

ДАЛЬНЕВОСТОЧНЫЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ТИХООКЕАНСКИЙ ИНСТИТУТ
ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ И ТЕХНОЛОГИЙ



КРАМОРЕНКО Н. В.

**БАЗЫ
ДАННЫХ**

ВЛАДИВОСТОК
2004

О Г Л А В Л Е Н И Е

ПРОГРАММА ДИСЦИПЛИНЫ	5
АННОТАЦИЯ	6
ВВЕДЕНИЕ	6
МОДУЛЬ 1. ОСНОВНЫЕ ПОНЯТИЯ	7
Глава 1.1. ВВЕДЕНИЕ В БАЗЫ ДАННЫХ	7
Глава 1.2. ПОЛЬЗОВАТЕЛИ БАНКОВ ДАННЫХ	9
1.2.1. Основные функции группы администратора БД	10
Глава 1.3. АРХИТЕКТУРА БАЗ ДАННЫХ	11
1.3.1. Трехуровневая архитектура баз данных	11
1.3.2. Процесс прохождения пользовательского запроса	12
Глава 1.4. КЛАССИФИКАЦИЯ МОДЕЛЕЙ ДАННЫХ	13
Глава 1.5. ЖИЗНЕННЫЙ ЦИКЛ БД	15
1.5.1. Системный анализ предметной области	17
МОДУЛЬ 2. ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ	17
Глава 2.1. ИНФОЛОГИЧЕСКОЕ (СЕМАНТИЧЕСКОЕ) МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	17
2.1.1. Модель «сущность-связь»	18
2.1.2. Пример построения модели «сущность-связь»	22
Глава 2.2. РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ	24
2.2.1. Реляционные объекты данных	25
2.2.2. Ограничения целостности в реляционной модели данных	28
2.2.3. Реляционная алгебра	31
2.2.4. Алгоритм перехода от модели «сущность-связь» к реляционной модели	40
Глава 2.3. ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ НА ОСНОВЕ ПРИНЦИПОВ НОРМАЛИЗАЦИИ	43
2.3.1. Функциональные зависимости	44
2.3.2. Первая нормальная форма	44
2.3.3. Вторая нормальная форма	45
2.3.4. Третья нормальная форма	45
2.3.5. Нормальная форма Бойса-Кодда	46
2.3.6. Четвертая нормальная форма	46
2.3.7. Пятая нормальная форма (нормальная форма проекции-соединения)	47
МОДУЛЬ 3. РЕАЛИЗАЦИЯ РЕЛЯЦИОННОЙ МОДЕЛИ В СРЕДЕ ВЫБРАННОЙ СУБД	47
Глава 3.1. РЕАЛИЗАЦИЯ РЕЛЯЦИОННОЙ МОДЕЛИ В СРЕДЕ ВЫБРАННОЙ СУБД (MS ACCESS)	47
3.1.1. Создание таблиц	47
3.1.2. Построение схемы данных. Задание ограничений целостности	51
Глава 3.2. ТАБЛИЧНЫЙ ЯЗЫК ЗАПРОСОВ QBE	53
3.2.1. Запросы с использованием одной таблицы	55
3.2.2. Возможности совместной обработки нескольких таблиц, связывание таблиц	58
3.2.3. Вычисляемые поля	61
3.2.4. Возможности группировки данных. Использование агрегатных функций	62
3.2.5. Вложенные запросы	64
3.2.6. Корректирующие запросы	65
3.2.7. QBE как «построитель» SQL-запросов	67
МОДУЛЬ 4. ЯЗЫК SQL	67
Глава 4.1. ОПЕРАТОР ВЫБОРА SELECT	68
4.1.1. Синтаксис оператора SELECT	68
4.1.2. Запросы с использованием одной таблицы	70
4.1.3. Возможности совместной обработки нескольких таблиц	71
4.1.4. Вычисляемые поля	72
Глава 4.2. ПРИМЕНЕНИЕ АГРЕГАТНЫХ ФУНКЦИЙ И ВЛОЖЕННЫХ ЗАПРОСОВ В ОПЕРАТОРЕ ВЫБОРА	73
4.2.1. SQL-функции	73
4.2.2. Вложенные подзапросы	74
Глава 4.3. ОПЕРАТОРЫ МАНИПУЛИРОВАНИЯ ДАННЫМИ	76

ЗАКЛЮЧЕНИЕ. НАПРАВЛЕНИЯ РАЗВИТИЯ БАЗ ДАННЫХ	77
ГЛОССАРИЙ.....	79
СПИСОК ЛИТЕРАТУРЫ	81
ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ	82
МОДЕЛЬ «СУЩНОСТЬ-СВЯЗЬ»	82
РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ	82
РЕЛЯЦИОННАЯ АЛГЕБРА	82
НОРМАЛИЗАЦИЯ ОТНОШЕНИЙ	83
ЗАПРОСЫ НА QBE	83
ЗАПРОСЫ НА SQL	84
ПРОЕКТИРОВАНИЕ БД «БИБЛИОТЕКА»	85

ПРОГРАММА ДИСЦИПЛИНЫ

Модуль 1. Основные понятия

Глава 1.1. Введение в базы данных

Глава 1.2. Пользователи баз данных

Основные функции группы администратора БД

Глава 1.3. Архитектура баз данных

Трехуровневая архитектура баз данных

Процесс прохождения пользовательского запроса

Глава 1.4. Классификация моделей данных.

Документальные модели. Фактографические модели.

Глава 1.5. Жизненный цикл БД.

Этапы проектирования БД. Системный анализ предметной области

Модуль 2. Проектирование базы данных

Глава 2.1 Инфологическое моделирование предметной области

Представление данных с помощью модели «сущность-связь» (ER-модели). Основные понятия: сущность, атрибут, ключ, связь. Виды связей.

Пример построения модели «сущность-связь»

Глава 2.1. Реляционная модель данных

Реляционные объекты данных

Ограничения целостности

Реляционная алгебра

Алгоритм перехода от модели «сущность-связь» к реляционной модели

Глава 2.3. Проектирование реляционных баз данных на основе принципов нормализации

Функциональные зависимости. 1НФ. 2НФ. 3НФ. НФБК. 4НФ. 5НФ

Модуль 3. Реализация реляционной модели в среде выбранной СУБД

Глава 3.1. Реализация реляционной модели в среде выбранной СУБД (MS Access)

Создание таблиц

Построение схемы данных. Задание ограничений целостности.

Глава 3.2. Табличный язык запросов (QBE)

Запросы с использованием одной таблицы

Возможности совместной обработки нескольких таблиц, связывание таблиц.

Вычисляемые поля.

Возможности группировки данных. Использование агрегатных функций.

Корректирующие запросы

QBE как «построитель» SQL-запросов

Модуль 4. Язык SQL

Глава 4.1. Оператор выбора Select

Синтаксис оператора Select

Запросы с использованием одной таблицы

Возможности совместной обработки нескольких таблиц

Вычисляемые поля

Глава 4.2. Применение агрегатных функций и вложенных запросов в операторе выбора

SQL-функции

Вложенные подзапросы

Глава 4.3. Операторы манипулирования данными

Заключение. Направления развития баз данных

Объектно-ориентированные БД

Распределенные БД

Темпоральные БД

Аннотация

В учебном пособии изложены принципы построения реляционных баз данных. Рассмотрен процесс построения концептуальных моделей, на примере модели «сущность-связь». Описываются все основные аспекты реляционной модели данных. Приведен пример реализации реляционной модели в СУБД MS Access.

Пособие предназначено для студентов специальности: 351400 «Прикладная информатика (по областям)» всех форм обучения.

Введение

В настоящее время большинство предприятий и организаций в той или иной мере используют в своей деятельности различные информационные системы (ИС). ИС могут быть связаны с различными областями деятельности предприятия, будь то бухгалтерия, управление персоналом или конкретный производственный процесс. В любом случае ИС имеют дело с огромными массивами информации, которые необходимо хранить, обновлять, корректировать, а также производить различные вычисления. Информация имеет достаточно сложную структуру и хранится в базах данных (БД). От эффективности управления БД непосредственно зависит эффективность работы ИС, а, следовательно, и самого предприятия, которое использует данную систему. Эффективность работы БД в большой степени зависит от грамотного проекта базы данных, построить который помогут основы теории баз данных, рассматриваемые в настоящем пособии.

Учебное пособие соответствует требованиям стандарта дисциплины «Базы данных» и предназначено для студентов специальности: 351400 «Прикладная информатика (по областям)» всех форм обучения.

В данном учебном пособии изложены принципы построения реляционных баз данных.

В модуле 1 содержится введение в понятия баз данных, в общих чертах представляются архитектура баз данных и жизненный цикл баз данных, дается классификация моделей данных, используемых на разных этапах жизненного цикла баз данных.

В модуле 2 обсуждаются вопросы проектирования реляционных баз данных. Подробно рассматривается этап инфологического моделирования предметной области на основе построения концептуальной модели «сущность-связь», рассматриваются основные аспекты реляционной модели данных, а также принципы нормализации отношений.

В модуле 3 дается описание процесса реализации реляционной модели в среде СУБД MS Access, на множестве примеров рассматривается табличный язык запросов QBE.

В модуле 4 рассматривается стандартный язык запросов SQL.

В заключении кратко характеризуются перспективы развития баз данных.

Модуль 1. Основные понятия

Глава 1.1. Введение в базы данных

Современные авторы часто употребляют термины «банк данных» и «база данных» как синонимы, однако в общеотраслевых руководящих материалах по созданию банков данных Государственного комитета по науке и технике (ГКНТ), изданных в 1982 г., эти понятия различаются. Там приводятся следующие определения банка данных, базы данных и СУБД:

Банк данных (БнД) — это система специальным образом организованных данных — баз данных, программных, технических, языковых, организационно-методических средств, предназначенных для обеспечения централизованного накопления и коллективного многоцелевого использования данных.

База данных (БД) — именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области.

Система управления базами данных (СУБД) — совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

Программы, с помощью которых пользователи работают с базой данных, называются *приложениями*. В общем случае с одной базой данных могут работать множество различных приложений. Например, если база данных моделирует некоторое предприятие, то для работы с ней может быть создано приложение, которое обслуживает подсистему учета кадров, другое приложение может быть посвящено работе подсистемы расчета заработной платы сотрудников, третье приложение работает как подсистема складского учета, четвертое приложение посвящено планированию производственного процесса. При рассмотрении приложений, работающих с одной базой данных, предполагается, что они могут работать параллельно независимо друг от друга, и именно СУБД призвана обеспечить работу множества приложений с единой базой данных таким образом, чтобы каждое из них выполнялось корректно, но учитывало все изменения в базе данных, вносимые другими приложениями.

Преимущества использования БД

Рассмотрим, какие преимущества получает пользователь при использовании БД как безбумажной технологии [1].

- компактность
информация хранится в БД, нет необходимости хранить многотомные бумажные картотеки
- скорость
скорость обработки информации (поиск, внесение изменений) компьютером намного выше ручной обработки
- низкие трудозатраты
нет необходимости в утомительной ручной работе над данными
- применимость
всегда доступна свежая информация

Дополнительные преимущества появляются при использовании БД в многопользовательской среде, поскольку становится возможным осуществлять централизованное управление данными. Рассмотрим подробнее преимущества, связанные с централизованным управлением:

- сокращение избыточности данных

В случае, когда для каждого приложения используется свой файл с данными, возможна ситуация, когда информация дублируется в разных файлах, т.е. является избыточной. Такая ситуация ведет к перерасходу памяти, а также к появлению противоречивой информации.

Например, приложение, связанное с учетом персонала, хранит информацию о сотрудниках. Подобную информацию может хранить и бухгалтерское приложение. Для сокращения избыточности, можно объединить общие данные в одном файле, к которому будут обращаться оба приложения (при условии, если администратор данных знает, какие данные нужны для каждого приложения).

Это не значит, что избыточность данных должна быть полностью устранена. Иногда приходится хранить резервные копии данных (например, для восстановления после сбоев). Такая избыточность должна строго контролироваться, т.е. должна быть предусмотрена возможность обновления таких копий.

- устранение противоречивости

Как было сказано выше, противоречивость появляется как следствие избыточности данных. Например, если информация о сотруднике хранится в нескольких файлах (БД, таблицах или записях), то может возникнуть ситуация, когда информация в одном месте будет обновлена, а в другом – нет. Т.е. информация станет противоречивой.

Если же факт представлен в одном экземпляре (т.е. при отсутствии избыточности), то противоречия возникнуть не могут.

Другой способ устранения противоречий заключается в контроле избыточности с помощью процесса каскадного обновления. В этом случае при внесении изменений (вставки, удалении или обновлении) в одном месте, оно должно автоматически распространяться на все записи.

- общий доступ к данным

Общий доступ к данным означает возможность доступа к данным со стороны нескольких приложений, как существующих, так и вновь создаваемых.

- возможность соблюдения стандартов

Благодаря централизованному управлению администратор БД может обеспечивать представление данных в определенных стандартах. Стандарты могут быть корпоративными, ведомственными, национальными, международными. Стандартизация важна для обмена данными, перенесения данных между системами, а также для совместного использования.

- возможность введения ограничений для обеспечения безопасности

Благодаря полному контролю над базой данных администратор БД может определить правила безопасности, которые будут проверяться при попытке доступа к уязвимым данным. Для разных типов доступа (выборки, вставки, удаления и т.д.) и разных частей БД можно определить разные правила доступа. Однако при отсутствии правил безопасность данных подвергается большему риску, чем в обычной (разрозненной) файловой системе, т.е. централизованная природа системы баз данных (СУБД) в некотором смысле *требует* наличия хорошей системы безопасности.

- обеспечение целостности данных

Задача целостности заключается в обеспечении правильности и точности данных в базе данных. Противоречие между двумя записями, представляющими один «факт», является примером недостатка целостности; конечно, эта проблема может возникнуть только при наличии избыточности в хранимых данных (см. пункт сокращение избыточности). Но даже если избыточность отсутствует, БД может содержать неправильную информацию. Например, год рождения сотрудника указан как 1999, тогда как сейчас 2004 год (возраст сотрудника – 5 лет?), или в домашнем адресе сотрудника указана несуществующая улица. Централизованное управление БД позволяет избежать подобных проблем – насколько их вообще можно избежать. Для этого определяются правила целостности, применяемые при каждой попытке обновления данных (т.е. операции обновления, вставки или удаления).

- обеспечение независимости данных

Приложения, реализованные на старых системах, в той или иной степени зависят от данных. В таких приложениях (называемых зависимыми от данных) невозможно изменить структуру хранения (т.е. способ физического хранения данных) или метод доступа (т.е. способ осуществления доступа к данным), не изменив самого приложения (возможно, радикально).

Современные системы управления базами данных обеспечивают как физическую (независимость от способа хранения и метода доступа), так и логическую независимость данных (возможность изменения одного приложения без изменения остальных приложений, работающих с этими же данными).

Глава 1.2. Пользователи банков данных

Как любой программно организационно-технический комплекс, банк данных существует во времени и в пространстве. Он имеет определенные стадии своего развития (жизненный цикл):

1. Проектирование
2. Реализация
3. Эксплуатация
4. Модернизация и развитие
5. Снятие с эксплуатации

На каждом этапе своего существования с банком данных связаны разные категории пользователей. Определим основные категории пользователей и их роль в функционировании банка данных.

Конечные пользователи

Это основная категория пользователей, в интересах которых и создается банк данных. В зависимости от особенностей создаваемого банка данных круг его конечных пользователей может существенно различаться. Это могут быть случайные пользователи, обращающиеся к БД время от времени за получением некоторой информации, а могут быть регулярные пользователи. В качестве случайных пользователей могут рассматриваться, например, возможные клиенты вашей фирмы, просматривающие каталог вашей продукции или услуг с обобщенным или подробным описанием того и другого. Регулярными пользователями могут быть ваши сотрудники, работающие со специально разработанными для них программами, которые обеспечивают автоматизацию их деятельности при выполнении своих должностных обязанностей. Например, менеджер, планирующий работу сервисного отдела компьютерной фирмы, имеет в своем распоряжении программу, которая помогает ему планировать и распределять текущие заказы, контролировать ход их выполнения, заказывать на складе необходимые комплектующие для новых заказов. Главный принцип состоит в том, что от конечных пользователей не должно требоваться каких-либо специальных знаний в области вычислительной техники и языковых средств.

