT(*) AS Количество сотрудников
  FROM Worker
  GROUP BY Position;
```

Должность	Количество сотрудников
бортинженер	3
бортпроводник	6
пилот	7

2. Определить минимальный и максимальный стаж среди пилотов:

```
SELECT MIN(Seniority)AS MinSeniority,
    MAX(Seniority) AS MaxSeniority
 FROM Worker
 WHERE Position='пилот';
```

MinSeniority	MaxSeniority
13	32

3. Подсчитать сумму зарплаты по должностям:

```
SELECT Position AS Должность, SUM(Salary) AS Суммарная зарплата
 FROM Worker
 GROUP BY Position;
```

Должность	Суммарная зарплата
бортинженер	102900
бортпроводник	156000
пилот	319200

4. Вывести минимальный стаж сотрудников среди бортпроводников и бортинженеров:

```
SELECT Position, MIN(Seniority)
 FROM Worker
 GROUP BY Position
 HAVING Position IN ('бортинженер', 'бортпроводник');
```

Position	Seniority
бортинженер	6
бортпроводник	3

5.

6. Вывести средний возраст сотрудников по должностям:

```
SELECT Position, AVG(DATEDIFF(year,BirthDate,GetDate()))
AS Age
```

```
FROM Worker
GROUP BY Position;
```

Position	Age
бортинженер	36
бортпроводник	36
пилот	55

При написании данного запроса понадобились две функции даты, используемые в MS SQL Server:

- DATEDIFF(item, Dat1, Dat2) – вычисляет разность между частями двух дат Dat1 и Dat2 и возвращает результат в виде целого числа в единицах, заданных значением item;
- GetDate() – возвращает текущую системную дату и время.

2.4.4. Многотабличные запросы

Важнейшей особенностью запросов SQL является их способность определять связи между несколькими таблицами и выводить информацию из них в терминах этих связей. При многотабличном запросе таблицы, представленные в виде списка в предложении FROM, отделяются друг от друга запятыми. Предикат запроса может ссылаться на любой столбец любой связанной таблицы и, следовательно, может использоваться для связи между ними. Обычно предикат сравнивает значения в столбцах различных таблиц, чтобы определить, удовлетворяет ли WHERE установленному условию.

Пример 2.6. Многотабличные запросы

1. Вывести модели самолетов, назначенные на рейсы:

```
SELECT Route.IdRoute, Airplane.Model
  FROM Airplane, Route
  WHERE Route.IdAirplane=Airplane.IdAirplane;
```

IdRoute	Model
98	A350–900
112	Boeing 787–8
113	A350–900
116	A300
117	A300
120	Boeing 787–10
121	Boeing 787–10
201	Ty–204–100
321	Boeing 787–8
421	ATR 42
455	SSJ–95

Данный запрос можно записать, используя ключевые слова **INNER JOIN**:

```
SELECT Route.IdRoute, Airplane.Model
  FROM Airplane INNER JOIN Route ON
     Route.IdAirplane=Airplane.IdAirplane;
```

2. Вывести сведения о самолетах вместимостью более 230 человек, вылетающих из Самары:

```
SELECT Airplane.IdAirplane, Airplane.Model, Airplane.Capacity
  FROM Airplane, Route
  WHERE Airplane.IdAirplane=Route.IdAirplane and
     Route.PointOfDept='Самара' and Airplane.Capacity>230;
```

IdAirplane	Model	Capacity
105	A350–900	420
107	Boeing 787–8	250
108	Boeing 787–10	440

3. Вывести информацию о фамилиях пилотов для каждого номера экипажа.

При написании данного запроса будем использовать две таблицы: Worker и AirCrew. Стоит отметить, что в таблице AirCrew хранятся только коды сотрудников, которые ссылаются на таблицу Worker.

Чтобы получить список кодов экипажей с первыми пилотами, необходимо написать запрос:

```
SELECT AirCrew.IdAirCrew, Worker.Name
  FROM AirCrew  INNER JOIN Worker
     ON AirCrew.IdPilot1=Worker.IdWorker;
```

Для того чтобы вывести список вторых пилотов, необходимо написать аналогичный запрос:

```
SELECT AirCrew.IdAirCrew, Worker.Name
  FROM AirCrew  INNER JOIN Worker
     ON AirCrew.IdPilot2=Worker.IdWorker;
```

Трудность возникнет, когда мы попытаемся вывести общий список с первыми и вторыми пилотами одновременно:

```
SELECT AirCrew.IdAirCrew, Worker.Name, Worker.Name
  FROM AirCrew  INNER JOIN Worker
     ON AirCrew.IdPilot1=Worker.IdWorker
  INNER JOIN Worker
     ON AirCrew.IdPilot2=Worker.IdWorker;
```

Попытка дважды присоединить одну и ту же таблицу приведет к выводу сообщения об ошибке.

Для повторного использования таблицы Worker нужно придумать другой способ, позволяющий избавиться от путаницы. Следует вернуть из таблицы две разные записи для каждого результата: одну для первого пилота, а другую - для второго. Значительно упростить ситуацию помогли бы две копии таблицы Worker. Решить проблему можно, указав для таблицы Worker два уникальных псевдонима (временных имени) внутри запроса.

Если после имени таблицы во фрагменте запроса SELECT, начинающегося с ключевого слова FROM, написать AS псевдоним, то вы получите временное имя, на которое можно ссылаться в любом другом месте запроса.

Теперь, дважды сославшись на таблицу Worker с помощью разных псевдонимов, мы сможем выполнить тройное объединение (в котором две таблицы на самом деле являются одной) и получить желаемый результат:

```
SELECT IdAirCrew
        P1.Name AS 'Пилот 1'
        P2.Name AS 'Пилот 2'
 FROM AirCrew  AS E INNER JOIN Worker as P1
   ON E.IdPilot1=P1.IdWorker
 INNER JOIN Worker AS P2
 ON E.IdPilot2=P2.IdWorker;
```

IdAirCrew	Пилот 1	Пилот 2
901	Петров И.А.	Иванов М.Г.
902	Киселев А.А.	Комаров Т.Н.
904	Комаров Т.Н.	Фадеев П.В.
905	Петров И.А.	Понкратов А.А.
909	Киселев А.А.	Иванов М.Г.
910	Петров И.А.	Фадеев П.В.

Кроме *внутреннего соединения* (внутренним принято считать соединение с использованием предложения WHERE) существует еще *внешнее соединение*.

В общем случае синтаксис внешнего соединения выглядит следующим образом:

FROM <Таблица1> <вид соединения> JOIN <Таблица2> ON
<Условие соединения>;

Вид соединения определяет главную (ведущую) таблицу в соединении и может определяться следующими служебными словами:

- LEFT – левое внешнее соединение, когда ведущей является таблица слева от вида соединения;
- RIGHT – правое внешнее соединение, когда ведущей является таблица справа от вида соединения;
- FULL – полное внешнее соединение, когда обе таблица равны в соединении.

По правилу внешних соединений, ведущая таблица должна войти в результат запроса всеми своими записями, независимо от наличия соответствующих записей в присоединяемой таблице.

Рассмотрим варианты использования различных видов соединения на примере двух таблиц: в таблице ГОРОД (City) содержится информация о городах, в которые летают самолеты, а в таблице АЭРОПОРТ (Aero) – информация о всех аэропортах в этих городах.

CITY

IdCity	City
10	Самара
20	Москва
30	Казань
40	Волгоград

AERO

NameA	City
Курумоч	Самара
Домодедово	Москва
Внуково	Москва
Гурмак	Волгоград

4. Соединить таблицы ГОРОД и АЭРОПОРТ, используя левое внешнее соединение:

```
SELECT IdCity, C.City, NameA
  FROM City C LEFT JOIN AERO A ON C.City=A.City;
```

Стоит обратить внимание, что при отсутствии соответствия строк правой и левой таблицы пустые ячейки заполняются NULL–значениями.

IdCity	City	NameA
10	Самара	Курумоч
20	Москва	Домодедово
20	Москва	Внуково
30	Казань	NULL
40	Волгоград	Гурмак

5. Соединить таблицы ГОРОД и АЭРОПОРТ, используя правое внешнее соединение:

```
SELECT IdCity, C.City, NameA
  FROM City C RIGHT JOIN AERO A ON C.City=A.City;
```

IdCity	City	NameA
10	Самара	Курумоч
20	Москва	Домодедово
20	Москва	Внуково
40	Волгоград	Гурмак

UNION – специальный оператор, с помощью которого можно из двух и более запросов построить единое результирующее множество. По сути UNION не является объединением, как рассмотренные ранее варианты JOIN. Его работа больше похожа на присоединение выходных данных одного запроса к выходным данным другого запроса. В то время как JOIN комбинирует данные, добавляя столбцы, UNION комбинирует данные, добавляя строки.

Когда два (или более) запроса подвергаются объединению, их столбцы вывода должны быть совместимы для объединения. Это означает, что каждый запрос должен указывать одинаковое число столбцов и в том же порядке, что и первый, второй, третий и так далее, и типы столбцов в объединяемых запросах должны быть

совместимы. UNION будет автоматически исключать дубликаты строк из вывода.

6. Вывести список номеров экипажей, которые назначены на рейсы из Самары и первый пилот которых имеет код сотрудника, равный 1.

```
SELECT IdAirCrew
  FROM Route
  WHERE PointOfDept='Самара'
UNION
SELECT IdAirCrew
  FROM AirCrew
  WHERE IdPilot1=1;
```

IdAirCrew
901
902
905
909
910

1.4.5. Подзапросы

Запросы могут управлять другими запросами. Для этого нужно поместить запрос внутри предиката другого запроса и использовать выходные данные внутреннего запроса в предикате внешнего.

Подзапрос может быть вложен в инструкции SELECT, INSERT, UPDATE или DELETE, а также в другой подзапрос.

Условно подзапросы иногда подразделяют на три типа, каждый из которых является сужением предыдущего:

- табличный (многострочный) подзапрос возвращает набор строк и столбцов;
- строковый (однострочный) подзапрос возвращает значения одного или нескольких столбцов таблицы, но в виде единственной строки;
- скалярный подзапрос возвращает значение, выбираемое из пересечения одного столбца в одной строке, т.е. единственное значение.

Подзапросы бывают *коррелированные* и *некоррелированные*.

Некоррелированный подзапрос может вычисляться как независимый запрос. Результаты подзапроса подставляются в основной оператор или внешний запрос. *Коррелированные подзапросы* от вложенных отличает то, что обмен информацией между коррелированными подзапросами и основными запросами осуществляется в обоих направлениях. Фактически, такой запрос осуществляется в три этапа [13]:

- внешний запрос извлекает запись и передает ее вложенному запросу;
- вложенный запрос выполняет операции на основе полученного значения или значений;
- вложенный запрос возвращает результаты своего выполнения внешнему запросу, который использует их для завершения своих действий.

Коррелированные подзапросы могут превратить несколько строк кода в одну, соответственно увеличив скорость выполнения запроса. Однако коррелированные подзапросы относятся к наиболее трудным для понимания элементам SQL, поскольку они идут вразрез с привычным способом мышления.

Подзапросы могут располагаться в различных частях команды SQL SELECT:

- в инструкции SELECT;
- в инструкции FROM;
- в условии WHERE.

Обычно подзапрос добавляется в условие WHERE оператора SQL SELECT. Чтобы выполнить основной запрос, SQL сначала должен оценить внутренний запрос внутри предложения WHERE.

В подзапросах допускается использование агрегатных функций, операторов сравнения, таких как >, < или =. При этом подзапрос всегда стоит справа от оператора сравнения или предиката.

Также подзапросы можно использовать внутри предложения HAVING.

Пример 2.7. Использование подзапросов

1. Выбрать рейсы, на которые назначены самолеты с дальностью полета более 14 тыс. км:

```
SELECT *
  FROM Route
  WHERE IdAirplane IN (SELECT IdAirplane FROM Airplane
    WHERE Distance>14000);
```

IdRoute	IdAirplane	IdAirCrew	PointOf Dept	PointOf Dest	Date
98	105	901	Самара	Иркутск	2018–01–01
113	105	902	Казань	Милан (Италия)	2018–01–02

2. Вывести пофамильный состав всех экипажей:

```
SELECT IdAirCrew,
  (Select Name From Worker Where IdWorker=AirCrew.IdPilot1)
    AS 'Пилот 1',
```