Администраторы банка данных

Это группа пользователей, которая на начальной стадии разработки банка данных отвечает за его оптимальную организацию с точки зрения одновременной работы множества конечных пользователей, на стадии эксплуатации отвечает за корректность работы данного банка информации в многопользовательском режиме. На стадии развития и реорганизации эта группа пользователей отвечает за возможность корректной реорганизации банка без изменения или прекращения его текущей эксплуатации.

Разработчики и администраторы приложений

Это группа пользователей, которая функционирует во время проектирования, создания и реорганизации банка данных. Администраторы приложений координируют работу разработчиков при разработке конкретного приложения или группы приложений, объединенных в функциональную подсистему. Разработчики конкретных приложений работают с той частью информации из базы данных, которая требуется для конкретного приложения.

Не в каждом банке данных могут быть выделены все типы пользователей. При разработке информационных систем с использованием настольных СУБД администратор банка данных, администратор приложений и разработчик часто существовали в одном лице. Однако при построении современных сложных корпоративных баз данных, которые используются для автоматизации всех или большей части бизнес-процессов в крупной фирме или корпорации, могут существовать и группы администраторов приложений, и отделы разработчиков. Наиболее сложные обязанности возложены на группу администратора БД.

Рассмотрим их более подробно. В составе группы администратора БД должны быть:

- системные аналитики;
- проектировщики структур данных и внешнего по отношению к банку данных информационного обеспечения;
- проектировщики технологических процессов обработки данных;
- системные и прикладные программисты;
- операторы и специалисты по техническому обслуживанию.

Если речь идет о коммерческом банке данных, то важную роль здесь играют специалисты по маркетингу.

1.2.1. Основные функции группы администратора БД

1. Анализ предметной области: описание предметной области, выявление ограничений целостности, определение статуса (доступности, секретности) информации, определение потребностей пользователей, определение соответствия «данные—пользователь», определение объемно-временных характеристик обработки данных.

2. Проектирование структуры БД: определение состава и структуры файлов БД и связей между ними, выбор методов упорядочения данных и методов доступа к информации, описание БД на языке описания данных (ЯОД).

3. Задание ограничений целостности при описании структуры БД и процедур обработки БД:

- задание декларативных ограничений целостности, присущих предметной области;
- определение динамических ограничений целостности, присущих предметной области в процессе изменения информации, хранящейся в БД;
- определение ограничений целостности, вызванных структурой БД;
- разработка процедур обеспечения целостности БД при вводе и корректировке данных;
- определение ограничений целостности при параллельной работе пользователей в многопользовательском режиме.

4. Первоначальная загрузка и ведение БД:

- разработка технологии первоначальной загрузки БД, которая будет отличаться от процедуры модификации и дополнения данными при штатном использовании базы данных;
- разработка технологии проверки соответствия введенных данных реальному состоянию предметной области. База данных моделирует реальные объекты некоторой предметной области и взаимосвязи между ними, и на момент начала штатной эксплуатации эта модель должна полностью соответствовать состоянию объектов предметной области на данный момент времени;
- в соответствии с разработанной технологией первоначальной загрузки может понадобиться проектирование системы первоначального ввода данных.

5. Защита данных:

- определение системы паролей, принципов регистрации пользователей, создание групп пользователей, обладающих одинаковыми правами доступа к данным;
- разработка принципов защиты конкретных данных и объектов проектирования; разработка специализированных методов кодирования информации при ее циркуляции в локальной и глобальной информационных сетях;
- разработка средств фиксации доступа к данным и попыток нарушения системы защиты;
- тестирование системы защиты;
- исследование случаев нарушения системы защиты и развитие динамических методов защиты информации в БД.

6. Обеспечение восстановления БД:

- разработка организационных средств архивирования и принципов восстановления БД;
- разработка дополнительных программных средств и технологических процессов восстановления БД после сбоев.

7. Анализ обращений пользователей БД: сбор статистики по характеру запросов, по времени их выполнения, по требуемым выходным документам

8. Анализ эффективности функционирования БД:

- анализ показателей функционирования БД;
- планирование реструктуризации (изменение структуры) БД и реорганизации БД.

9. Работа с конечными пользователями:

- сбор информации об изменении предметной области;
- сбор информации об оценке работы БД;
- обучение пользователей, консультирование пользователей;
- разработка необходимой методической и учебной документации по работе конечных пользователей.

10. Подготовка и поддержание системных средств:

- анализ существующих на рынке программных средств и анализ возможности и необходимости их использования в рамках БД;
- разработка требуемых организационных и программно-технических мероприятий по развитию БД;
- проверка работоспособности закупаемых программных средств перед подключением их к БД;
- курирование подключения новых программных средств к БД.

11. Организационно-методическая работа по проектированию БД:

- выбор или создание методики проектирования БД;
- определение целей и направления развития системы в целом;
- планирование этапов развития БД;
- разработка общих словарей-справочников проекта БД и концептуальной модели;
- стыковка внешних моделей разрабатываемых приложений;
- курирование подключения нового приложения к действующей БД;
- обеспечение возможности комплексной отладки множества приложений, взаимодействующих с одной БД.

Глава 1.3. Архитектура баз данных

1.3.1. Трехуровневая архитектура баз данных

Терминология в СУБД, да и сами термины «база данных» и «банк данных» частично заимствованы из финансовой деятельности. Это заимствование – не случайно и объясняется тем, что работа с информацией и работа с денежными массами во многом схожи, поскольку и там и там отсутствует персонификация объекта обработки: две банкноты достоинством в сто рублей столь же неотличимы и взаимозаменяемы, как два одинаковых байта (естественно, за исключением серийных номеров). Вы можете положить деньги на некоторый счет и предоставить возможность вашим родственникам или коллегам использовать их для иных целей. Вы можете поручить банку оплачивать ваши расходы с вашего счета или получить их наличными в другом банке, и это будут уже другие денежные купюры, но их ценность будет эквивалентна той, которую вы имели, когда клали их на ваш счет.

В процессе научных исследований, посвященных тому, как именно должна быть устроена СУБД, предлагались различные способы реализации. Самым жизнеспособным из них оказалась предложенная американским комитетом по стандартизации ANSI (American National Standards Institute) трехуровневая система организации БД, изображенная на Рис. 1-1:

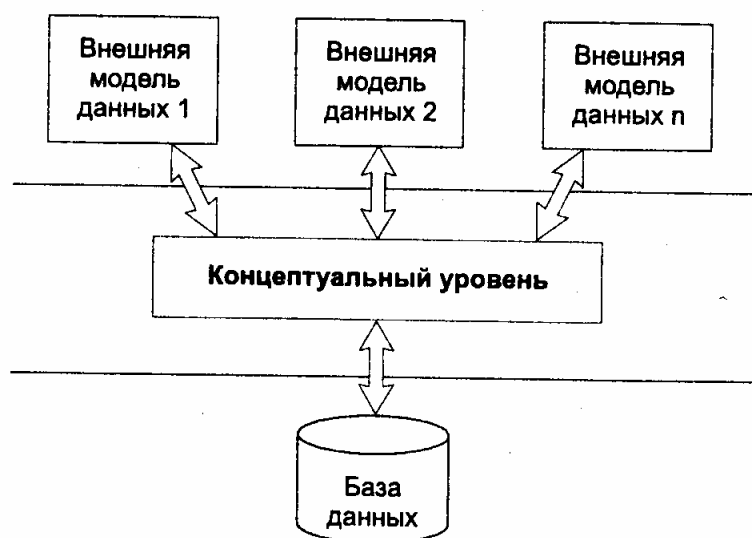


Рис. 1-1. Трехуровневая модель системы управления базой данных

1. **Уровень внешних моделей** – самый верхний уровень, где каждая модель имеет свое «видение» данных. Этот уровень определяет точку зрения на БД отдельных приложений. Каждое приложение видит и обрабатывает только те данные, которые необходимы именно этому приложению. Например, система распределения работ использует сведения о квалификации сотрудника, но ее не интересуют сведения об окладе, домашнем адресе и телефоне сотрудника, и наоборот, именно эти сведения используются в подсистеме отдела кадров.
2. **Концептуальный уровень** – центральное управляющее звено, здесь база данных представлена в наиболее общем виде, который объединяет данные, используемые всеми приложениями, работающими с данной базой данных. Фактически концептуальный уровень отражает обобщенную модель предметной области (объектов реального мира), для которой создавалась база данных. Как любая модель, концептуальная модель отражает только существенные, с точки зрения обработки, особенности объектов реального мира.
3. **Физический уровень** – собственно данные, расположенные в файлах или в страничных структурах, расположенных на внешних носителях информации.

Эта архитектура позволяет обеспечить логическую (между уровнями 1 и 2) и физическую (между уровнями 2 и 3) независимость при работе с данными.

Логическая независимость предполагает возможность изменения одного приложения без корректировки других приложений, работающих с этой же базой данных.

Физическая независимость предполагает возможность переноса хранимой информации с одних носителей на другие при сохранении работоспособности всех приложений, работающих с данной базой данных.

Выделение концептуального уровня позволило разработать аппарат централизованного управления базой данных.

1.3.2. Процесс прохождения пользовательского запроса

Рис. 1-2 иллюстрирует взаимодействие пользователя, СУБД и операционной системы (ОС) при обработке запроса на получение данных. Цифрами помечена последовательность взаимодействий:

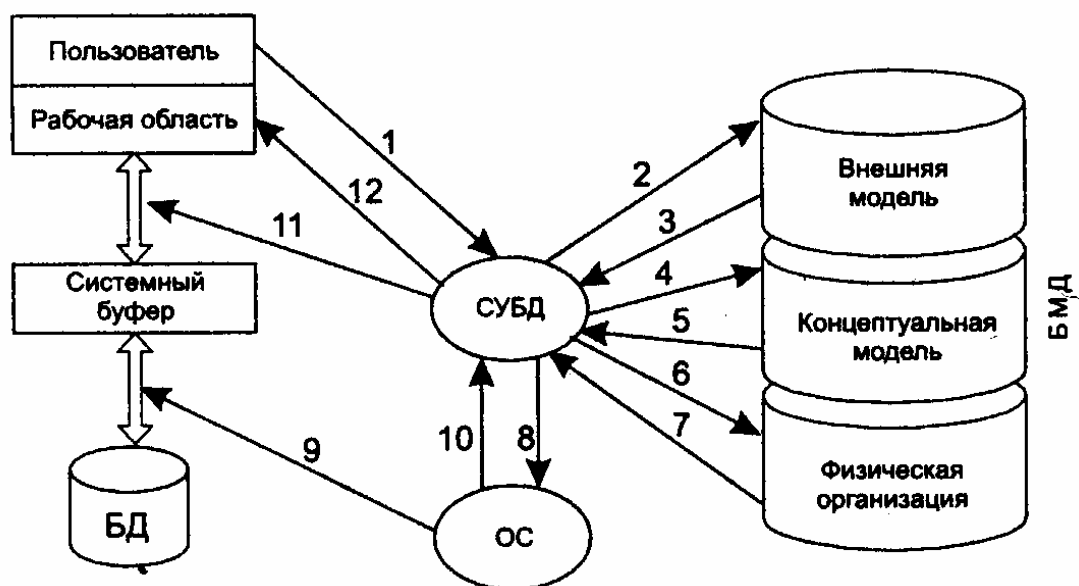


Рис. 1-2. Схема прохождения запроса к БД

1. Пользователь посылает СУБД запрос на получение данных из БД.
2. Анализ прав пользователя и внешней модели данных, соответствующей данному пользователю, подтверждает или запрещает доступ данного пользователя к запрошенным данным.
3. В случае запрета на доступ к данным СУБД сообщает пользователю об этом (стрелка 12) и прекращает дальнейший процесс обработки данных, в противном случае СУБД определяет часть концептуальной модели, которая затрагивается запросом пользователя (стрелка 4)

4. СУБД получает информацию о запрошенной части концептуальной модели.
5. СУБД запрашивает информацию о местоположении данных на физическом уровне (файлы или физические адреса).
6. В СУБД возвращается информация о местоположении данных в терминах операционной системы.
7. СУБД просит операционную систему предоставить необходимые данные, используя средства операционной системы.
8. Операционная система осуществляет перекачку информации из устройств хранения и пересылает ее в системный буфер.
9. Операционная система оповещает СУБД об окончании пересылки.
10. СУБД выбирает из доставленной информации, находящейся в системном буфере, только то, что нужно пользователю, и пересылает эти данные в рабочую область пользователя.

БМД — это *База Метаданных*, именно здесь и хранится вся информация об используемых структурах данных, логической организации данных, правах доступа пользователей и, наконец, физическом расположении данных. Для управления БМД существует специальное программное обеспечение администрирования баз данных, которое предназначено для корректного использования единого информационного пространства многими пользователями.

Всегда ли запрос проходит полный цикл? Конечно, нет. СУБД обладает достаточно развитым интеллектом, который позволяет ей не повторять бессмысленных действий. И поэтому, например, если этот же пользователь повторно обратится к СУБД с новым запросом, то для него уже не будут проверяться внешняя модель и права доступа, а если дальнейший анализ запроса покажет, что данные могут находиться в системном буфере, то СУБД осуществит только 11 и 12 шаги в обработке запроса.

Разумеется, механизм прохождения запроса в реальных СУБД гораздо сложнее, но и эта упрощенная схема показывает, насколько серьезными и сложными должны быть механизмы обработки запросов, поддерживаемые реальными СУБД.

Глава 1.4. Классификация моделей данных

Одними из основополагающих в концепции баз данных являются обобщенные категории «данные» и «модель данных».

Понятие «данные» в концепции баз данных – это набор конкретных значений, параметров, характеризующих объект, условие, ситуацию или любые другие факторы. Примеры данных: Петров Николай Степанович, \$30 и т. д. Данные не обладают определенной структурой, данные становятся информацией тогда, когда пользователь задает им определенную структуру, то есть осознает их смысловое содержание. Поэтому центральным понятием в области баз данных является понятие модели. Не существует однозначного определения этого термина, у разных авторов эта абстракция определяется с некоторыми различиями, но, тем не менее, можно выделить нечто общее в этих определениях.

Модель данных — это некоторая абстракция, которая, будучи приложима к конкретным данным, позволяет пользователям и разработчикам трактовать их уже как информацию, то есть сведения, содержащие не только данные, но и взаимосвязь между ними.

На Рис. 1-3 представлена классификация моделей данных.

В соответствии с рассмотренной ранее трехуровневой архитектурой мы сталкиваемся с понятием модели данных по отношению к каждому уровню. И действительно, физическая модель данных оперирует категориями, касающимися организации внешней памяти и структур хранения, используемых в данной операционной среде. В настоящий момент в качестве физических моделей используются различные методы размещения данных, основанные на файловых структурах: это организация файлов прямого и последовательного доступа, индексных файлов и инвертированных файлов, файлов, использующих различные методы хеширования, взаимосвязанных файлов. Кроме того, современные СУБД широко используют страничную организацию данных. Физические модели данных, основанные на страничной организации, являются наиболее перспективными.