```
(Select Name From Worker Where IdWorker=AirCrew.IdPilot2)
  AS 'Пилот 2',
(Select Name From Worker Where
  IdWorker=AirCrew.IdFlightEng1)  AS 'Бортинженер 1',
(Select Name From Worker Where
  IdWorker=AirCrew.IdFlightEng2)  AS 'Бортинженер 2',
(Select Name From Worker Where
  IdWorker=AirCrew.IdSteward1)  AS 'Бортпроводник 1',
(Select Name From Worker Where
   IdWorker=AirCrew.IdSteward 2)   AS 'Бортпроводник 2',
(Select Name From Worker Where
   IdWorker=AirCrew.IdSteward3)   AS 'Бортпроводник 3',
FROM AirCrew;
```

IdAirCrew	Пилот 1	Пилот 2	Бортинженер 1	Бортинженер 2	Бортпроводник 1	Бортпроводник 2	Бортпроводник3
901	Петров И.А	Иванов М.Г.	Аносов Т.Ю.	Фаизов М.Н.	Кольцова О.М.	Кузьмина Е.С.	Гостева С.А.
902	Киселев А.А.	Комаров Т.Н.	Аносов Т.Ю.	NULL	Яновский И.Р.	Гостева С.А.	Васильева Б.Ю.
904	Комаров Т.Н.	Фадеев П.В.	Рубцов Л.И.	NULL	Котова И.Т.	NULL	NULL
905	Петров И.А.	Понкратов А.А.	Фаизов М.Н.	Котова И.Т.	Кольцова О.М.	Кузьмина Е.С.	Гостева С.А.
909	Киселев А.А.	Иванов М.Г.	Аносов Т.Ю.	NULL	Кузьмина Е.С.	Гостева С.А.	Котова И.Т.
910	Петров И.А.	Фадеев П.В.	Фаизов М.Н.	Аносов Т.Ю.	Кольцова О.М.	Кузьмина Е.С.	Яновский И.Р.

3. Посчитать средний стаж сотрудников по должностям и вывести те должности, у которых средний стаж меньше среднего стажа среди всех сотрудников:

```
SELECT Position, AVG(Seniority)
  FROM Worker
  GROUP BY Position
  HAVING AVG(Seniority)<(SELECT AVG(Seniority) FROM
Worker);
```

Position	
бортинженер	9
бортпроводник	11

Следующие операторы используются для модификации операторов сравнения:

— ALL – оператор, эквивалентный понятию «все».

Например: *> ALL (< ALL)* – больше (меньше) каждого значения элементов результирующего множества.

— ANY (SOME) – оператор, эквивалентный понятию «любой».

Например: = *ANY* – равно одному из значений элементов результирующего множества (эквивалентно использованию предиката IN).

> ANY (< ANY) – больше (меньше) любого значения элементов результирующего множества.

<>*ANY*(…) - не эквивалентно *NOT IN*.

— EXISTS – оператор, эквивалентный понятию «существует». Может использоваться с логическим оператором *NOT*.

4. Вывести информацию о первых пилотах:

```
SELECT *
  FROM Worker S
  WHERE EXISTS (SELECT * FROM AirCrew E WHERE
S.IdWorker=E.IdPilot1);
```

Id Worker	Name	Position	Seniority	BirthDate	Salary	Gender
1	Петров И.А.	пилот	25	25.09.1968	45600	м
6	Киселев А.А.	пилот	30	03.03.1956	45600	м
7	Комаров Т.Н.	пилот	23	15.09.1971	45600	м

5. Вывести информацию о бортпроводниках женского пола, у которых стаж больше, чем у любого бортпроводника мужского пола.

```
SELECT *
  FROM Worker
```

```
WHERE Position='бортпроводник' And Gender='ж'
  And Seniority>ANY(SELECT Seniority FROM Worker
    WHERE Position='бортпроводник' And Gender='м');
```

Id Worker	Name	Position	Seniority	BirthDate	Salary	Gender
10	Кольцова О.М.	борт–проводник	20	03.05.1974	26000	ж

6. Вывести фамилии пилотов, которые входят в состав экипажей в качестве только первых пилотов:

```
SELECT IdWorker, Name
  FROM Worker
  WHERE IdWorker IN (SELECT DISTINCT IdPilot1 FROM
AirCrew
    WHERE IdPilot1<>ALL (SELECT IdPilot2 FROM AirCrew));
```

IdWorker	Name
1	Петров И.А.
6	Киселев А.А.

2.5. Обработка NULL–значений

В стандарт SQL включено понятие NULL–значения. NULL не является конкретным значением, поэтому необходимо понимать, как операторы сравнения будут с ним работать. Так как значения NULL рассматриваются как неизвестные, два значения NULL, сравниваемые друг с другом, не считаются равными. В выражениях, использующих арифметические операторы, если какие-либо из операндов равны NULL, результат также равен NULL.

Правила для работы с NULL–значениями:

- предложение GROUP BY объединяет все NULL–значения в одну группу;
- предикат DISTINCT оставляет только одно значение NULL;
- функция AVG не учитывает NULL–значения, и сумма значений поля делится на количество ненулевых значений;
- если список, модифицированный оператором ALL, содержит NULL–значение, то результирующий запрос будет пуст, т.к. нельзя сравнить никакое значение с NULL–значением;
- выражение <>ANY(…) не выполняется для NULL–значений.

Для обработки NULL–значений предназначено несколько функций. В MS SQL Server Функция ISNULL() принимает в качестве входных параметров переменную или выражение и проверяет, не является ли оно NULL. Если переданное значение действительно NULL, то функция возвращает некоторое предварительно определенное выражение. Если оригинальное выражение не NULL, тогда функция возвращает полученное выражение без изменений. Синтаксис функции:

ISNULL(<проверяемое выражение>,

<возвращаемое значение, если проверяемое значение NULL>)

Пример 2.8. Обработка NULL-значений

Вывести таблицу, содержащую следующую информацию: код экипажа и соответствующие этому экипажу фамилии бортинженеров; в тех случаях, когда бортинженер не назначен, вывести надпись – 'отсутствует':

```
SELECT
   IdAirCrew,
```

```
    ISNULL((SELECT Name FROM Worker WHERE
      IdWorker=AirCrew.IdFlightEng1),  'отсутствует')
      AS 'Бортинженер1',
    ISNULL((SELECT Name FROM Worker WHERE
      IdWorker=AirCrew.IdFlightEng2), 'отсутствует')
     AS 'Бортинженер2'
FROM AirCrew;
```

IdAirCrew	Бортинженер1	Бортинженер2
901	Аносов Т.Ю.	Фаизов М.Н.
902	Аносов Т.Ю.	отсутствует
904	Рубцов Л.И.	отсутствует
905	Фаизов М.Н.	Котова И.Т.
909	Аносов Т.Ю.	отсутствует
910	Фаизов М.Н.	Аносов Т.Ю.

2.6. Представления

Механизм представлений (VIEW) является мощным средством СУБД, позволяющим скрыть реальную структуру БД от некоторых пользователей.

Представление является некоторым хранимым в БД запросом, созданным на основе команды SELECT. Представление реально не содержит данных. Запрос, определяющий представление, выполняется тогда, когда к представлению происходит обращение с другим запросом, например, *SELECT, UPDATE* и т.д.

Назначение представлений:

1) хранение сложных запросов;
2) представление данных в удобном для пользователя виде;
3) сокрытие конфиденциальной информации;
4) предоставление дифференцированного доступа к данным.

При создании представления информация о нем записывается в каталог БД под собственным именем, а у пользователя создается впечатление, что в БД реально существует отношение с таким именем. При этом любые изменения в данных адекватно отобразятся в представлении – в этом есть его отличие от запроса к БД, на которое представление очень похоже.

Создание представления выполняется командой CREATE VIEW:

CREATE VIEW <имя представления> [(<имя столбца>.,..)]
AS <запрос> [WITH CHECK OPTION];

Пример 2.9. Создание представлений

1. Создать представление «Пилот», которое будет содержать информацию только о пилотах авиакомпании:

```
CREATE VIEW Pilot
 AS SELECT IdWorker, Name, Position, Seniority
  FROM Worker
  WHERE Position='пилот';

SELECT * FROM Pilot;
```

IdWorker	Name	Position	Seniority
1	Петров И.А.	пилот	25
2	Иванов М.Г.	пилот	20
6	Киселев А.А.	пилот	30
7	Комаров Т.Н.	пилот	23
14	Понкратов А.А.	пилот	17
15	Фадеев П.В.	пилот	13
16	Наумов К.Ю.	пилот	32

2. Создать представление «Рейс», которое будет содержать информацию о номерах рейсов, городах вылета, городах назначения и моделях самолетов, назначенных на эти рейсы:

```
CREATE VIEW VRoute (Route, Model, PointOfDeptet,
   PointOfDestnach)
 AS SELECT R.IdRoute, S.Model, R.PointOfDept,
R.PointOfDest
  FROM Route R, Airplane S
  WHERE R.IdAirplane=S.IdAirplane;
SELECT * FROM VRoute;
```

Route	Model	PointOfDeptet	PointOfDestnach
98	A350–900	Самара	Иркутск
112	Boeing 787–8	Самара	Берлин
113	A350–900	Казань	Милан
116	A300	Волгоград	Челябинск
117	A300	Москва	Екатеринбург
120	Boeing 787–10	Самара	Милан
121	Boeing 787–10	Казань	Москва
201	Ту–204–100	Самара	Москва
321	Boeing 787–8	Казань	Челябинск
421	ATR 42	Самара	Казань
455	SSJ–95	Уфа	Самара

Запрос, на основании которого создаётся представление, называется *определяющим запросом*, а таблицы, к которым происходит обращение в определяющем запросе – *базовыми таблицами* [7].

Представление может быть *обновляемым* и *необновляемым*. Обновляемым является представление, при обращении к которому можно обновить базовую таблицу.

Обновляемое представление определяется следующими критериями:

- основывается только на одной базовой таблице;
- содержит первичный ключ этой таблицы;

- не содержит DISTINCT в своем определении;
- не использует GROUP BY или HAVING в своем определении;
- по возможности не применяет в своем определении подзапросы;
- не использует константы или выражения значений среди выбранных полей вывода;
- включает каждый столбец базовой таблицы, имеющий атрибут NOT NULL;
- оператор SELECT не использует агрегирующие (итоговые) функции, соединения таблиц, хранимые процедуры и функции, определенные пользователем;
- основывается на одиночном запросе, поэтому объединение UNION не разрешено.

Эти ограничения являются очень сильными. В реализациях они могут быть ослаблены. Различия между обновляемыми представлениями и представлениями, предназначенными только для чтения, не случайны. Цели, для которых их используют, различны. С модифицируемыми представлениями, в основном, обходятся точно так же, как и с базовыми таблицами. Фактически, пользователи не могут даже осознать, является ли объект, который они запрашивают, базовой таблицей или представлением, т.е. прежде всего это средство защиты для сокрытия конфиденциальных или не относящихся к потребностям данного пользователя частей таблицы.

3. Необходимо обновить базовую таблицу Worker через представление Pilot (созданное в предыдущем примере):

```
UPDATE Pilot
SET Seniority=30 WHERE Name='Петров И.А.';
```

Изменения будут произведены в базовой таблице и отразятся в представлении.

SELECT * FROM Pilot;

IdWorker	Name	Position	Seniority
1	Петров И.А.	пилот	30
2	Иванов М.Г.	пилот	20
6	Киселев А.А.	пилот	30
7	Комаров Т.Н.	пилот	23
14	Понкратов А.А.	пилот	17
15	Фадеев П.В.	пилот	13
16	Наумов К.Ю.	пилот	32

В свою очередь, необновляемые представления позволяют создавать целый арсенал сложных запросов, которые можно выполнить и повторить снова, сохраняя полученную информацию. Результаты этих запросов могут затем использоваться в других запросах, что позволит избежать сложных предикатов и снизить вероятность ошибочных действий.

При обновлении представлений может возникнуть следующая проблема: вносимые изменения выходят за рамки определяющего запроса и поэтому не могут быть отражены в представлении. Если необходимо защитить данные от такого изменения, необходимо добавить в команду создания представления ключевые слова WITH CHECK OPTION: тогда система отвергнет изменения, выходящие за рамки определяющего запроса [7].

В общем случае WITH CHECK OPTION следует использовать для всех обновляемых представлений, если только нет каких-либо специфических причин для помещения в таблицу значений, которые не будут отображаться в представлении. В представлении “только для чтения” такая опция не нужна [4]. Поясним как работает опция WITH CHECK OPTION на следующем примере.

4. Для данного примера будем использовать ранее созданное представление Pilot. Допустим, необходимо выполнить следующую команду:

```
INSERT INTO Pilot
VALUES (30, 'Орлов М.М. ', 'механик', 11);
```

С помощью команды INSERT и обновляемого представления Pilot в таблицу Worker будет вставлена новая строка. Однако, когда эта информация будет добавлена, строка исчезнет из представления, поскольку значение должности 'механик' не удовлетворяет условию WHERE представления Pilot (должность должна быть – 'пилот').

Это может стать проблемой, поскольку данные уже находятся в таблице Worker, но пользователь этого не видит и не в состоянии выполнить удаление или модификацию данной информации. Для исключения подобных ситуаций необходимо включить опцию WITH CHECK OPTION в определение представления:

```
CREATE VIEW Pilot
 AS SELECT IdWorker, Name, Position, Seniority
  FROM Worker
  WHERE Position='пилот'
  WITH CHECK OPTION;
```

После этого вышеупомянутая вставка будет отклонена системой.

Существуют и другие виды представлений, описанные в источнике [7]:

- *горизонтальное представление*. Этот вид представления широко применяется для уменьшения объема реальных таблиц в обработке и ограничения доступа пользователей к закрытой для них информации;
- *вертикальное представление*. Этот вид представления практически соответствует выполнению операции проек-

тирования некоторого отношения на ряд столбцов. Он используется в основном для скрытия информации, которая не должна быть доступна в конкретной внешней модели;

- *сгруппированные представления*. Эти представления содержат запросы, которые имеют группировку. Сгруппированные представления всегда должны содержать список столбцов. Они могут использовать агрегированные функции в качестве результирующих столбцов, а в дальнейшем это представление может использоваться как виртуальная таблица, например, в других запросах;
- *объединенные представления*. Часто представления базируются на многотабличных запросах. Такое использование позволяет упростить разработку пользовательского интерфейса, сохранив при этом корректность схемы базы данных.

Синтаксис, позволяющий исключить представление из базы данных, похож на синтаксис удаление базовой таблицы:

DROP VIEW <имя_представления >

2.7. Контрольные вопросы к главе 2

1. Назовите основные типы данных SQL.
2. Поясните следующие ограничения и ключевые слова: NULL, PRIMARY KEY, UNIQUE, DEFAULT, CHECK.
3. Какими командами можно создать таблицу, удалить таблицу, изменить определение сущностей таблицы?
4. Назовите основные команды модификации данных.
5. Поясните ключевые операторы команды SELECT: WHERE, ORDER BY, GROUP BY, HAVING, UNION, DISTINCT, ALL.

6. Назовите предикаты, используемые в запросах.

7. Для чего нужны функции агрегирования? Приведите примеры.

8. Каковы особенности выполнения многотабличных запросов?

9. Как в запросах обрабатываются NULL–значения?

10. Какие виды представлений вы знаете?

11. Как можно защитить представление от изменений, выходящих за рамки запроса, на основе которого было сформировано представление?

12. Какие виды соединений предусмотрены первым стандартом SQL?

13. Как с помощью операторов SQL можно получить декартово произведение двух таблиц?

14. Чем отличается соединение от объединения?

15. Какой синтаксис имеют соединения?

16. Какие виды соединений вы знаете?

17. Для чего необходимы вложенные запросы?

18. Какие ограничения налагаются на вложенные запросы?

19. Как можно использовать предикат IN или служебное слово ANY?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Базы данных [Текст]: учебник для высших учебных заведений / [А.Д. Хомоненко, В.М. Цыганков, М.Г. Мальцев]; под ред. проф. А.Д. Хомоненко.– 6–е изд., доп.– СПб.: КОРОНА–Век, 2009.– 736 с.
2. Вендров, А.М. CASE–технологии. Современные методы и средства проектирования информационных систем [Текст] / А.М. Вендров. – М.: Финансы и статистика, 2005. – 176 с.
3. Глушаков, С.В. Базы данных [Текст]: учеб. издание / С.В. Глушаков, Д.В. Ломотько. – Харьков: Фолио; М.: ООО «Издательство АСТ», 2002. – 504с.
4. Грабер, М. SQL: пер. с анг. [Текст] / М. Грабер.– М: Издательство «Лори», 2003. – 645 с.
5. Дейт, К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL: пер. с анг. [Текст] / К. Дж. Дейт. – СПб.: Символ–Плюс, 2010. – 480 с.
6. Додонов, М.В. Автоматизированное проектирование баз данных [Текст]: метод. указания по лабораторным работам и курсовому проектированию для студентов всех специальностей дневной и заочной форм обучения / М.В. Додонов, Д.К. Тюмиков. – Самара: СамГАПС, 2005. – 21 с.
7. Карпова, И.П. Базы данных [Текст]: учеб. пособие / И.П. Карпова. – СПб.: Питер, 2013. – 240 с.
8. Маклаков, С.В. BPwin и ERwin: CASE–средства для разработки информационных систем [Текст] / С.В. Маклаков. – М.: Диалог–Мифи, 1999. – 295 с.

9. Менеджмент качества. Автоматизация СМК. Моделирование бизнес процессов. CASE средства. URL: http://www.kpms.ru/Automatization/CASE_tools.htm (дата обращения 23.02.2019).
10. Мокеев, В.В. Методология моделирования данных в среде ERWin [Текст]: учеб. пособие для лабораторных работ / В.В. Мокеев. – Челябинск: Изд. ЮУрГУ, 2005. – 88 с.
11. Точилкина, Т.Е. Принципы создания информационных систем и моделирования бизнес–процессов с использованием пакета программ AllFusion Modeling Suite. Часть II. Моделирование данных и проектирование баз данных с ERwin Date Modeler [Текст]: учеб. пособие / Т.Е. Точилкина. – М.: изд. «Академии бюджета и казначейства», 2009.– 167 с.
12. Фиайли, К. SQL: пер. с анг. [Текст] / К. Фиайли. – М.: ДМК Пресс, 2003.– 451 с.
13. SQL Server 2000. Программирование [Текст]. В 2 ч. Ч. 2.: справочное издание / [Р. Вьейра]; Часть I; пер. с англ.; под ред. С.М. Молявко. – .М.:БИНОМ. Лаборатория знаний, 2004. – 735 с.

ПРИЛОЖЕНИЕ

ЛАБОРАТОРНЫЕ РАБОТЫ

Целью лабораторного практикума является изучение основ проектирования и реализации баз данных. В процессе выполнения лабораторных работ приобретаются практические навыки анализа и моделирования предметной области, включая принципы построения на концептуальном, логическом и физическом уровнях. В ходе выполнения лабораторных работ приобретаются навыки применения структурированного языка запросов SQL для создания и обработки данных в реляционных базах данных.