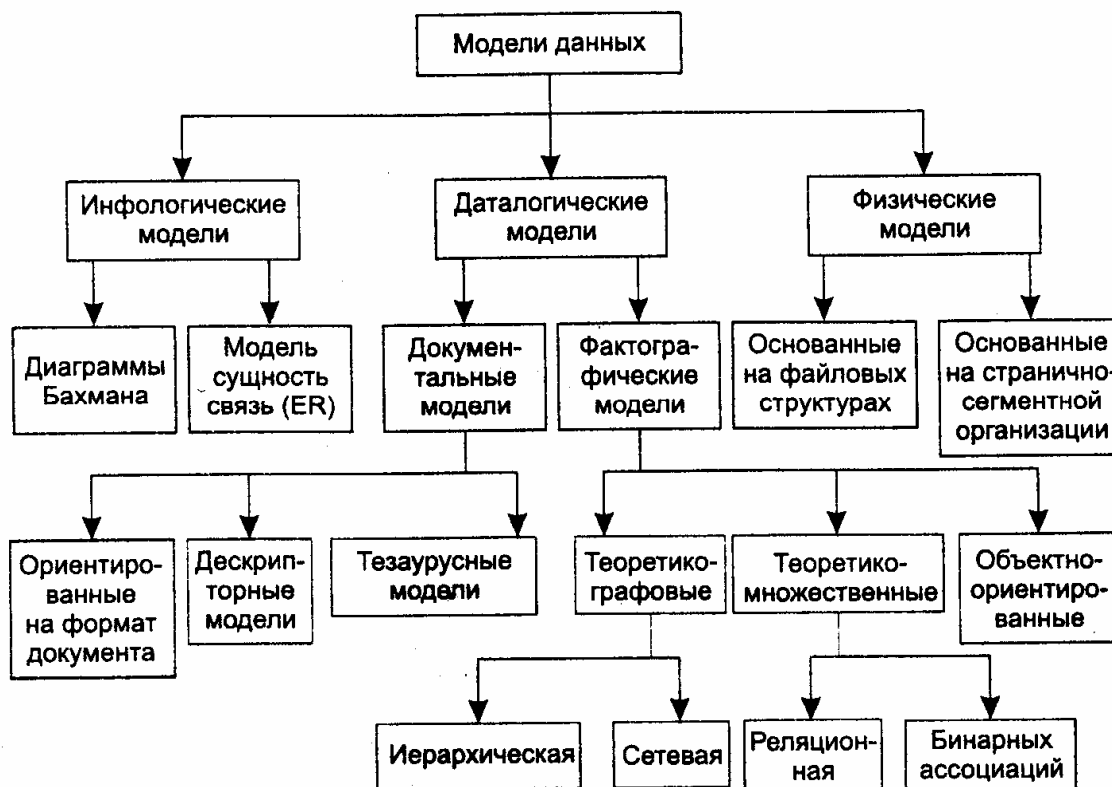


Рис. 1-3. Классификация моделей данных

Наибольший интерес вызывают модели данных, используемые на концептуальном уровне. По отношению к ним внешние модели называются подсхемами и используют те же абстрактные категории, что и концептуальные модели данных.

Модели концептуального уровня должны выражать информацию о предметной области в виде, независимом от используемой СУБД. Эти модели называются *инфологическими*, или *семантическими*, и отражают в естественной и удобной для разработчиков и других пользователей форме информационно-логический уровень абстрагирования, связанный с фиксацией и описанием объектов предметной области, их свойств и их взаимосвязей.

Инфологические модели данных используются на ранних стадиях проектирования для описания структур данных в процессе разработки приложения, а *даталогические* модели уже поддерживаются конкретной СУБД.

Фактографические модели данных соответствуют представлению информации в виде определенных структур данных (дерево, сеть, таблица).

Документальные модели данных соответствуют представлению о слабоструктурированной информации, ориентированной в основном на свободные форматы документов, текстов на естественном языке.

Модели, основанные на языках разметки документов, связаны, прежде всего, со стандартным общим языком разметки — SGML (Standart Generalised Markup

Language), который был утвержден ISO в качестве стандарта еще в 80-х годах. Этот язык предназначен для создания других языков разметки, он определяет допустимый набор тегов (ссылок), их атрибуты и внутреннюю структуру документа. Контроль за правильностью использования тегов осуществляется при помощи специального набора правил, называемых DTD-описаниями, которые используются программой клиента при разборе документа. Для каждого класса документов определяется свой набор правил, описывающих грамматику соответствующего языка разметки. С помощью SGML можно описывать структурированные данные, организовывать информацию, содержащуюся в документах, представлять эту информацию в некотором стандартизованном формате. Но ввиду некоторой своей сложности SGML использовался в основном для описания синтаксиса других языков (наиболее известным из которых является HTML), и немногие приложения работали с SGML-документами напрямую.

Гораздо более простой и удобный, чем SGML, язык HTML позволяет определять оформление элементов документа и имеет некий ограниченный набор инструкций — тегов, при помощи которых осуществляется процесс разметки. Инструкции HTML в первую очередь предназначены для управления процессом вывода содержимого документа на экране программы-клиента и определяют этим самым способ представления документа, но не его структуру. В качестве элемента гипертекстовой базы данных, описываемой HTML, используется текстовый файл, который может легко передаваться по сети с использованием протокола HTTP. Эта особенность, а также то, что HTML является открытым стандартом и огромное количество пользователей имеет возможность применять возможности этого языка для оформления своих документов, безусловно, повлияли на рост популярности HTML и сделали его сегодня главным механизмом представления информации в Интернете.

Однако HTML сегодня уже не удовлетворяет в полной мере требованиям, предъявляемым современными разработчиками к языкам подобного рода. И ему на смену был предложен новый язык гипертекстовой разметки, мощный, гибкий и, одновременно с этим, удобный язык XML. В чем же заключается его достоинства?

XML (Extensible Markup Language) — это язык разметки, описывающий целый класс объектов данных, называемых XML-документами. Он используется в качестве средства для описания грамматики других языков и контроля за правильностью составления документов. То есть сам по себе XML не содержит никаких тегов, предназначенных для разметки, он просто определяет порядок их создания.

Тезаурусные модели основаны на принципе организации словарей, содержат определенные языковые конструкции и принципы их взаимодействия в заданной грамматике. Эти модели эффективно используются в системах-переводчиках, особенно многоязыковых переводчиках. Принцип хранения информации в этих системах и подчиняется тезаурусным моделям,

Дескрипторные модели — самые простые из документальных моделей, они широко использовались на ранних стадиях использования документальных баз данных. В этих моделях каждому документу соответствовал дескриптор — описатель. Этот дескриптор имел жесткую структуру и описывал документ в соответствии с теми характеристиками, которые требуются для работы с документами в разрабатываемой документальной БД. Например, для БД, содержащей описание патентов, дескриптор содержал название области, к которой относился патент, номер патента, дату выдачи патента и еще ряд ключевых параметров, которые заполнялись для каждого патента. Обработка информации в таких базах данных велась исключительно по дескрипторам, то есть по тем параметрам, которые характеризовали патент, а не по самому тексту патента.

Глава 1.5. Жизненный цикл БД

Под **жизненным циклом** базы данных понимаются этапы развития БД, начиная от анализа предметной области, и заканчивая эксплуатацией БД.

Этапы жизненного цикла базы данных изображены на Рис. 0-4. Они аналогичны, в основном, развитию любой программной системы, однако в них есть определенная специфика, касающаяся только баз данных.

Процесс проектирования БД представляет собой последовательность переходов от неформального словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели. В общем случае можно выделить следующие этапы проектирования:

1. Системный анализ и словесное описание информационных объектов предметной области.
2. Проектирование инфологической модели предметной области — частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ER-модели.
3. Даталогическое или логическое проектирование БД, то есть описание БД в терминах принятой даталогической модели данных.
4. Физическое проектирование БД, то есть выбор эффективного размещения БД на внешних носителях для обеспечения наиболее эффективной работы приложения.

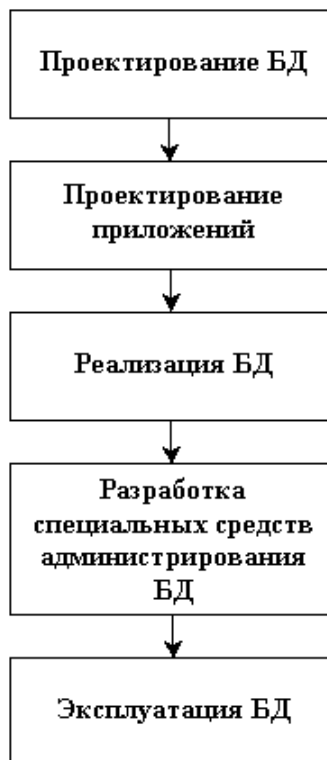


Рис. 0-1. Этапы жизненного цикла БД

Если мы учтем, что между вторым и третьим этапами необходимо принять решение, с использованием какой стандартной СУБД будет реализовываться наш проект, то условно процесс проектирования БД можно представить последовательностью выполнения пяти соответствующих этапов (Рис. 0-5). Рассмотрим более подробно этапы проектирования БД.



Рис. 0-2. Этапы проектирования БД

1.5.1. Системный анализ предметной области

С точки зрения проектирования БД в рамках системного анализа, необходимо осуществить первый этап, то есть провести подробное словесное описание объектов предметной области и реальных связей, которые присутствуют между описываемыми объектами. Желательно, чтобы данное описание позволяло корректно определить все взаимосвязи между объектами предметной области.

В общем случае существуют два подхода к выбору состава и структуры предметной области:

Функциональный подход – он реализует принцип движения «от задач» и применяется тогда, когда заранее известны функции некоторой группы лиц и комплексов задач, для обслуживания информационных потребностей которых создается рассматриваемая БД. В этом случае мы можем четко выделить минимальный необходимый набор объектов предметной области, которые должны быть описаны.

Предметный подход – когда информационные потребности будущих пользователей БД жестко не фиксируются. Они могут быть многоаспектными и весьма динамичными. Мы не можем точно выделить минимальный набор объектов предметной области, которые необходимо описывать. В описание предметной области в этом случае включаются такие объекты и взаимосвязи, которые наиболее характерны и наиболее существенны для нее. БД, конструируемая при этом, называется предметной, то есть она может быть использована при решении множества разнообразных, заранее не определенных задач. Конструирование предметной БД в некотором смысле кажется гораздо более заманчивым, однако трудность всеобщего охвата предметной области с невозможностью конкретизации потребностей пользователей может привести к избыточно сложной схеме БД, которая для конкретных задач будет неэффективной.

Чаще всего на практике рекомендуется использовать некоторый компромиссный вариант, который, с одной стороны, ориентирован на конкретные задачи или функциональные потребности пользователей, а с другой стороны, учитывает возможность наращивания новых приложений.

Системный анализ должен заканчиваться подробным описанием информации об объектах предметной области, которая требуется для решения конкретных задач и которая должна храниться в БД, формулировкой конкретных задач, которые будут решаться с использованием данной БД с кратким описанием алгоритмов их решения, описанием выходных документов, которые должны генерироваться в системе, описанием входных документов, которые служат основанием для заполнения данными БД.

Модуль 2. Проектирование баз данных

Глава 2.1. Инфологическое (семантическое) моделирование предметной области

Инфологическое моделирование (иногда используется термин семантическое моделирование) применяется на втором этапе проектирования БД, то есть после **системного анализа предметной области**. На этапе системного анализа были сформированы понятия о предметах, фактах и событиях, которыми будет оперировать БД. Инфологическое проектирование связано с представлением семантики предметной области в модели БД, т.е. моделирование структур данных, опираясь на смысл этих данных. Инфологическое моделирование было предметом исследований в конце 1970-х и начале 1980-х годов. Было предложено несколько моделей данных, названных семантическими моделями. Наибольшее распространение получила модель "сущность-связь" (entity-relationship model, ER-модель), предложенная в 1976 г. Питером Пин-Шэн Ченом.

Модель «сущность-связь» является **концептуальной моделью**, т.е. не учитывает особенности конкретной СУБД. Из модели "сущность-связь" могут быть получены все основные **фактографические модели данных** (иерархическая, сетевая, реляционная, объектно-ориентированная).

Модели "сущность-связь" удобны тем, что процесс создания модели является итерационным. Разработав первый приближенный вариант модели, можно уточнять ее, опрашивая экспертов предметной области. При этом документацией, в которой фиксируются результаты бесед, является сама модель "сущность-связь".

В этой главе мы рассмотрим основные понятия модели "сущность-связь":

- Сущности
 - Атрибуты
 - Ключевые атрибуты
 - Связи
4. степени связей
 5. классы принадлежности сущностей в связях

А также рассмотрим пример построения диаграммы "сущность-связь".

2.1.1. Модель «сущность-связь»

Основными понятиями модели "сущность-связь" являются: сущность, связь и атрибут.

Любой фрагмент предметной области может быть представлен как *множество сущностей*, между которыми существует некоторое *множество связей*.

Сущность - это реальный или представляемый объект, информация о котором должна сохраняться в проектируемой системе.

Сущность имеет имя, уникальное в пределах системы. Сущность соответствует некоторому классу однотипных объектов, то есть в системе существует множество экземпляров данной сущности.

Примеры сущностей: люди, продукты, студенты и т.д.

Примеры экземпляров сущностей: конкретный человек, конкретный продукт, конкретный студент и т.д.

Сущности не обязательно должны быть непересекающимися. Например, экземпляр сущности СТУДЕНТ, также принадлежит сущности ЛЮДИ.

Объект, которому соответствует понятие сущности, имеет свой набор **атрибутов** – характеристик, определяющих свойства данного объекта. Атрибут должен иметь имя, уникальное в пределах данной сущности.

Пример:

Рассмотрим множество пищевых продуктов, имеющихся в магазине. Каждый продукт можно представить следующими характеристиками: Код продукта, Продукт, , Срок хранения, Условия хранения

В дальнейшем для определения сущности и ее атрибутов будем использовать обозначение вида

Продукты(КодПродукта, Продукт, ЕдиницаИзмерения, СрокХранения, УсловияХранения)

Например, поставщиков, поставляющих продукты в магазин, можно описать как
Поставщики(КодПоставщика, Поставщик, Адрес)

Множество допустимых значений (область определения) атрибута называется **доменом**.

Например, атрибут *СрокХранения* хранит информацию о количестве дней, в течение которых продукт годен к употреблению. То есть этот атрибут принадлежит домену *КоличествоДней*, который задается интервалом целых чисел больших нуля, поскольку количество дней отрицательным быть не может.

Набор атрибутов сущности должен быть таким, чтобы можно было однозначно найти требуемый экземпляр сущности.

Например, сущность *Продукты* однозначно определяется атрибутом *КодПродукта*, поскольку все коды продуктов различны.

Отсюда определяется **ключ сущности** - это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся.

Пример. Сущность *Продажа* с атрибутами *ДатаПродажи*, *КодПродукта*, *Количество*, *Цена* содержит информацию о продаже продуктов за конкретный день. Для этой сущности ключом будут атрибуты *ДатаПродажи* и *КодПродукта*, поскольку за день могут быть проданы несколько продуктов, а конкретный продукт может быть продан в разные дни. Исключение любого атрибута из

ключа не позволит однозначно найти требуемый экземпляр сущности, т.е. будет нарушено условие минимальности. Обозначим эту сущность как

Продажа(ДатаПродажи, КодПродукта, Количество, Цена)

Ключевой атрибут выделен подчеркиванием

Между сущностями могут быть установлены связи.

Связь - это ассоциация, установленная между несколькими сущностями, и показывающая как взаимодействуют сущности между собой.

Примеры:

- в магазине происходит продажа продуктов, т.е. между сущностями *Продукты* и *Продажа* существует связь «*происходит*» или *Продукты-Продажа* (обычно, но необязательно, связь обозначается глаголом или двойным названием сущностей, между которыми установлена эта связь)
- так как продукты в магазин поставляют поставщики, то между сущностями *Продукты* и *Поставщики* существует связь «*поставка*» или *Продукты-Поставщики*
- могут существовать и связи между экземплярами одной и той же сущности (рекурсивная связь), например связь *Родитель-Потомок* между экземплярами сущности *Человек*

Связь также может иметь атрибуты. Например, для связи *Продукты-Поставщики* можно задать атрибуты *ДатаПоставки*, *Цена* и т.д.

Связь, существующая между двумя сущностями, называется **бинарной связью**.

Связь, существующая между *n* сущностями, называется ***n*-арной связью**.

Рекурсивная связь – это связь между экземплярами одной сущности.

Доказано, что любую *n*-арную связь всегда можно заменить множеством бинарных, однако первые лучше отображают семантику предметной области.

То число экземпляров сущностей, которое может быть ассоциировано через связь с экземплярами другой сущности, называют **степенью связи**. Рассмотрение степеней особенно полезно для бинарных связей.

Очень важным свойством модели "сущность-связь" является то, что она может быть представлена в виде графической схемы (диаграммы). Это значительно облегчает анализ предметной области. Существует несколько вариантов обозначения элементов диаграммы "сущность-связь" (нотаций). Для обозначения сущностей, связей и атрибутов будем использовать нотацию Чена, а для обозначения степеней и кардинальностей связей – нотацию Мартина (понятие кардинальности связи поясним чуть позже). В Таблице 2-1 приводится список используемых здесь обозначений.

Таблица 2-1

Обозначение	Пояснение
	Независимая сущность
	Зависимая сущность
	Атрибут
	Ключевой атрибут
	Связь
	Связь степени 1, необязательный класс принадлежности
	Связь степени 1, обязательный класс принадлежности

	Связь степени N, необязательный класс принадлежности
	Связь степени N, обязательный класс принадлежности
	Связь от зависимой к независимой сущности

Рассмотрим теперь существующие степени бинарных связей:

- **один-к-одному** (обозначается 1:1). Это означает, что в такой связи в каждый момент времени каждому экземпляру сущности А соответствует 1 или 0 экземпляров сущности В. Данный факт представлен на Рис. 2-1, где прямоугольники обозначают сущности, а ромб - связь. Так как степень связи для каждой сущности равна 1, то они соединяются одной линией.



Рис. 2-1. Связь один-к-одному

- **один-ко-многим** (1:N). Одному экземпляру сущности А соответствуют 0, 1 или N экземпляров сущности В. Графически степень связи N отображается "древовидной" линией, так это сделано на следующем рисунке (Рис. 2-2).

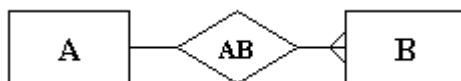


Рис. 2-2. Связь один-ко-многим

- **многие-к-одному** (N:1). Эта связь аналогична отображению 1:N. Одному экземпляру сущности В соответствуют 0, 1 или N экземпляров сущности А (Рис. 2-3).



Рис. 2-3. Связь многие-к-одному

- **многие-ко-многим** (M:N). В этом случае одному экземпляру сущности А соответствуют 0, 1 или N экземпляров сущности В, и наоборот, одному экземпляру сущности В соответствуют 0, 1 или N экземпляров сущности А (Рис. 2-4).

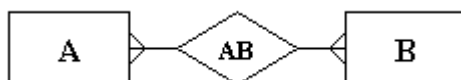


Рис. 2-4. Связь многие-ко-многим

Другой важной характеристикой связи помимо ее степени является **класс принадлежности** входящих в нее сущностей. Существует **обязательный** и **необязательный** классы принадлежности.

Рассмотрим примеры степени связи один-к-одному:

1. Если рассматривать **традиционный брак**, то степень связи между сущностями *Мужчина* и *Женщина* будет 1:1. Кроме того, каждый мужчина, состоящий в браке, обязательно ДОЛЖЕН иметь жену. Таким образом, говорят, что сущность *Женщина* имеет **обязательный класс принадлежности**. И, наоборот, каждая женщина, состоящая в браке, ДОЛЖНА иметь мужа. То есть, сущность *Мужчина* также имеет **обязательный класс принадлежности** (2-5).



Рис. 2-5. Связь 1:1, обязательные классы принадлежности

2. **Сотрудник руководит отделом** (Рис. 2-6). Поскольку сотрудник может руководить только ОДНИМ отделом, а в отделе может быть только ОДИН руководитель, то степень связи в этом

примере 1:1. Кроме того, в каждом отделе ДОЛЖЕН быть руководитель, т.е. каждому экземпляру сущности *Отдел* ДОЛЖЕН соответствовать экземпляр сущности *Сотрудник* (сущность *Сотрудник* имеет обязательный класс принадлежности). С другой стороны, далеко не все сотрудники должны быть руководителями, т.е. сотрудник МОЖЕТ (но не должен) быть руководителем. Таким образом, есть экземпляры сущности *Сотрудник*, которым не соответствует ни один экземпляр сущности *Отдел* (необязательный класс принадлежности).



Рис. 2-6. Связь 1:1, обязательный и необязательный классы принадлежности

3. **Человек читает книгу** (Рис. 2-7). Человек может читать сразу только ОДНУ книгу, а конкретная книга может быть читаема только ОДНИМ человеком, следовательно, степень связи 1:1. Человек МОЖЕТ читать книгу, а может ничего не читать. С другой стороны не каждая книга должна читаться, некоторые стоят на полке. Таким образом, обе сущности имеют необязательный класс принадлежности.



Рис. 2-7. Связь 1:1, необязательные классы принадлежности

Рассмотрим примеры степени связи один-ко-многим (многие-к-одному):

4. В процессе обучения студенты объединены в группы. Каждая группа может содержать множество студентов, а каждый студент может входить только в одну группу, т.е. степень связи 1:N (Рис. 2-8). Каждая группа ДОЛЖНА содержать студентов, а каждый студент ДОЛЖЕН быть зачислен в конкретную группу, т.е. обе сущности имеют обязательные классы принадлежности

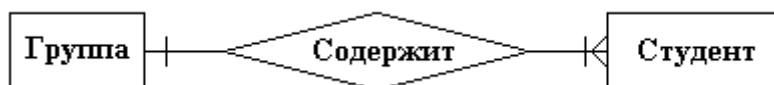


Рис. 2-8. Связь 1:N, обязательные классы принадлежности

5. Поставщики продуктов имеют один юридический адрес, следовательно, ДОЛЖНЫ находиться в одном конкретном городе. А в одном городе МОГУТ находиться один, несколько или ни одного поставщика. Т.е. связь будет N:1, сущность Города будет иметь обязательный, а сущность Поставщики – необязательный классы принадлежности (Рис. 2-9).

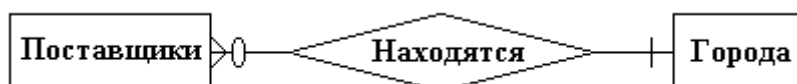


Рис. 2-9. Связь N:1, обязательный и необязательный классы принадлежности

Если существование сущности *x* зависит от существования сущности *y*, то *x* называется **зависимой сущностью** (иногда сущность *x* называют "слабой", а "сущность" *y* - *сильной*).

Степень связи для сильной сущности всегда будет 1 и обязательный класс принадлежности. Класс принадлежности и степень связи для зависимой сущности могут быть любыми.

6. В магазине происходит продажа продуктов. Продукт не может быть продан, если его нет в магазине. Поэтому сущность *Продажи* является зависимой от сущности *Продукты* (Рис. 2-10). Продукт МОЖЕТ быть продан в разные дни (а может быть вообще не продан), конкретная продажа связана только с одним продуктом. Таким образом, степень связи N:1, сущность *Продажи* имеет необязательный, а сущность *Продукты* - обязательный классы принадлежности (в самом деле, продажа без продукта теряет смысл)

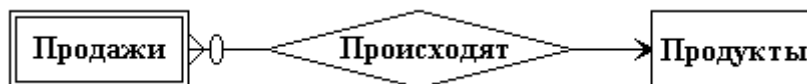


Рис. 2-10. Зависимая и независимая сущности

Рассмотрим пример степени связи многие-ко-многим:

7. Продукты в магазин поставляются поставщиками. Каждый продукт, имеющийся в магазине, ДОЛЖЕН быть поставлен одним или несколькими поставщиками, а каждый из поставщиков МОЖЕТ поставлять один или несколько продуктов или не поставлять ни одного. Т.е. степень связи M:N (Рис. 2-11), а класс принадлежности для *Поставщиков* – обязательный, для *Продуктов* – необязательный.



Рис. 2-11. Связь M:N, обязательный и необязательный классы принадлежности

Между одними и теми же сущностями могут существовать несколько связей, например:

8. С одной стороны продукты в магазин поставляются заказчиками, с другой стороны, чтобы продукты были поставлены в магазин, необходимо заказать поставщикам необходимые продукты. Таким образом, между сущностями *Продукты* и *Поставщики* существуют связи «*Поставляют*» и «*Заказаны*» (Рис. 2-12). Связь «*Поставляют*» рассмотрена в предыдущем примере. Рассмотрим подробнее связь «*Заказаны*». Каждый продукт ДОЛЖЕН быть заказан одному или нескольким поставщикам, каждый поставщик МОЖЕТ получить заказ на один или несколько продуктов или вообще не получить заказ.

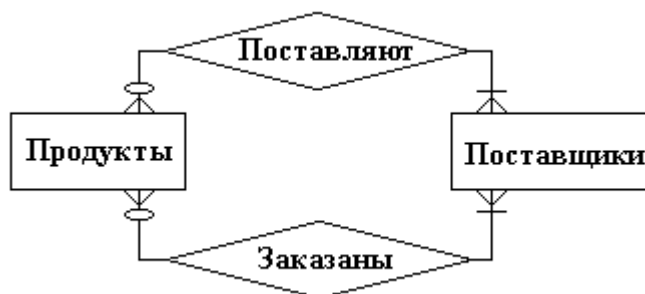


Рис. 2-12. Несколько связей между двумя сущностями

9. Рассмотрим сущности *Врач* и *Пациент*. Пациент ДОЖЕН иметь одного лечащего врача, а врач МОЖЕТ лечить нескольких пациентов. Кроме того, пациент МОЖЕТ иметь нескольких врачей-консультантов, а врач МОЖЕТ консультировать нескольких пациентов (Рис.2-13).



Рис. 2-13. Несколько связей между двумя сущностями

2.1.2. Пример построения модели «сущность-связь»

В процессе построения диаграммы "сущность-связь" можно выделить несколько этапов:

- Определение списка сущностей выбранной предметной области
- Определение списка атрибутов сущностей
- Описание связей между сущностями (степени, классы принадлежности связей, а также атрибуты связей, если они необходимы)
- Организация данных в виде диаграммы "сущность-связь"

В качестве примера построим диаграмму, отображающую связь данных для информационной системы учета продажи продуктов в магазине. БД должна хранить информацию о продуктах, поставляемых в магазин, их ежедневной продаже, заказах на поставку продуктов, а также о поставщиках продуктов.

Составим список сущностей, необходимых для реализации поставленной задачи:

1. Продукты

Для этой сущности необходимы следующие атрибуты:

- *Код продукта* – уникальный идентификатор, ключевой атрибут

- *Продукт* – название продукта
- *Единица измерения* – литры, килограммы, штуки и т.п.
- *Срок хранения в днях* – для определения даты окончания срока годности продукта
- *Условия хранения* – температура, влажность и т.п.

2. Поставщики

- *Код поставщика* – уникальный идентификатор, ключевой атрибут
- *Поставщик* – название организации или ФИО физического лица
- *Код города* – выделим отдельно город, где находится поставщик, для удобства дальнейшей работы (например, для поиска)
- *Адрес* – поскольку город выделен в отдельный атрибут, то в адресе остается улица и дом (а также квартира – для физического лица)
- ФИО директора
- Телефон
- Факс

3. Продажи

- Дата продажи
- Код продукта – какой именно продукт был продан
- Количество – сколько продано этого продукта в тех единицах измерения, которые указаны для этого продукта в сущности Продукт
- Цена продажи – цена при продаже за единицу продукта

4. Города – поскольку мы выделили отдельно город из адреса поставщика, то возникает необходимость в этой сущности

Код города – уникальный идентификатор, ключевой атрибут
Город

Сократив для удобства названия атрибутов, получим список сущностей:

- Продукты(КодПрод, Продукт, ЕдИзм, СрокХран(дней), УсловияХран)
- Поставщики(КодПост, Поставщик, КодГорода, Адрес, ФИОДиректора, Телефон, Факс)
- Продажи(ДатаПродажи, КодПрод, Количество, ЦенаПродажи)
 обратите внимание, что в этой сущности ключ составной, поскольку каждый день продается множество продуктов, и конкретный продукт может быть продан в разные дни
- Города(КодГорода, Город)

Рассмотрим связи, существующие между описанными выше сущностями:

1. Продукты в магазин поставляются поставщиками, т.е. существует связь М:Н «Поставляют» между сущностями Продукты и Поставщики (подробно эта связь рассмотрена в примере 7 параграфа 2.1.1., Рис. 2-11). Эта связь имеет следующие атрибуты:

- Дата поставки
- Код поставщика – какой поставщик поставил этот продукт
- Код продукта – какой именно продукт был поставлен
- КоличествоП – сколько поставлено этого продукта в тех единицах измерения, которые указаны для этого продукта в сущности Продукт
- Цена поставки – цена при поставке за единицу продукта
- Дата изготовления – дата изготовления продукта

Ключом будет составной атрибут: *Дата поставки, Код поставщика, Код продукта*
(объясните, почему именно эти атрибуты вошли в составной ключ)

2. Продукты должны быть заказаны поставщикам, т.е. существует связь М:Н «Заказаны» между сущностями Продукты и Поставщики (подробно эта связь рассмотрена в примере 8 параграфа 2.1.1, Рис. 2-12). Эта связь имеет следующие атрибуты:

- Дата заказа
- Код поставщика – какому поставщику заказан этот продукт
- Код продукта – какой именно продукт был заказан
- КоличествоЗ – сколько поставлено этого продукта в тех единицах измерения, которые указаны для этого продукта в сущности Продукт

Ключом будет составной атрибут: *Дата заказа, Код поставщика, Код продукта* (объясните, почему именно эти атрибуты вошли в составной ключ)

3. В магазине происходит продажа продуктов, т.е. существует связь N:1 «**Происходит**» между сущностями Продажи и Продукты (подробно эта связь рассмотрена в примере 6 параграфа 2.1.1, Рис. 2-10)
4. Поставщики находятся в определенном городе, т.е. существует связь N:1 «**Находятся**» между сущностями Поставщики и Города (подробно эта связь рассмотрена в примере 5 параграфа 2.1.1., Рис. 2-9)

После объединения всех фрагментов в общую модель и добавления атрибутов, получится диаграмма "сущность-связь", приведенная на Рис. 2-14.



Рис. 2-14. Диаграмма «сущность-связь» учета продажи продуктов в магазине

Глава 2.2. Реляционная модель данных

В этой главе мы рассмотрим следующие вопросы:

- Первую часть реляционной модели – объекты (структура)
 - отношения
 - домены
 - кортежи
 - атрибуты
 - свойства отношений
- Вторую часть реляционной модели – целостность
 - структурная целостность
 - языковая целостность
 - ссылочная целостность
- Третью часть реляционной модели – операторы реляционной алгебры
 - теоретико-множественные операции
 - специальные операции реляционной алгебры
 - свойство замкнутости
 - совместимость по типу
 - свойства ассоциативности и коммутативности
 - примитивные операции
 - примеры использования реляционных операций

2.2.1. Реляционные объекты данных

Основные понятия и ограничения реляционной модели (от английского relation – отношение) впервые были сформулированы сотрудником компании IBM Е.Ф.Коддом в 1970 г.

Реляционная модель связана с тремя аспектами данных: объектами данных (структурой данных), целостностью данных и обработкой данных [1, 2].

Основной структурной частью (объектом) реляционной модели является отношение.

Основные понятия

Рассмотрим наиболее важные термины, используемые в структурной части реляционной модели.

Декартово произведение Для заданных конечных множеств D_1, D_2, \dots, D_n (не обязательно различных) декартовым произведением $D_1 \times D_2 \times \dots \times D_n$ называется множество произведений вида: $d_1 \times d_2 \times \dots \times d_n$, где $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$.

Пример: Имеем три домена $D_1 = \{a, b, c\}, D_2 = \{m, k\}, D_3 = \{y, z\}$.

Декартово произведение этих доменов

$D = D_1 \times D_2 \times D_3 = (a \times m \times y, a \times m \times z, a \times k \times y, a \times k \times z,$

$b \times m \times y, b \times m \times z, b \times k \times y, b \times k \times z,$

$c \times m \times y, c \times m \times z, c \times k \times y, c \times k \times z)$

Отношением R , определенным на множествах D_1, D_2, \dots, D_n ($n \geq 1$), необязательно различных, называется подмножество декартова произведения $D_1 \times D_2 \times \dots \times D_n$.