Практикум предполагает последовательное выполнение лабораторных работ (1-3), при этом для выполнения заданий практикума предлагаются различные варианты предметных областей.

Первая лабораторная работа предполагает создание модели базы данных с помощью CASE-средства ERwin Data Modeler (бесплатную версию данного программного продукта Community Edition можно получить на сайте разработчика http://erwin.com). Теоретический материал для выполнения данной лабораторной работы содержится в главе 1.

Вторая и третья лабораторные работы требуют знаний структурированного языка запросов SQL. Теоретический материал для выполнения этих работ приведен в главе 2. Данные лабораторные работы потребуют подключения к СУБД. Лабораторные работы 2 и 3 предлагается выполнить с использованием СУБД Microsoft SQL Server (можно воспользоваться бесплатной версией SQL Server Express, предоставляемой на портале Microsoft).

Описание лабораторных работ

Лабораторная работа № 1
Логическое и физическое моделирование базы данных, реализация реляционной схемы БД в выбранной СУБД

Цель работы: приобретение навыков создания логической и физической модели базы данных с помощью CASE-средства. Приобретение навыков реализации базы данных в рамках реляционной модели с помощью СУБД.

Содержание работы:

1. Изучение основ создания логической и физической модели данных с помощью CASE-средства.
2. Изучение особенностей проектирования реляционных баз данных.

Последовательность выполнения работы:

1. С помощью Case-средства создайте логическую модель базы данных в соответствии с вариантом задания.
   - 1.1. Определите сущности, описывающие заданную предметную область.
   - 1.2. Определите атрибуты, характеризующие каждую сущность.
   - 1.3. Укажите домены для всех атрибутов сущностей.
   - 1.4. Определите первичные ключи отношений.
   - 1.5. Определите связи между сущностями (идентифицирующая/неидентифицирующая, полная/неполная категория, один ко многим/ многие ко многим)
2. Для созданной модели опишите физический уровень представления данных, пользуясь выбранным в п.1 Case-средством.
3. Выберите необходимую СУБД, для которой нужно создать схему данных, и сгенерируйте DDL-скрипт для выбранной СУБД.

4. Внесите необходимые изменения в имена таблиц, полей, связей и их свойства.

5. Внесите необходимые изменения в созданный DDL-скрипт.

6. Выполните DDL-скрипт и заполните созданные таблицы данными командой SQL Insert (не менее 12–15 записей в таблице).

Лабораторная работа № 2
Обработка данных в реляционных БД с помощью структурированного языка запросов SQL

Цель работы: приобретение навыков использования языка SQL. Выборка данных из таблиц, добавление, удаление, редактирование информации.

Содержание работы:

1. Изучение основ использования языка SQL: выборка данных из одной таблицы, выборка данных из связанных таблиц, сортировка и группировка результатов запроса, использование агрегатных функций и специальных предикатов.

2. Изучение SQL команд для модификации данных.

Последовательность выполнения работы:

1. Напишите SQL запросы, реализующие функциональные требования системы в соответствии с вариантом задания.

2. Придумайте и напишите SQL запросы, которые будут необходимы для предметной области (в соответствии с вариантом задания):

2.1. Запрос на выборку избранных полей таблицы, с использованием синонима (алиаса) и сортировкой записей (ORDER BY).

2.2. Запрос с использованием сортировки (ORDER BY) и группировки (GROUP BY).

2.3. Запрос с использованием предложения DISTINCT.

2.4. Запрос с использованием операций сравнения.

2.5. Запросы для предикатов: IN, BETWEEN, LIKE, ISNULL.

2.6. Запросы с использованием агрегатных функций (COUNT, SUM, AVG, MAX, MIN), производящие обобщенную групповую обработку значений полей (используя ключевые фразы GROUP BY и HAVING).

2.7. Запрос на выборку данных из двух связанных таблиц. Выбрать несколько полей, по которым сортируется вывод.

2.8. Многотабличный запрос с использованием внутреннего и внешнего соединения.

2.9. Многотабличный запрос с использованием оператора UNION.

3. Создайте SQL команды для модификации данных (INSERT, UPDATE, DELETE).

Лабораторная работа № 3
Использование подзапросов и представлений

Цель работы: приобретение навыков использования подзапросов в команде SQL SELECT. Создание и работа с представлениями.

Содержание работы:

1. Изучение основ написания подзапросов в различных частях команды SQL SELECT.

2. Изучение основ создания виртуальных таблиц (представлений). Создание обновляемых представлений. Использование команды WITH CHECK OPTION.

Последовательность выполнения работы:

1. Придумайте и напишите SQL запросы, которые будут необходимы для предметной области (в соответствии с вариантом задания):

1.1. Запрос с применением подзапроса в части WHERE команды SQL SELECT и внутри предложения HAVING.

1.2. Запрос с применением подзапроса с применением следующих операторов: ALL, EXIST, ANY.

2. Создайте представления, основанные на запросах из пункта 1.

3. Создайте представления с выборкой, сортировкой, группировкой, левым, правым и внешним объединением.

4. Создайте обновляемые представления для всех таблиц. Проверьте работоспособность созданных представлений командами SQL: Select, Insert, Update и Delete.

5. Для обновляемого представления примените команду WITH CHECK OPTION. Объясните смысл ее применения.

Виды предметных областей

Вариант № 1

Предметная область: Администрирование платного ресурса в социальной сети.

В известной социальной сети организован платный ресурс с ограничением прав доступа. Администратор ведет реестр подписчиков и ресурсов, а также занимается организацией доступа различного уровня в зависимости от оплаты ресурсов пользователями. Ресурс представляет собой электронный контент, к которому может быть открыт платный/бесплатный срочный/бессрочный доступ.

Основные предметно-значимые сущности: Подписчик, Ресурс, Доступ.

Основные предметно-значимые атрибуты сущностей:

- подписчик – ФИО, электронный адрес и телефон.
- ресурс – название, цена, краткая аннотация, длительность доступа;