Исходные множества D_1, D_2, \dots, D_n называются **доменами** отношения

Элементы декартова произведения $d_1 \times d_2 \times \dots \times d_n$ называются **кортежами**

Число n определяет **степень отношения** ($n=1$ - унарное, $n=2$ - бинарное, ..., n -арное)

Количество кортежей называется **кардинальным числом** или **мощностью отношения**

Домен представляет собой именованное множество атомарных значений одного типа. Под атомарным значением понимается “наименьшая семантическая единица данных”, т.е. это значение, не имеющее внутренней структуры при рассмотрении в реляционной модели. Это не значит, что такое значение не имеет внутренней структуры вообще. Например, название должности состоит из букв, но, разложив название по буквам, мы потеряем значение.

Домены являются общими совокупностями значений, из которых берутся конкретные значения атрибутов. Т.е. каждый *атрибут должен быть определен на основе одного домена*; это значит, что значения атрибута должны браться из этого домена.

Значение доменов заключается в том, что **домены ограничивают сравнения**. Т.е. если два атрибута определены на одном и том же домене, то их можно сравнивать, применяя операции сравнения допустимые для данного домена. Например, атрибуты *Дата приема на работу* и *Дата окончания ВУЗа* определены на одном домене *Даты*; для этого домена допустимы операции сравнения: $=, \neq, <, \leq, >, \geq$. Поэтому данные атрибуты можно сравнивать, используя все указанные операции сравнения.

Отношение удобно представить в виде таблицы, столбцы которой соответствуют вхождениям доменов в отношение, а строки – наборам из n значений, взятых из исходных доменов, и расположенным в соответствии с заголовком отношения (Рис. 2-15). Столбцы отношения называются **атрибутами**, а строки – **кортежами**. Однако нельзя сказать, что отношение и таблица полностью идентичны. Различие между отношением и таблицей мы рассмотрим чуть позже, когда будем рассматривать свойства отношений.

Отношение содержит две части: *заголовок* и *тело* (заголовок – это строка заголовков столбцов, тело – это множество строк данных).

Заголовок (или **схема отношения**) содержит фиксированное множество атрибутов или, точнее, пар <имя-атрибута : имя-домена>:

$\{ \langle A1:D1 \rangle, \langle A2:D2 \rangle, \dots, \langle An:Dn \rangle \},$

причем каждый атрибут A_j соответствует только одному из лежащих в основе доменов D_j ($j = 1, 2, \dots, n$). Все имена атрибутов $A1, A2, \dots, An$ разные.

Схемы двух отношений называются *эквивалентными*, если они имеют одинаковую степень и возможно такое упорядочение имен атрибутов в схемах, что на одинаковых местах будут находиться сравнимые атрибуты, т.е. атрибуты, принимающие значения из одного домена.

Схема БД (в структурном смысле) - это набор именованных схем отношений. Тогда **реляционная БД** – это набор отношений, имена которых совпадают с именами схем отношений в схеме БД.

Тело содержит множество кортежей.

Каждый **кортеж**, в свою очередь, содержит множество пар <имя-атрибута : значение-атрибута>:

{<A1:vi1>, <A2:vi2>, ..., <An:vin>},

($i = 1, 2, \dots, m$, где m – количество кортежей в этом множестве). В каждом таком кортеже есть одна такая пара <имя-атрибута : значение-атрибута>, т.е. <A_j:vi_j>, для каждого атрибута A_j в заголовке. Для любой такой пары <A_j:vi_j> vi_j является значением из уникального домена D_j, связанного с атрибутом A_j.

Т.е. можно сказать, что *отношение* – это множество кортежей, соответствующих одной схеме отношения.

Атрибут, значение которого однозначно идентифицирует кортежи, называется **ключевым** (или просто **ключом**). Если кортежи идентифицируются только сцеплением значений нескольких атрибутов, то говорят, что отношение имеет составной ключ.

Отношение может содержать несколько ключей. Всегда один из ключей объявляется **первичным**, его значения не могут обновляться. Все остальные ключи отношения называются *возможными (потенциальными или альтернативными) ключами*.

Пример: Для иллюстрации введенных терминов рассмотрим отношение *Расписание*, приведенное на Рис. 2-15. В этом отношении есть четыре основных **домена**: домен номеров рейса (*№ рейса*), домен наименований населенных пунктов (*Населенные пункты*), домен времени (*Время*) и домен типов поездов (*Тип поезда*).

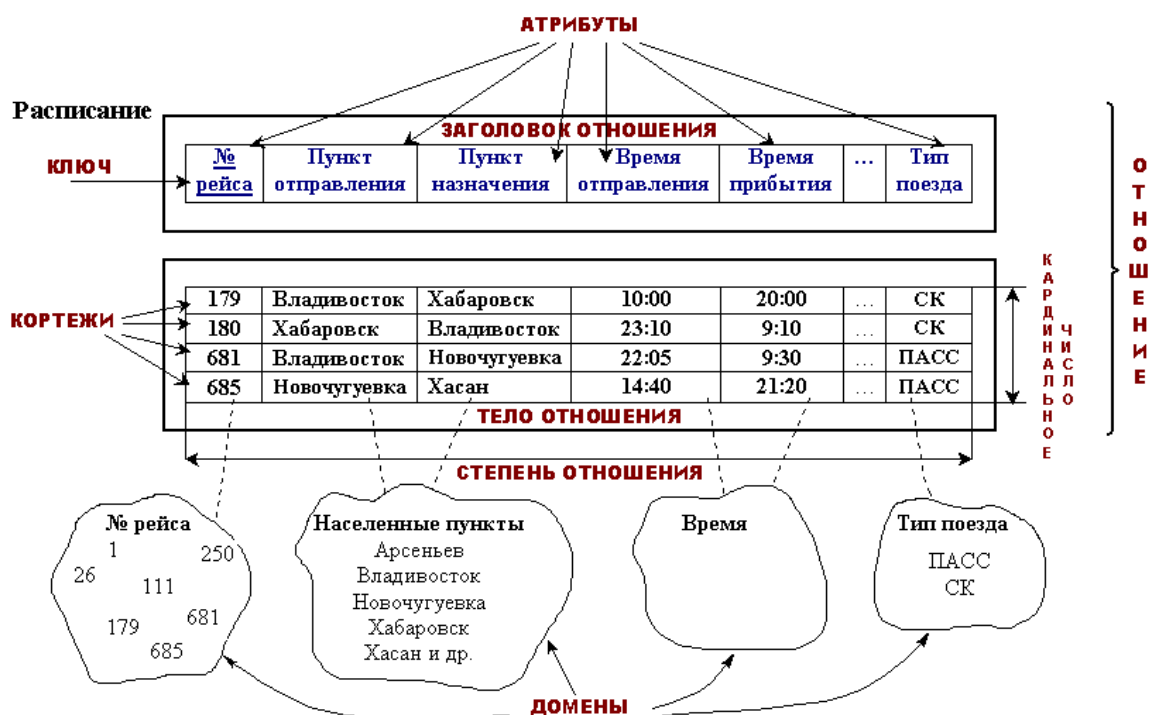


Рис. 2-15. Пример отношения ж/д расписание

Обратите внимание, что количество доменов меньше количества атрибутов, т.е. некоторые атрибуты определены на одном и том же домене. Так атрибуты *Пункт отправления* и *Пункт назначения* определены на домене *Населенные пункты*, а атрибуты *Время отправления* и *Время прибытия* – на домене *Время*. Т.е. атрибуты *Пункты отправления* и *Пункт назначения* (и соответственно *Время отправления* и *Время прибытия*) можно сравнивать. Например, для определения транзитных рейсов, которыми можно добраться из Владивостока в Хасан, необходимо

сравнить пункт назначения с пунктом отправления, а также время прибытия одного рейса с временем отправления другого (чтобы узнать есть ли запас времени для пересадки на транзитный рейс).

Схема отношения (заголовок отношения) выглядит как (№ рейса, Пункт отправления, Пункт назначения, Время отправления, Время прибытия, Тип поезда) или по определению схема представляет собой набор упорядоченных пар:

{<№ рейса : № рейса>,
<Пункт отправления : Населенные пункты>,
<Пункт назначения : Населенные пункты>,
<Время отправления : Время>,
<Время прибытия : Время>,
<Тип поезда : Тип поезда>},

где первым компонентом каждой пары является имя атрибута, а вторым компонентом – имя соответствующего домена. На практике чаще всего имена доменов в схеме опускают, и схема отношения представляет собой перечень атрибутов отношения.

Тело отношения представляет собой набор строк (кортежей). Рассмотрим подробнее один из кортежей:

(681, Владивосток, Новочугуевка, 22:05, 9:30, ПАСС)
по определению этот кортеж представляет собой набор упорядоченных пар:
{<№ рейса : 681>,
<Пункт отправления : 'Владивосток'>,
<Пункт назначения : 'Новочугуевка'>,
<Время отправления : 22:05>,
<Время прибытия : 9:30>,
<Тип поезда : 'ПАСС'>},

где первым компонентом каждой пары является имя атрибута, а вторым компонентом – значение соответствующего атрибута.

Часто на практике имена атрибутов опускают, так как известно, что каждое отдельное значение в таблице является значением атрибута, имя которого находится сверху соответствующего столбца; кроме того, значение принадлежит лежащему в основе этого атрибута домену. Например, значение “Владивосток” – это значение атрибута *Пункт отправления*, и оно взято из домена *Населенные пункты*.

Ключевым атрибутом отношения *Расписание* будет атрибут *№ рейса*, т.к. он однозначно идентифицирует кортежи. В самом деле, нет ни одного повторяющегося номера рейса, и по конкретному номеру рейса мы можем найти соответствующий кортеж отношения.

Свойства отношений

Рассмотрим теперь свойства отношений, которые следуют из приведенного выше определения отношения. В любом отношении

- Отсутствуют одинаковые кортежи
- Отсутствует упорядоченность кортежей
- Отсутствует упорядоченность атрибутов
- Все значения атрибутов атомарные

Отсутствие одинаковых кортежей

Это свойство следует из определения отношения как множества кортежей, а множества в математике по определению не содержат одинаковых элементов.

Это свойство служит прекрасным примером различия отношения и таблицы, т.к. таблица вполне может содержать одинаковые строки, а отношение не может содержать одинаковые кортежи.

Важным следствием этого свойства является наличие у каждого отношения так называемого **первичного ключа** – набора атрибутов, значения которых однозначно определяют кортеж отношения. Для каждого отношения, по крайней мере, полный набор его атрибутов обладает этим свойством. Однако при формальном определении первичного ключа требуется обеспечение его "минимальности", т.е. в набор атрибутов первичного ключа не должны входить такие атрибуты, которые можно отбросить без ущерба для основного свойства – однозначно определять кортеж. Понятие первичного ключа является исключительно важным в связи с понятием целостности баз данных.

Отсутствие упорядоченности кортежей

Свойство отсутствия упорядоченности кортежей (сверху вниз) также следует из того, что тело отношения – это математическое множество, а простые множества в математике не упорядочены.

Второе свойство отношений также служит примером различия отношения и таблицы, т.к. строки таблицы упорядочены сверху вниз, а кортежи отношения – нет.

Отсутствие требования к поддержанию порядка на множестве кортежей отношения дает дополнительную гибкость СУБД при хранении баз данных во внешней памяти и при выполнении запросов к базе данных. Это не противоречит тому, что при формулировании запроса к БД, например, на языке SQL можно потребовать сортировки результирующей таблицы в соответствии со значениями некоторых столбцов. Такой результат, вообще говоря, не отношение, а некоторый упорядоченный список кортежей.

Отсутствие упорядоченности атрибутов

Свойство отсутствия упорядоченности атрибутов (слева направо) следует из того факта, что схема отношения также определена как множество пар {имя атрибута, имя домена}. Для ссылки на значение атрибута в кортеже отношения всегда используется имя атрибута.

Это свойство также иллюстрирует отличие таблицы от отношения, поскольку столбцы таблицы упорядочены слева направо, а атрибуты отношения – нет.

Атомарность значений атрибутов

Значения всех атрибутов являются атомарными. Это свойство является следствием того, что все домены, лежащие в основе отношения, содержат только атомарные значения. Иначе можно сказать, что в каждой позиции пересечения столбца и строки таблицы расположено в точности одно значение, а не набор значений. Отношение, удовлетворяющее этому условию, называется нормализованным (представленным в первой нормальной форме). Т.е. с точки зрения реляционной модели все отношения нормализованы, поэтому в реляционных базах данных допускаются только нормализованные отношения или отношения, представленные в первой нормальной форме. Примером ненормализованного отношения является отношение R1 на Рис.2-16. Чтобы можно было использовать отношение в реляционной БД, его необходимо привести в виду отношения R2 (Рис. 2-16). Процесс получения отношения R2 из R1 называется нормализацией (подробнее процесс нормализации описан в Главе?).

Это свойство также иллюстрирует отличие таблицы от отношения. Строго говоря, на Рис. 2-16 только R2 является отношением, а таблицей можно назвать как R1, так и R2.

R1 – Ненормализованное отношение

КодПоставщика	КодПродукта	Продукт
P ₁	1	Сахар
	2	Соль
	13	Мука
P ₂	26	Рис
	58	Гречка
	130	Крупа манная
	162	Пшено
P ₃	474	Молоко
	891	Кефир

R2 – Нормализованное отношение

КодПоставщика	КодПродукта	Продукт
P ₁	1	Сахар
P ₁	2	Соль
P ₁	13	Мука
P ₂	26	Рис
P ₂	58	Гречка
P ₂	130	Крупа манная
P ₂	162	Пшено
P ₃	474	Молоко
P ₃	891	Кефир

Рис. 2-16. Пример нормализации отношения

2.2.2. Ограничения целостности в реляционной модели данных

Вторым аспектом реляционной модели данных является поддержка целостности.

Целостность данных понимается как правильность данных в любой момент времени при манипулировании данными. Поддержание целостности базы данных может рассматриваться как защита данных от неверных изменений или разрушений.

В классическом понимании поддержка целостности включает 3 части:

- Структурная целостность
- Языковая целостность
- Ссылочная целостность

Эти 3 вида целостности определяют допустимую форму представления и обработки информации в реляционных БД.

Для определения некоторых ограничений, связанных с содержанием БД, используется другой вид целостности, а именно:

- Семантическая целостность

Структурная целостность

Структурная целостность подразумевает, что реляционная СУБД может работать только с реляционными отношениями. А реляционное отношение, в свою очередь, должно удовлетворять ограничениям, накладываемым на него в классической теории реляционных БД (отсутствие одинаковых кортежей и, следовательно, наличие первичного ключа, отсутствие упорядоченности атрибутов и кортежей).

Требование структурной целостности осуществляется с помощью двух ограничений:

- при добавлении кортежей в отношение проверяется уникальность их первичных ключей
- не допускается, чтобы какой-либо атрибут, участвующий в первичном ключе, принимал неопределенное значение

Здесь возникает необходимость рассмотреть проблему неопределенных значений (Null-значений) [1, 2]. Неопределенное значение интерпретируется в реляционной модели как значение, неизвестное на данный момент времени. При сравнении неопределенных значений не действуют стандартные правила сравнения: одно Null-значение никогда не считается равным другому Null-значению.

Для выявления равенства значения некоторого атрибута неопределенному применяют стандартные предикаты:

- <Имя атрибута> Is Null
- <Имя атрибута> Is Not Null

Таблица 2-2 содержит пример проверки атрибута Адрес на неопределенное значение.

Таблица 2-2

Адрес	Адрес Is Null	Адрес Is Not Null
Null	True	False
ул.Мордовцева, 12	False	True

Введение Null-значений привело к модификации классической двузначной логики к трехзначной. Таблица 2-3 содержит таблицу истинности для трехзначной логики.

Таблица 2-3

A	B	Not A	A & B	A ∨ B
True	True	False	True	True
True	False	False	False	True
True	Null	False	Null	True
False	True	True	False	True
False	False	True	False	False
False	Null	True	False	Null
Null	True	Null	Null	True
Null	False	Null	False	Null
Null	Null	Null	Null	Null

Языковая целостность

Языковая целостность состоит в том, что реляционная СУБД должна обеспечивать языки описания и манипулирования данными не ниже стандарта SQL. Не должны быть доступны иные низкоуровневые средства манипулирования данными, не соответствующие стандарту.

Ссылочная целостность

При установлении связи между отношениями возникает необходимость поддержания целостности по ссылкам. Отношение со стороны «один» будем называть – основным отношением, а отношение со стороны «многие» – подчиненным.

Требование ссылочной целостности состоит в следующем: для каждого значения внешнего ключа, появляющегося в подчиненном отношении, в основном отношении должен существовать кортеж с таким же значением первичного ключа.

У первичного и внешнего ключей, образующих связь, должен быть одинаковый тип данных.

То есть значение внешнего ключа должно либо:

- быть равным значению первичного ключа
- быть полностью неопределенным, т.е. каждое значение поля, участвующего во внешнем ключе должно быть неопределенным.

Для каждого внешнего ключа в процессе проектирования необходимо решить три вопроса:

1. Может ли данный внешний ключ принимать неопределенные значения
2. Что произойдет при попытке УДАЛЕНИЯ записи из основного отношения, на которую ссылается внешний ключ подчиненного отношения?

Например, удалить поставщика, для которого имеется, по крайней мере, одна поставка.

В общем случае существует три ситуации:

- *Каскадирование удаления*, при котором удаляются все записи из подчиненного отношения, соответствующие удаляемому первичному ключу основного отношения (будет удален поставщик и все его поставки)
 - *Ограничение удаления*, при котором удаляется запись из основного отношения только в том случае, если в подчиненном отношении нет соответствующих значений внешнего ключа, иначе удаление отменяется (удаление поставщика невозможно, пока существует хотя бы одна его поставка)
 - *Установка неопределенных значений*, при которой внешний ключ подчиненного отношения устанавливается в неопределенное значение (Null-значение), а соответствующая запись из основного отношения удаляется (все значения внешнего ключа в поставках принимают Null-значение, а поставщик удаляется)
Данное свойство поддерживается не всеми СУБД. Если необходимо применить эту ситуацию, то в подчиненном отношении сначала нужно удалить все значения внешнего ключа соответствующие первичному, и только после этого удалять запись из основного отношения с соответствующим первичным ключом
3. Что произойдет при попытке ОБНОВЛЕНИЯ первичного ключа основного отношения, на который ссылается некоторый внешний ключ подчиненного отношения? Например, при попытке обновления кода поставщика, для которого имеется хотя бы одна поставка.

Здесь также возможны три ситуации:

- *Каскадирование обновления*, при котором при обновлении первичного ключа обновляются все соответствующие внешние ключи (будет обновлен код поставщика в основном отношении и все соответствующие ему внешние ключи в поставках)
- *Ограничение обновления*, при котором обновляется первичный ключ в основном отношении только в том случае, если в подчиненном отношении нет соответствующих значений внешнего ключа, иначе обновление отменяется (обновление кода поставщика невозможно, пока существует хотя бы одна поставка этого поставщика)
- *Установка неопределенных значений*, при которой внешний ключ подчиненного отношения устанавливается в неопределенное значение, а соответствующий первичный ключ в основном отношении обновляется (все значения внешнего ключа в поставках принимают Null-значение, а код поставщика в основном отношении обновляется)

Семантическая целостность

Данный вид целостности задается разработчиком в процессе проектирования БД посредством задания ограничений для свойств полей. Обычно задаются ограничения свойств:

- *уникальность значений полей*. Например, в отношении Студент(№ зачетной книжки, ФИО, Паспорт, Адрес) свойство уникальности значений должно быть установлено для атрибутов: № зачетной книжки (т.к. это первичный ключ) и Паспорт (т.к. номера всех паспортов уникальны)
- *обязательность заполнения полей* (допустимость или недопустимость Null-значений). Например, при вводе данных о поставщиках не вся информация может быть доступна сразу: адрес, телефоны для связи могут быть уточнены позднее. Т.е. для атрибутов *Код города*, *Адрес*, *Телефон* устанавливается допустимость Null-значений
- *значение по умолчанию*. Задание значения по умолчанию по умолчанию означает, что каждый раз при вводе новой строки в отношение, при отсутствии данных этому атрибуту присваивается значение по умолчанию. Например, если большинство поставщиков находятся во Владивостоке, то для атрибута *Код города* присваивается значение по умолчанию соответствующее коду Владивостока
- *диапазон значений*. Например, оценки выставляются по пяти бальной шкале от 1 до 5, тогда условие для этого диапазона (для MS Access) будет выглядеть как: Between 1 And 5

- *принадлежность набору значений* Например, атрибут *РезультатЗачета* может принимать значения только «Зачтено» или «Не зачтено», тогда условие на проверку принадлежности набору значений (для MS Access) будет выглядеть как: “Зачтено” Or “Не зачтено”.

2.2.3. Реляционная алгебра

Третьим аспектом реляционной модели данных является обработка данных, осуществляемая с помощью операторов реляционной алгебры. В основном операторы имеют на входе отношения и возвращают отношения в качестве результата.

Реляционная алгебра состоит из восьми операторов: четырех традиционных операций над множествами (теоретико-множественных операций) и четырех специальных реляционных операций.

К традиционным операциям относятся операции:

- объединение
- возвращает отношение, содержащее все кортежи, принадлежащие или одному из двух определенных отношений, или обоим
- пересечение
- возвращает отношение, содержащее все кортежи, принадлежащие одновременно двум определенным отношениям
- вычитание
- возвращает отношение, содержащее все кортежи, которые принадлежат первому из двух определенных отношений и не принадлежат второму
- расширенное декартово произведение
- возвращает отношение, содержащее всевозможные кортежи, являющиеся сочетанием двух кортежей, принадлежащих соответственно двум определенным отношениям

К специальным операциям относятся:

- выборка (ограничение)
- возвращает отношение, содержащее все кортежи из определенного отношения, удовлетворяющие определенным условиям
- проекция
- возвращает отношение, содержащее все кортежи (называемые как подкортежи) определенного отношения после исключения из него некоторых атрибутов
- соединение (естественное)
- возвращает отношение, кортежи которого – это сочетание двух кортежей (принадлежащих соответственно двум определенным отношениям), имеющих общее значение для одного или нескольких атрибутов этих двух отношений (и такие общие значения в результирующем кортеже появляются только один раз)
- деление
- для двух отношений, бинарного и унарного, возвращает отношение, содержащее все значения одного атрибута бинарного отношения, соответствующее (в другом атрибуте) всем значениям в унарном отношении

Замкнутость

Как уже отмечалось, результат каждой операции над отношением также является отношением. Это реляционное свойство называется свойством **замкнутости**. Отсюда можно сделать вывод: поскольку результат любой операции имеет тот же тип, что и исходные объекты (отношения), то результат одной операции может использоваться в качестве исходных данных для другой. Другими словами, можно записывать **вложенные выражения**, т.е. выражения, в которых операнды сами представлены выражениями вместо простых имен отношений.

Совместимость по типу

Операции объединения, пересечения и вычитания требуют от операндов совместимости по типу. Будет говорить, что два отношения **совместимы по типу**, если у них **эквивалентные схемы**, а точнее:

1. если каждое из них имеет одно и то же множество атрибутов (а значит и одинаковую степень)
2. если возможно такое упорядочение атрибутов в схемах, что на одинаковых местах будут находиться сравнимые атрибуты, т.е. атрибуты, определенные на одном и том же домене

Пример: имеются следующие отношения (Рис. 2-17)

Отношение *Продукты1* содержит продукты, имеющиеся в магазине

Отношение *Продукты2* содержит продукты, поставляемые поставщиком Р₂

Отношение *Поставщики* содержит поставщиков продуктов

Отношение *ВидПродукта* содержит виды продуктов

Первые три отношения имеют одинаковую степень, т.е. выполняется первое условие совместимости по типу. Второе условие выполняется только для отношений *Продукты1* и *Продукты2*, т.е. только эти отношения совместимы по типу, а значит с ними можно выполнять операции объединения, пересечения и вычитания.

Продукты1

КодПродукта	Продукт	КодПоставщика
1	Сахар	Р ₁
2	Соль	Р ₁
13	Мука	Р ₁
26	Рис	Р ₂
58	Гречка	Р ₂
130	Крупа манная	Р ₂
162	Пшено	Р ₂
474	Молоко	Р ₃
891	Кефир	Р ₃

Продукты2

КодПродукта	Продукт	КодПоставщика
26	Рис	Р ₂
35	Перловка	Р ₂
58	Гречка	Р ₂
130	Крупа манная	Р ₂
162	Пшено	Р ₂
200	Крупа ячневая	Р ₂

Поставщики

КодП	Наименование	Город
Р ₁	ООО «Восток»	Владивосток
Р ₂	ОАО «Приморье»	Уссурийск
Р ₃	ПБООЛ Сидоров А.С.	Находка
Р ₄	ОАО «Владхлеб»	Владивосток

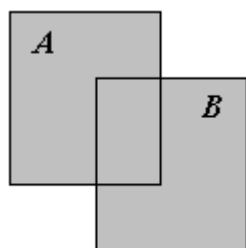
ВидПродукта

КодВида	Вид
1	Молочная
2	Мясная
3	Хлебопродукция

Рис. 2-17. База данных продуктов и поставщиков (значения для примера)

Теоретико-множественные операции реляционной алгебры

Объединением двух совместимых по типу отношений *A* и *B* называется отношение с тем же заголовком, как в исходных отношениях, и с телом, состоящим из множества всех кортежей, принадлежащих *A* или *B* или обоим отношениям (за исключением повторяющихся).



Пусть заданы два отношения $A=\{a\}$, $B=\{b\}$, где a и b – соответственно кортежи отношений *A* и *B*, то объединение

$$A \cup B = \{c \mid c \in A \vee c \in B\},$$

Здесь c – кортеж нового отношения, \vee – операция логического сложения «ИЛИ».

Пример: Объединим, приведенные на Рис. 2-17, отношения *Продукты1* (содержащее продукты, имеющиеся в магазине) и *Продукты2* (содержащее продукты, поставляемые поставщиком Р₂). Результатом объединения станет отношение *R1* (Рис. 0-18), содержащее продукты, которые или имеются в магазине или поставляются поставщиком Р₂ (либо и то и другое).

Обратите внимание, что дублирующие кортежи исключены из результирующего отношения *R1*.

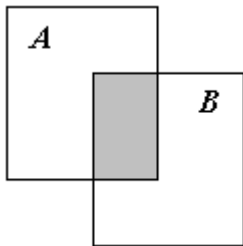
$$R1 = \text{Продукты1} \cup \text{Продукты2}$$

R1

КодПродукта	Продукт	КодПоставщика
1	Сахар	P ₁
2	Соль	P ₁
13	Мука	P ₁
26	Рис	P ₂
58	Гречка	P ₂
130	Крупа манная	P ₂
162	Пшено	P ₂
474	Молоко	P ₃
891	Кефир	P ₃
35	Перловка	P ₂
200	Крупа ячневая	P ₂

Рис. 2-18. Пример объединения

Пересечением двух совместимых по типу отношений A и B называется отношение с тем же заголовком, как в исходных отношениях, и с телом, состоящим из множества всех кортежей, принадлежащих одновременно обоим отношениям A и B .



$$A \cap B = \{c \mid c \in A \wedge c \in B\},$$

Здесь \wedge – операция логического умножения (логическое «И»).

Пример: Пересечением отношений *Продукты1* и *Продукты2* (Рис. 2-17) станет отношение $R2$ (Рис. 2-19), содержащее продукты, имеющиеся в магазине и поставляемые поставщиком P_2 .

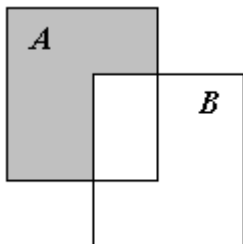
$$R2 = \text{Продукты1} \cap \text{Продукты2}$$

R2

КодПродукта	Продукт	КодПоставщика
26	Рис	P ₂
58	Гречка	P ₂
130	Крупа манная	P ₂
162	Пшено	P ₂

Рис. 2-19. Пример пересечения

Вычитанием двух совместимых по типу отношений A и B называется отношение с тем же заголовком, как в исходных отношениях, и с телом, состоящим из множества всех кортежей, принадлежащих отношению A и не принадлежащих отношению B .



$$A \setminus B = \{c \mid c \in A \wedge c \notin B\}$$

Отметим, что операции объединения и пересечения являются *коммутативными* операциями, т.е. результат операции не зависит от порядка аргументов в операции. Операция вычитания является несимметричной операцией, т.е. результат операции будет различным для разного порядка аргументов.

Пример: При вычитании отношения *Продукты2* из отношения *Продукты1* (Рис. 2-17) получится отношение $R3$ (Рис. 2-20), содержащее продукты, имеющиеся в магазине, кроме тех продуктов, которые поставляет поставщик P_2 .

При вычитании отношения *Продукты1* из отношения *Продукты2* получится другое отношение $R4$ (поскольку операция вычитания не коммутативная). Отношение $R4$ (Рис. 2-20) будет содержать продукты, поставляемые поставщиком P_2 , кроме тех продуктов, которые имеются в магазине.

$$R3 = \text{Продукты1} \setminus \text{Продукты2}$$

R3

КодПродукта	Продукт	КодПоставщика
1	Сахар	P ₁
2	Соль	P ₁
13	Мука	P ₁
474	Молоко	P ₃
891	Кефир	P ₃

$$R4 = \text{Продукты2} \setminus \text{Продукты1}$$

R4

КодПродукта	Продукт	КодПоставщика
35	Перловка	P ₂
200	Крупа ячневая	P ₂

Рис. 2-21. Примеры вычитания

Расширенное декартово произведение

Эта операция не накладывает никаких дополнительных условий на схемы исходных отношений, поэтому операция расширенного декартова произведения допустима для любых двух отношений.

Прежде чем определить саму операцию, введем дополнительно понятие конкатенации, или сцепления, кортежей.

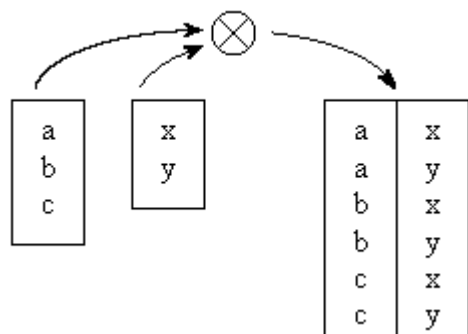
Сцеплением, или конкатенацией, кортежей $s = \langle c_1, c_2, \dots, c_n \rangle$ и $q = \langle q_1, q_2, \dots, q_m \rangle$ называется кортеж, полученный добавлением значений второго в конец первого. Сцепление кортежей s и q обозначается как (s, q) .

$$(s, q) = \langle c_1, c_2, \dots, c_n, q_1, q_2, \dots, q_m \rangle$$

здесь n – число элементов в первом кортеже s , m – число элементов во втором кортеже q .

Все предыдущие операция не меняли степени или арности отношений – это следует из определения эквивалентности схем отношений. Операция расширенного декартова произведения меняет степень результирующего отношения.

Расширенное декартово произведение двух отношений A и B , где A и B не имеют общих атрибутов, определяется как отношение с заголовком, который представляет собой сцепление двух заголовков исходных отношений A и B , и телом, состоящим из множества всех кортежей s , таких, что s представляет собой сцепление кортежа a , принадлежащего отношению A , и кортежа b , принадлежащего отношению B .



Т.е. если $A = \{a\}$, $B = \{b\}$, то расширенное декартово произведение

$$A \otimes B = \{(a, b) \mid a \in A \wedge b \in B\}$$

Обратите внимание, что кардинальное число результата равно произведению кардинальных чисел исходных отношений A и B , а степень равна сумме их степеней.

Операция декартова произведения довольно редко используется как самостоятельная операция, чаще результат этой операции подвергается дальнейшей обработке.

Пример: Декартовым произведением отношений *Поставщики* и *ВидПродукта* (Рис. 2-17) будет отношение $R5$ (Рис. 2-21). Отношение $R5$ соответствует ситуации, когда **все** поставщики поставляют **все** виды продуктов.