- доступ – код подписчика, код ресурса, дата открытия доступа, дата закрытия доступа, стоимость.

Основные требования к функциям системы:

- выбрать все активные ресурсы на текущую дату;
- выбрать все ресурсы со стоимостью, ниже заданной;
- выбрать ресурсы, на которые подписан определенный подписчик;
- выбрать ресурсы, которые оплатил определенный подписчик;
- выбрать подписчиков, которые подписаны на заданный ресурс;
- выбрать подписчиков с максимальным количеством ресурсов;
- выбрать зарегистрированных подписчиков, не подписанных ни на один платный ресурс;
- выбрать ресурсы, пользующиеся наибольшим спросом;
- выбрать ресурсы, которые в настоящий момент не оплатил ни один подписчик.

Вариант № 2

Предметная область: Организация учебного процесса.

База данных описывает организацию обучения в высшем учебном заведении. Преподаватель работает на кафедре, студенты учатся в группах. Один преподаватель может вести несколько дисциплин, одна дисциплина может преподаваться в нескольких семестрах несколькими преподавателями.

Основные предметно-значимые сущности: Преподаватель, Студент, Дисциплина.

Основные предметно-значимые атрибуты сущностей:

- преподаватель – фамилия, имя, отчество, номер паспорта, кафедра;

- студент – фамилия, имя, отчество, дата рождения, номер группы;
- дисциплина – название, семестр, количество часов.

Основные требования к функциям системы:

- выбрать преподавателя, ведущего наибольшее количество дисциплин;
- выбрать группы, в которых ведет указанный преподаватель;
- выбрать группы, в которых у студентов в настоящее время наибольшая успеваемость;
- выбрать группы, в которых преподают преподаватели заданной кафедры;
- выбрать дисциплины, по которым у студентов в настоящее время наибольшая успеваемость;
- выбрать дисциплины, которые ведет указанный преподаватель;
- выбрать дисциплины, изучаемые группой студентов в заданном семестре;
- выбрать дисциплины с максимальным количеством часов в указанном семестре;
- выбрать дисциплины, которые не изучаются студентами в настоящий момент.

Вариант № 3

Предметная область: Организация учета читателей в библиотеке.

Библиотека производит выдачу книг читателям и прием книг в библиотеку от читателей в заданный срок.

Основные предметно-значимые сущности: Книга, Читатель, Формуляр.

Основные предметно-значимые атрибуты сущностей:

- книга – шифр, автор книги, название, год издания, цена;
- читатель - номер читательского билета, ФИО, адрес и телефон читателя, год рождения;
- формуляр - номер читательского билета, шифр книги, дата выдачи.

Основные требования к функциям системы:

- выбрать книги, пользующиеся наибольшим спросом у читателей заданной возрастной категории;
- выбрать все свободные экземпляры определенной книги;
- выбрать все книги со стоимостью выше указанной;
- выбрать книги, которые находятся у заданного читателя;
- выбрать читателей, которые брали ту или иную книгу с указанием даты выдачи книги и даты сдачи книги читателем;
- выбрать читателей, у которых на руках находится более пяти книг;
- выбрать читателей, взявших наибольшее количество книг за указанный период;
- выбрать книги, не пользующиеся спросом;
- выдать список всех читателей – должников.

Вариант № 4

Предметная область: Организация склада.

На складе организовано хранение и продажа товаров покупателям. Товары имеют срок годности, после которого их реализация запрещена. Товар одного наименования может поставляться различными поставщиками по различной цене.

Основные предметно-значимые сущности: Товар, Покупатель

Основные предметно-значимые атрибуты сущностей:

- товары – название, поставщик, срок годности, отпускная цена;

- покупатель – название, контактное лицо, телефон, адрес;
- продажа - код товара, количество товара, дата.

Основные требования к функциям системы:

- выбрать все товары, купленные заданным покупателем за указанный период;
- выбрать товары, у которых истекает срок годности в указанный период;
- выбрать товары, поставляемые определенным поставщиком;
- выбрать товары определенного наименования, которых сейчас нет на складе;
- выбрать товары, пользующиеся максимальным спросом;
- выбрать товары, не пользующиеся спросом;
- выбрать покупателей, оплативших, но не забравших товар со склада;
- проверить, были ли реализованы со склада товары с истекшим сроком годности.

Вариант № 5

Предметная область: Банковские счета.

Клиент банка может хранить свои сбережения на нескольких счетах. При этом счета могут иметь разные процентные ставки в зависимости от типа вклада, который выбрал клиент для этого счета.

Основные предметно-значимые сущности: Клиент, Счет, Вклад.

Основные предметно-значимые атрибуты сущностей:

- клиент – ФИО клиента, номер паспорта, адрес, телефон;
- счет– №счета, дата открытия счета, дата закрытия счета, сумма на счете*;*
- вклад – наименование; срок хранения(месяцев), ставка (% годовых).

Основные требования к функциям системы:

- выбрать все счета определенного клиента;
- выбрать всех клиентов, открывших указанный вклад;
- выбрать клиентов с максимальным количеством счетов;
- выбрать клиентов с максимальной суммой на всех счетах;
- выбрать вклады, пользующиеся максимальным спросом;
- выбрать вклады, не пользующиеся спросом;
- выбрать все счета с указанной процентной ставкой;
- выбрать все активные (открытые) счета на текущую дату;
- выбрать все неактивные (закрытые) счета в указанный период времени;
- выбрать все счета определенного клиента, закрытые в прошлом месяце.

Вариант № 6

Предметная область: Организация работы цветочного питомника.

Питомник осуществляет выполнение заказов на поставку цветочных растений и кустарников.

Основные предметно-значимые сущности: Растение, Заказчик, Заказ.

Основные предметно-значимые атрибуты сущностей:

- растение – название, тип, цена;
- заказчик – ИНН, название заказчика, адрес заказчика, телефон;
- заказ – номер заказа, срок начала, срок окончания, сумма заказа, тип растений, количество.

Основные требования к функциям системы:

- выбрать все договоры определенного клиента;
- выбрать всех клиентов, купивших определенное растение;