R5 = Поставщики \otimes ВидПродукта

R5

КодП	Наименование	Город	КодВида	Вид
P ₁	ООО «Восток»	Владивосток	1	Молочная
P ₁	ООО «Восток»	Владивосток	2	Мясная
P ₁	ООО «Восток»	Владивосток	3	Хлебопродукция
P ₂	ОАО «Приморье»	Уссурийск	1	Молочная
P ₂	ОАО «Приморье»	Уссурийск	2	Мясная
P ₂	ОАО «Приморье»	Уссурийск	3	Хлебопродукция
P ₃	ПБОЮЛ Сидоров А.С.	Находка	1	Молочная
P ₃	ПБОЮЛ Сидоров А.С.	Находка	2	Мясная
P ₃	ПБОЮЛ Сидоров А.С.	Находка	3	Хлебопродукция
P ₄	ОАО «Владхлеб»	Владивосток	1	Молочная
P ₄	ОАО «Владхлеб»	Владивосток	2	Мясная
P ₄	ОАО «Владхлеб»	Владивосток	3	Хлебопродукция

Рис. 2-21. Пример расширенного декартова произведения

Операцию декартова произведения, с учетом возможности перестановки атрибутов в отношении, можно считать коммутативной. Таким образом, все операции, кроме вычитания, являются *коммутативными*, т.е.

$$A \cup B = B \cup A$$

$$A \cap B = B \cap A$$

$$A \otimes B = B \otimes A$$

Все теоретико-множественные операции являются *ассоциативными* операциями, т.е.

$$A \cup (B \cup C) = (A \cup B) \cup C = A \cup B \cup C$$

$$A \cap (B \cap C) = (A \cap B) \cap C = A \cap B \cap C$$

$$A \setminus (B \setminus C) = (A \setminus B) \setminus C = A \setminus B \setminus C$$

$$A \otimes (B \otimes C) = (A \otimes B) \otimes C = A \otimes B \otimes C$$

Специальные реляционные операции

Выборка (ограничение, горизонтальное подмножество)

Для определения этой операции необходимо ввести дополнительные обозначения.

Пусть α – булевское выражение, составленное из термов сравнения с помощью связок И (\wedge), ИЛИ (\vee), НЕ (\neg) и, возможно, скобок. В качестве термов сравнения допускаются:

1. терм $A \theta a$,

где A – имя некоторого атрибута, принимающего значения из домена D ; a – константа, взятая из того же домена D , $a \in D$; θ – одна из допустимых для данного домена D операций сравнения ($=$, \neq , $<$, \leq , $>$, \geq);

2. терм $A \theta B$,

где A , B – имена некоторых θ -сравнимых атрибутов, то есть атрибутов, принимающих значение из одного и того же домена D .

Тогда **выборкой** (θ -выборкой), заданной на отношении A в виде булевского выражения, определенного на атрибутах отношения A , называется отношение, имеющее тот же заголовок, что и отношение A , и тело, содержащее множество всех кортежей отношения A , для которых истинно условие выбора или ограничения:

$$A[\alpha(t)] = \{t \mid t \in A \wedge \alpha(t) = \text{“Истина”}\}$$

Операция ограничения является одной из основных при работе с реляционной моделью данных. Условие α может быть сколь угодно сложным.

Пример:

- Результатом выборки продуктов, поставляемых поставщиком P_3 , из отношения *Продукты1* (Рис. 2-17) будет отношение R_6 (Рис. 2-22, а)
- Результатом выборки Владивостокских поставщиков из отношения *Поставщики* (Рис. 2-17) будет отношение R_7 (Рис. 2-22, б)

а) $R6 = \text{Продукты1} [\text{КодПоставщика} = "P_3"]$

R6

КодПродукта	Продукт	КодПоставщика
474	Молоко	P ₃
891	Кефир	P ₃

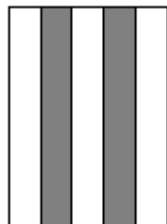
б) $R7 = \text{Поставщики} [\text{Город} = "Владивосток"]$

R7

КодП	Наименование	Город
P ₁	ООО «Восток»	Владивосток
P ₄	ОАО «Владхлеб»	Владивосток

Рис. 2-22. Примеры операций выборки

Проекцией отношения A по атрибутам X, Y, \dots, Z , где каждый из атрибутов принадлежит отношению $A(A[X, Y, \dots, Z])$, называется отношение с заголовком $\{X, Y, \dots, Z\}$ и телом, содержащим множество всех кортежей $\{X:x, Y:y, \dots, Z:z\}$, таких, для которых в отношении A значение атрибута X равно x , атрибута Y равно y , ..., атрибута Z равно z . Таким образом, с помощью оператора проекции получено «вертикальное» подмножество данного отношения, т.е. подмножество, получаемое исключением всех атрибутов, не указанных в списке атрибутов, и последующим исключением дублирующих кортежей (подкортежей) из того, что осталось.



Никакой атрибут не может быть указан в списке атрибутов более одного раза.

Пример:

- Проекцией отношения *Продукты1* (Рис. 2-17) по атрибуту *КодПоставщика* будет отношение *R8* (Рис. 2-23, а). Обратите внимание, что дублирующие кортежи исключены из отношения *R8*
- Проекцией отношения *Поставщики* (Рис. 2-17) по атрибуту *Город* будет отношение *R9* (Рис. 2-23, б)
- Довольно часто операция проекции используется в сочетании с другими операциями. Например, нужно выбрать названия поставщиков из Владивостока (на основе отношения *Поставщики* – Рис. 2-17). Сначала выполняется операция выборки, а затем – проекция (Рис. 2-23, с).

а) $R8 = \text{Продукты1} [\text{КодПоставщика}]$

R8

КодПоставщика
P ₁
P ₂
P ₃

б) $R9 = \text{Поставщики} [\text{Город}]$

R9

Город
Владивосток
Уссурийск
Находка

с) $R7 = \text{Поставщики} [\text{Город} = "Владивосток"]$

$R10 = R7 [\text{Наименование}]$

или $R10 = (\text{Поставщики} [\text{Город} = "Владивосток"]) [\text{Наименование}]$

R7

КодП	Наименование	Город
P ₁	ООО «Восток»	Владивосток
P ₄	ОАО «Владхлеб»	Владивосток

R10

Наименование
ООО «Восток»
ОАО «Владхлеб»

Рис. 2-23. Примеры операций проекции

Соединение (естественное, условное)

Операция соединения имеет несколько разновидностей. Однако наиболее важным является естественное соединение, поэтому часто для обозначения именно естественного соединения используют общий термин «соединение».

Пусть отношения A и B имеют заголовки:

$\{X_1, X_2, \dots, X_m, Y_1, Y_2, \dots, Y_n\}$

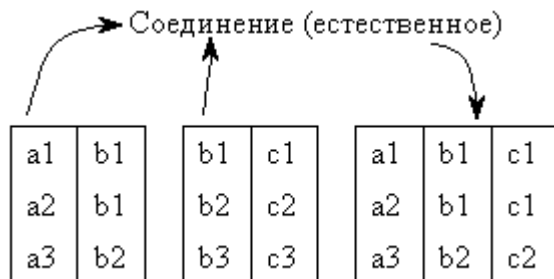
и

$\{Y_1, Y_2, \dots, Y_n, Z_1, Z_2, \dots, Z_p\}$ соответственно;

т.е. атрибуты Y_1, Y_2, \dots, Y_n (и только они) – общие для двух отношений;

X_1, X_2, \dots, X_m – остальные атрибуты отношения A ; Z_1, Z_2, \dots, Z_p – остальные атрибуты отношения B . Предположим также, что соответствующие атрибуты (т.е. атрибуты с одинаковыми именами) определены на одном и том же домене. Будем рассматривать выражения $\{X_1, X_2, \dots, X_m\}$, $\{Y_1, Y_2, \dots, Y_n\}$, $\{Z_1, Z_2, \dots, Z_p\}$ как три составных атрибута X, Y, Z соответственно.

Тогда **естественным соединением** отношений A и B называется отношение с заголовком $\{X, Y, Z\}$ и телом, содержащим множество всех кортежей $\{X:x, Y:y, Z:z\}$, таких, для которых в отношении A значение атрибута X равно x , а атрибута Y равно y , и в отношении B значение атрибута Y равно y , а атрибута Z равно z .



Естественное соединение обладает свойствами *коммутативности* и *ассоциативности*.

Отметим также, что если отношения A и B не имеют общих атрибутов, то естественное соединение превращается в декартово произведение.

Пример: Рассмотрим отношения *Продукты1* и *Поставщики* (Рис. 2-17). Атрибуты *КодПоставщика* и *КодП* определены на одном и том же домене кодов поставщиков. Поскольку при естественном соединении также требуется, чтобы общие атрибуты соединяемых отношений имели одинаковые имена, переименуем атрибут *КодП* отношения *Поставщики* в *КодПоставщика*. Тогда естественным соединением отношений *Продукты1* и *Поставщики* по атрибуту *КодПоставщика* будет отношение *R11* (Рис. 2-24).

$R11 = \text{Продукты1} [\text{Продукты1.КодПоставщика} = \text{Поставщики.КодПоставщика}] \text{Поставщики}$

R11

КодПродукта	Продукт	КодПоставщика	Наименование	Город
1	Сахар	P ₁	ООО «Восток»	Владивосток
2	Соль	P ₁	ООО «Восток»	Владивосток
13	Мука	P ₁	ООО «Восток»	Владивосток
26	Рис	P ₂	ОАО «Приморье»	Уссурийск
58	Гречка	P ₂	ОАО «Приморье»	Уссурийск
130	Крупа манная	P ₂	ОАО «Приморье»	Уссурийск
162	Пшено	P ₂	ОАО «Приморье»	Уссурийск
474	Молоко	P ₃	ПБОЮЛ Сидоров А.С.	Находка
891	Кефир	P ₃	ПБОЮЛ Сидоров А.С.	Находка

Рис. 2-24. Пример естественного соединения

Рассмотрим теперь **условное соединение** (или θ -соединение). Эта операция используется, когда необходимо соединить два отношения на основе некоторых условий, отличных от эквивалентности.

Пусть отношения A и B не имеют общих имен атрибутов, и θ определяется как в операции выборки. Тогда **условным соединением** отношения A по атрибуту X с отношением B по атрибуту Y называется отношение с заголовком, который представляет собой сцепление двух заголовков исходных отношений A и B (как и при операции декартова произведения), и с телом, содержащим множество кортежей t , таких что t принадлежит этому декартову произведению и вычисление условия " $X \theta Y$ " дает значение «истина» для этого кортежа. Атрибуты X и Y должны быть определены на одном и том же домене, а операция должна иметь смысл для этого домена.

Пример: Получить названия продуктов (отношение *Продукты1* – Рис. 2-17), поставляемых поставщиками из Владивостока (отношение *Поставщики* – Рис. 2-17). По сути, в этом примере необходимо использовать две операции: условного соединения – для получения непосредственно

списка продуктов, поставляемых Владивостокскими поставщиками (*R12*); и проекции – для получения только названий продуктов (*R13*) (Рис. 2-25).

$R12 = \text{Продукты1} [(\text{Продукты1.КодПоставщика} = \text{Поставщики.КодП}) \wedge$
 $\text{Поставщики.Город} = \text{“Владивосток”}] \text{Поставщики}$
 $R13 = R12 [\text{Продукт}]$
или
 $R13 = (\text{Продукты1} [(\text{Продукты1.КодПоставщика} = \text{Поставщики.КодП}) \wedge$
 $\text{Поставщики.Город} = \text{“Владивосток”}] \text{Поставщики}) [\text{Продукт}]$

R12

КодПродукта	Продукт	КодПоставщика	КодП	Наименование	Город
1	Сахар	P ₁	P ₁	ООО «Восток»	Владивосток
2	Соль	P ₁	P ₁	ООО «Восток»	Владивосток
13	Мука	P ₁	P ₁	ООО «Восток»	Владивосток

R13

Продукт
Сахар
Соль
Мука

Рис. 2-25. Пример условного соединения

Деление

Пусть отношения А и В имеют заголовки:

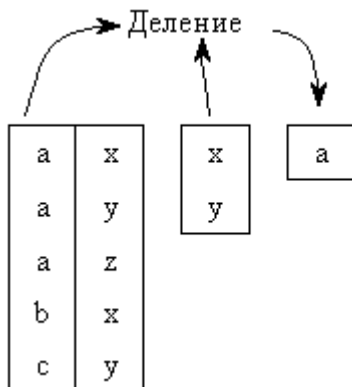
{X₁, X₂,..., X_m, Y₁, Y₂,..., Y_n}

и

{Y₁, Y₂,..., Y_n} соответственно;

т.е. атрибуты Y₁, Y₂,..., Y_n – общие для двух отношений, и отношение А

имеет дополнительные атрибуты X₁, X₂,..., X_m, а отношение В не имеет дополнительных атрибутов. (Отношения А и В представляют соответственно делимое и делитель). Предположим также, что соответствующие атрибуты (т.е. атрибуты с одинаковыми именами) определены на одном и том же домене. Пусть выражения {X₁, X₂,..., X_m} и {Y₁, Y₂,..., Y_n} обозначают два составных атрибута X и Y соответственно.



Тогда делением отношений А и В называется отношение с заголовком {X} и телом, содержащим множество всех кортежей {X:x} таких, что существует кортеж {X:x, Y:y}, который принадлежит отношению А для всех кортежей {Y:y}, принадлежащих отношению В.

Нестрого это можно сформулировать так: результат содержит такие X-значения из отношения А, для которых соответствующие Y-значения (из А) включают все Y-значения из отношения В.

Если запрос на естественном языке включает слово “все” (“получить поставщиков, поставляющих все виды продуктов”), то почти наверняка потребуется операция деления.

Пример: Пусть отношение R14 содержит поставщиков и виды поставляемых ими продуктов, а отношение ВидПродукта содержит виды продуктов (Рис. 2-26). Тогда, чтобы получить поставщиков поставляющих ВСЕ виды продуктов, необходимо отношение R14 разделить на отношение ВидПродукта по атрибуту КодВида (Рис. 2-26).

$R15 = R14 [\text{КодВида} \div \text{КодВида}] \text{ ВидПродукта}$

R14

КодП	Наименование	КодВида
P ₁	ООО «Восток»	1
P ₁	ООО «Восток»	2
P ₂	ОАО «Приморье»	1
P ₂	ОАО «Приморье»	2
P ₂	ОАО «Приморье»	3
P ₃	ПБОЮЛ Сидоров А.С.	3
P ₄	ОАО «Владхлеб»	1
P ₄	ОАО «Владхлеб»	2
P ₄	ОАО «Владхлеб»	3

ВидПродукта

КодВида	Вид
1	Молочная
2	Мясная
3	Хлебопродукция

R15

КодП	Наименование
P ₂	ОАО «Приморье»
P ₄	ОАО «Владхлеб»

Рис. 2-26. Пример операции деления

Итак, мы рассмотрели операции реляционной алгебры. Эти восемь операций (объединение, пересечение, вычитание, декартово произведение, выборка, проекция, соединение, деление) не представляют собой *минимальный* набор операций. Т.е. некоторые операции можно выразить через другие операции, а именно – соединение, пересечение и деление. Например, соединение – это проекция выборки декартова произведения. Таким образом, **примитивными операциями**, т.е. которые нельзя выразить через другие операции, являются остальные пять операций: объединение, вычитание, выборка, проекция, декартово произведение. Эти пять примитивных операций будут составлять минимальный набор операций реляционной алгебры.

Однако на практике другие три операции (особенно соединение) настолько часто используются, что имеет смысл обеспечить их непосредственную поддержку, несмотря на то, что они не примитивны.

Примеры использования операций реляционной алгебры

Рассмотрим несколько примеров для демонстрации возможностей операций реляционной алгебры.

I. Пусть даны три отношения с эквивалентными схемами: (ФИО, Должность, Отдел). Отношение R1 содержит список сотрудников предприятия, работающих над проектом P1, R2 – список сотрудников, работающих над проектом P2, R3 – список сотрудников, живущих во Владивостоке. Сотрудники могут работать над несколькими проектами.