- выбрать договоры, актуальные на текущую дату;
- выбрать все исполненные заказы;
- выбрать все растения, указанные в определенном заказе;
- выбрать растения, не пользующиеся спросом;
- выбрать растения, пользующиеся максимальным спросом;
- выбрать все кустарники, купленные в заданный период;
- выбрать всех заказчиков, купивших растения на сумму, больше заданной.

Вариант № 7

Предметная область: Организация работы автозаправок.

На автозаправках федеральной сети реализуется автомобильное топливо всех видов: бензин-92, бензин-95, бензин-98, дизельное топливо, газ пропан-бутан, газ метан - с использованием специальных карт лояльности клиентов.

Основные предметно-значимые сущности: Клиент, Автозаправка, Топливо.

Основные предметно-значимые атрибуты сущностей:

- клиент – № карты лояльности клиента, Ф.И.О. клиента, электронный адрес, телефон;
- автозаправка – № заправки, адрес заправки, телефон;
- топливо – вид, цена за литр;
- продажа – номер чека, дата, вид топлива, количество литров, сумма.

Основные требования к функциям системы:

- выбрать все покупки топлива определенным клиентом;
- выбрать заправки, которыми чаще всего пользуется определенный клиент;
- выбрать все продажи по заданной карте лояльности;
- выбрать все продажи, при которых не была предъявлена карта лояльности;

- выбрать виды топлива, пользующиеся максимальным спросом;
- выбрать клиента, купившего максимальное количество топлива за указанный период;
- выбрать все продажи топлива указанной марки за указанный период;
- выбрать все виды топлива, покупаемые по заданной карте лояльности.

Вариант № 8

Предметная область: Организация работы отдела продаж косметической продукции.

Производственно-хозяйственное предприятие выпускает различную косметическую продукцию: кремы, шампуни, бальзамы и др. Некоторые наборы косметических средств составляют одну линию – бренд.

Предприятия торговли и сервиса(клиенты) осуществляют заказы у производителя на поставку им определенных видов продукции с указанием необходимого количества и даты поставки.

Основные предметно-значимые сущности: Товар, Бренд, Клиент, Заказ.

Основные предметно-значимые атрибуты сущностей:

- товар – наименование товара, код бренда, единица измерения, цена;
- бренд – код бренда, наименование бренда;
- клиент – ИНН заказчика, наименование, адрес, телефон;
- заказ – код товара, количество товара в заказе, дата заказа, отметка о выполнении;

Основные требования к функциям системы:

- выбрать все заказы определенного клиента за указанный период;

- выбрать все заказы определенного товара за указанный период;
- выбрать все заказы определенного бренда за указанный период;
- выбрать товары, пользующиеся максимальным спросом;
- выбрать бренды, пользующиеся максимальным спросом;
- выбрать бренды, не пользующиеся спросом;
- выбрать бренд, с максимальной суммарной стоимостью заказав за указанный период;
- выбрать клиентов с максимальным количеством заказов за определенный период;
- выбрать все выполненные заказы;
- выбрать все невыполненные заказы.

Вариант № 9

Предметная область: Организация работы издательства учебных изданий.

Издательский центр заключает с авторами контракты на издание книг, учебников, учебных пособий, методических указаний. Центр издает написанные книги и продает их заказчикам: организациям, магазинам, библиотекам и др. За изданные книги авторы получают гонорары.

Основные предметно-значимые сущности: Автор, Контракт, Издание.

Основные предметно-значимые атрибуты сущностей:

- автор – ФИО, адрес, телефон;
- контракт – номер контракта, дата заключения контракта, срок контракта (лет), контракт завершен или нет;
- издание – шифр книги, название, тираж, дата выхода из печати, себестоимость, цена продажи, гонорар;

Основные требования к функциям системы:

- выбрать все заказы определенного клиента за указанный период;
- выбрать все заказы определенного издания за указанный период;
- выбрать все заказы определенного автора за указанный период;
- выбрать издания, пользующиеся максимальным спросом;
- выбрать издания, не пользующиеся спросом;
- выбрать все контракты указанного автора;
- выбрать клиентов с максимальным количеством заказов за определенный период;
- выбрать автора с максимальной суммой гонораров за указанный период;
- выбрать издание с максимальной суммарной ценой реализации;
- выбрать все незавершенные контракты;
- выбрать все завершенные контракты.

Вариант № 10

Предметная область: Организация работы фирмы по продажам грузовой спецтехники.

Коммерческая фирма занимается поставкой различных моделей грузовой спецтехники: грузовики, подъемные краны, асфальтоукладчики, погрузчики и т.д.

Основные предметно-значимые сущности: Спецтехника, Заказчик, Заказ.

Основные предметно-значимые атрибуты сущностей:

- спецтехника – марка, модель, тип, стоимость;
- заказчик – ИНН заказчика, наименование, адрес, телефон;

- договор – номер договора, дата заключения договора, количество единиц спецтехники, дата заказа, отметка о выполнении.

Основные требования к функциям системы:

- выбрать все заказы определенного заказчика за указанный период;
- выбрать все заказы определенного типа спецтехники за указанный период;
- выбрать модели спецтехники, пользующиеся максимальным спросом;
- выбрать модели спецтехники, не пользующиеся спросом;
- выбрать клиентов с максимальным количеством заказов за определенный период;
- выбрать модель спецтехники с максимальной суммарной ценой реализации;
- выбрать все незавершенные договоры;
- выбрать все завершенные договоры.

Учебное издание

Попова-Коварцева Дарья Александровна,
Сопченко Елена Вильевна

ОСНОВЫ СОВРЕМЕННЫХ ТЕХНОЛОГИЙ БАЗ ДАННЫХ

Учебное пособие

Текст печатается в авторской редакции
Техническое редактирование Т.К. К р е т и н и н о й
Подготовка оригинал-макета Л.Р. Д м и т р и е н к о

Подписано в печать 27.11.2019. Формат 60x84 1/16.
Бумага офсетная. Печ. л. 5,75.
Тираж 120 экз. (1 з-д 1-30). Заказ .

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)
443086, Самара, Московское шоссе, 34.

Изд-во Самарского университета.
443086, Самара, Московское шоссе, 34.