Составить формулы для следующих запросов:

1. Список сотрудников, живущих не во Владивостоке, т.е. это такие сотрудники, которые могут содержаться в отношении R1 или R2 или обоих отношениях, но не входят в отношение R3

$$R1 \cup R2 \setminus R3 \quad \text{или} \quad (R1 \setminus R3) \cup (R2 \setminus R3)$$
этот пример показывает, что один и тот же запрос можно сформулировать несколькими способами
2. Список сотрудников, работающих над двумя проектами и живущих во Владивостоке, т.е. это сотрудники, содержащиеся во всех трех отношениях

$$R1 \cap R2 \cap R3$$
3. Список сотрудников, работающих только над одним из проектов

$$(R1 \cup R2) \setminus (R1 \cap R2)$$
4. Список сотрудников, работающих только над одним из проектов, и живущих во Владивостоке

$$(R1 \cup R2) \setminus (R1 \cap R2) \cap R3 \quad \text{или} \quad (R1 \cap R3) \cup (R2 \cap R3) \setminus (R1 \cap R2)$$

II. Даны следующие отношения:

R1(Таб№, ФИО, Должность, Отдел) – список сотрудников по отделам

R2(Должность, Оклад) – должностные оклады

R3(Должность, Отдел) – штатные должности по отделам, т.е. какие должности должны быть в каждом отделе.

Сотрудник может занимать несколько должностей в одном и том же или разных отделах.

Составить формулы для следующих запросов:

1. Табельные номера и ФИО сотрудников отдела “Бухгалтерия”

$$R1[\text{Отдел} = \text{“Бухгалтерия”}] [\text{Таб№, ФИО}]$$
2. ФИО сотрудников с окладом больше 3000 руб

$$R1[R1.\text{Должность} = R2.\text{Должность} \wedge R2.\text{Оклад} > 3000] R2 [\text{ФИО}]$$
3. Список вакантных должностей в отделах

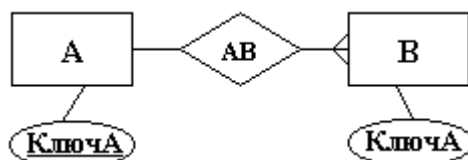
$$R3 \setminus (R1[\text{Должность, Оклад}])$$

4. Отделы, не имеющие по штату должность “Мастер”
 $R3 \setminus R3[\text{Должность} = \text{“Мастер”}]$
5. ФИО сотрудников, занимающих больше одной должности
 $(R1[R1.\text{ФИО} = R1'.\text{ФИО} \wedge R1.\text{Должность} \neq R1'.\text{Должность}]R1')[\text{ФИО}]$
 Обратите внимание, что для поиска строк, удовлетворяющих условию больше одного, применяется операция соединения отношения с самим собой. Поэтому мы взяли копию отношения R1 и назвали ее R1'.
6. ФИО сотрудников, занимающие только одну должность
 найдем сначала сотрудников, занимающих больше одной должности
 $R4 = (R1[R1.\text{ФИО} = R1'.\text{ФИО} \wedge R1.\text{Должность} \neq R1'.\text{Должность}]R1')[\text{ФИО}]$
 вычтем из проекции R1 по ФИО полученное отношение R4
 $(R1[\text{ФИО}]) \setminus R4$
7. Отделы, имеющие в штате, по крайней мере, все те должности, что и в отделе “Цех 1”
 найдем все должности положенные по штату в отделе “Цех 1”
 $R4 = (R3[\text{Отдел} = \text{“Цех 1”}]) [\text{Должность}]$
 ответ на поставленный вопрос
 $R3[\text{Должность} \supseteq \text{Должность}]R4$

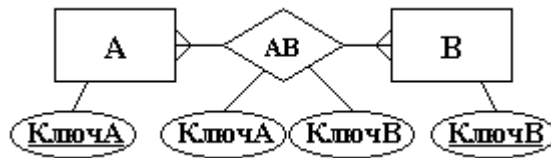
2.2.4. Алгоритм перехода от модели «сущность-связь» к реляционной модели

Модель «сущность-связь» используется на ранних стадиях проектирования БД. Как уже говорилось модель «сущность-связь» является **концептуальной моделью** и не учитывает особенности конкретной **СУБД** (допустимые типы и наименования полей и таблиц, ограничения **целостности** и т.п.). Существует алгоритм однозначного преобразования модели «сущность-связь» в реляционную модель данных (т.е. осуществляется переход от **инфологического моделирования** к логическому проектированию **схемы** реляционной БД). Рассмотрим этот алгоритм.

1. Каждой сущности модели «сущность-связь» ставится в соответствие отношение реляционной модели. При этом на имена отношений накладываются ограничения, присущие конкретной СУБД.
2. Каждый атрибут сущности становится атрибутом соответствующего отношения. На имена атрибутов отношения также накладываются ограничения выбранной СУБД. Для каждого атрибута задается конкретный допустимый в СУБД тип данных и обязательность или необязательность данного атрибута (т.е. допустимость или недопустимость Null-значений)
3. Первичный ключ сущности становится первичным ключом соответствующего отношения. Атрибуты, входящие в первичный ключ отношения, автоматически получают свойство отсутствия неопределенных значений (Not Null).
4. В каждое отношение, соответствующее сущности со стороны «многие» (связь 1:N), добавляется набор атрибутов сущности со стороны «один», являющихся первичным ключом сущности со стороны «один». В отношении, соответствующим сущности со стороны «многие», этот набор атрибутов становится внешним ключом.



5. Для моделирования необязательного класса принадлежности у атрибутов, соответствующих внешнему ключу, устанавливается свойство допустимости неопределенных значений. При обязательном классе принадлежности атрибуты получают свойство отсутствия неопределенных значений.
6. Разрешение связей типа M:N. Связи становится в соответствие отношение, имеющего атрибуты, которые в сущностях являются первичными ключами, а в новом отношении будут внешними ключами. Первичным ключом нового отношения будет совокупность внешних ключей.



Пример преобразования модели «сущность-связь» к реляционной модели

Рассмотрим на примере, как осуществить переход от модели «сущность-связь» к реляционной модели. Возьмем пример, приведенный в параграфе 2.1.2. На Рис. 2-14 приведен окончательный вариант диаграммы «сущность-связь».

1. В указанной модели мы имеем дело со следующими сущностями: Продукты, Поставщики, Города, Продажи. Следовательно, и в реляционной модели будут участвовать четыре отношения с такими же именами.
2. Далее в процессе задания конкретных типов данных для каждого атрибута отношений (**домены** могли быть определены уже на этапе **инфологического моделирования**) получаем отношения, приведенные на Рис. 2-27. Таблица 2-4 содержит типы данных для каждого атрибута.
3. Первичные ключи отношений, описанных в Таблица 2-4, помечены знаком √.

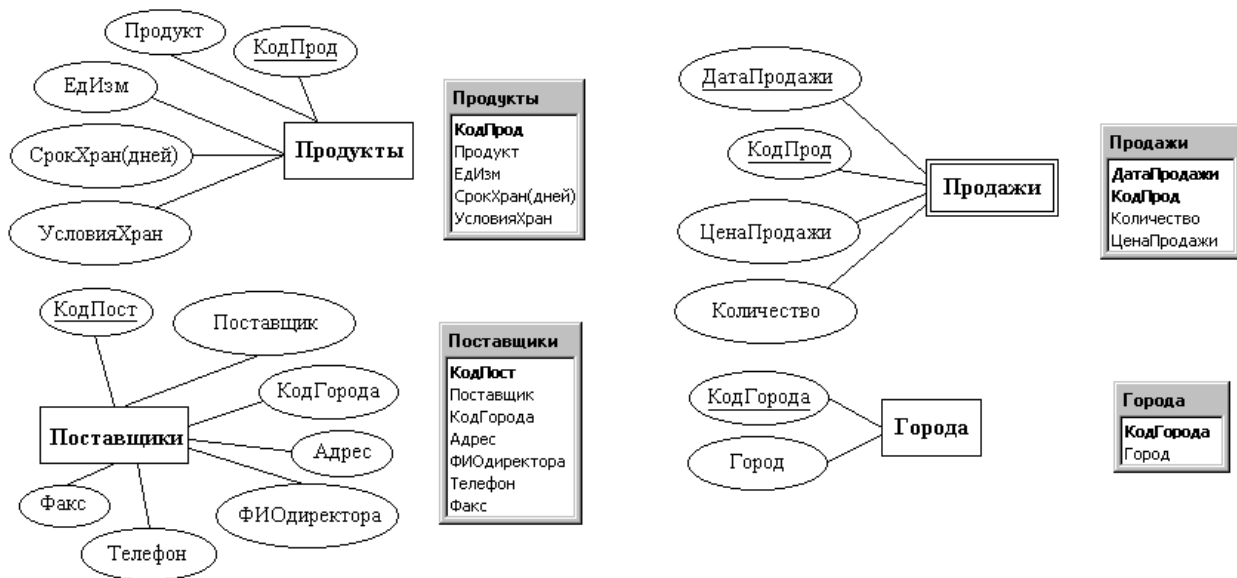


Рис. 2-27. Переход от сущностей ER-модели к отношениям реляционной модели

Таблица 2-4

Атрибут	Тип данных (СУБД Access)	Допустимость Null-значений	Первичный ключ	Внешний ключ
Отношение Продукты				
КодПрод	Целое	Нет	√	
Продукт	Текстовый (30)	Нет		
ЕдИзм	Текстовый (5)	Да		
СрокХран(дней)	Целое	Да		
УсловияХран	Текстовый (200)	Да		
Отношение Поставщики				
КодПост	Целое	Нет	√	
Поставщик	Текстовый (50)	Нет		
КодГорода	Целое	Да		√
Адрес	Текстовый (100)	Да		
ФИОДиректора	Текстовый (50)	Да		
Телефон	Текстовый (15)	Да		
Факс	Текстовый (15)	Да		
Отношение Продажи				
ДатаПродажи	Дата/время	Нет	√	
КодПрод	Целое	Нет		√
Количество	Одинарное с плавающей точкой	Да		
ЦенаПродажи	Денежный	Да		
Отношение Города				
КодГорода	Целое	Нет	√	
Город	Текстовый (30)	Нет		

- Степень связи между сущностями **Поставщики** и **Города** – N:1, поэтому первичный ключ **КодГорода** (сущности **Города**) должен войти в сущность **Поставщики** в качестве внешнего ключа (мы это сделали еще на этапе создания модели «сущность-связь»); степень связи между сущностями **Продажи** и **Продукты** – N:1, поэтому первичный ключ **КодПрод** (сущности **Продукты**) должен войти в сущность **Продажи** в качестве внешнего ключа (мы это сделали еще на этапе создания модели «сущность-связь»).
- Для внешнего ключа **КодГорода** (отношение **Поставщики**) устанавливаем свойство допустимость Null-значений: «Да», т.к. в модели «сущность-связь» сущность **Поставщики** имела необязательный класс принадлежности; для внешнего ключа **КодПост** (отношение **Поставщики**) устанавливаем свойство допустимость Null-значений: «Нет», поскольку этот внешний ключ входит в состав первичного ключа (Таблица 0-4).
- В нашем примере две связи имеют степень M:N. Это связи **Поставляют** и **Заказаны**. Следовательно, дополнительно появляются еще два отношения **Поставки** и **Заказы** соответственно. Таблица 0-5 содержит описание атрибутов этих отношений.

Таблица 2-5

Атрибут	Тип данных (СУБД Access)	Допустимость Null-значений	Первичный ключ	Внешний ключ
Отношение Поставки				
<i>ДатаПоставки</i>	Дата/время	Нет	√	
<i>КодПост</i>	Целое	Нет		√
<i>КодПрод</i>	Целое	Нет		√
<i>КоличествоП</i>	Одинарное с плавающей точкой	Да		
<i>ЦенаПоставки</i>	Денежный	Да		
<i>ДатаИзгот</i>	Дата/время	Да		
Отношение Заказы				
<i>ДатаЗаказа</i>	Дата/время	Нет	√	
<i>КодПост</i>	Целое	Нет		√
<i>КодПрод</i>	Целое	Нет		√
<i>КоличествоЗ</i>	Одинарное с плавающей точкой	Да		

Окончательный вариант реляционной модели (схемы БД) приведен на Рис. 2-28.

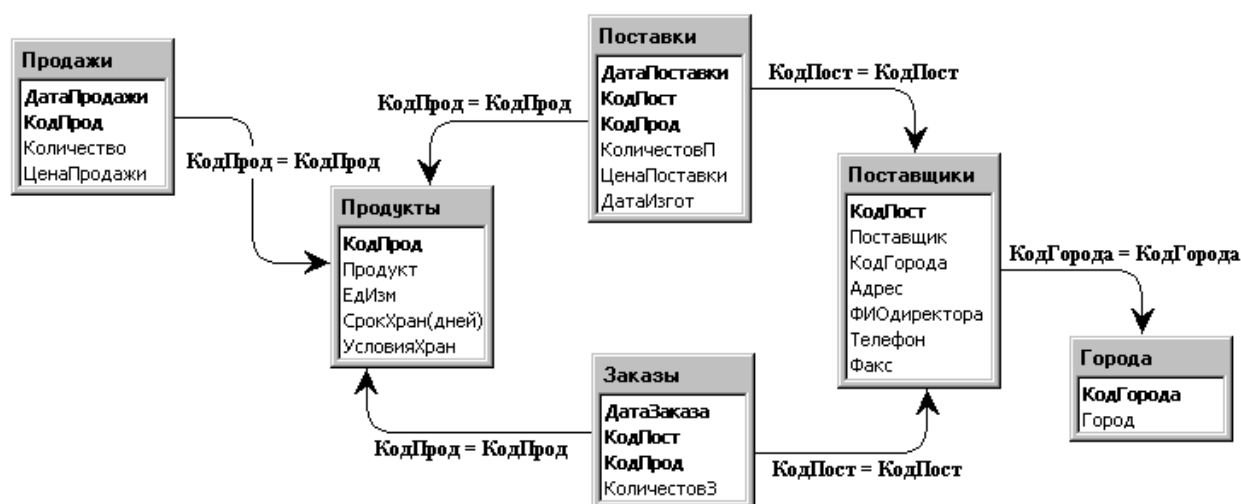


Рис. 2-28. Реляционная модель данных учета продажи продуктов в магазине

Глава 2.3. Проектирование реляционных баз данных на основе принципов нормализации

Следующим этапом жизненного цикла БД, который мы рассмотрим, будет этап **даталогического или логического проектирования БД**, приводящий к разработке схемы БД. **Схема БД** – совокупность схем отношений, адекватно моделирующих абстрактные объекты предметной области и семантические связи между этими объектами. Основой анализа корректности схемы являются анализ функциональных зависимостей между атрибутами отношений БД. Некоторые функциональные зависимости являются нежелательными из-за побочных эффектов и аномалий, возникающих при модификации БД.

На этапе **инфологического моделирования** была построена модель «сущность-связь», и с помощью алгоритма перехода к реляционной модели получена схема БД (Рис. 2-28), т.е. был начат этап логического проектирования. Для продолжения процесса проектирования необходимо проверить полученную схему БД на отсутствие избыточных функциональных зависимостей и при необходимости нормализовать схему БД.

Процесс нормализации может быть проведен уже к концептуальной модели «сущность-связь», тогда после перехода к реляционной модели получим нормализованную схему БД.

Построение схемы БД может быть выполнено двумя путями:

- *путем декомпозиции* (разбиения), когда исходное множество отношений, входящих в схему БД заменяется другим множеством отношений (их число при этом возрастает), являющихся проекциями исходных отношений
- *путем синтеза*, то есть путем компоновки из заданных исходных элементарных зависимостей между объектами предметной области схемы БД

Процесс проектирования с использованием декомпозиции представляет собой процесс последовательной нормализации схем отношений, при этом каждая последующая итерация соответствует нормальной форме более высокого уровня и обладает лучшими свойствами по сравнению с предыдущей.

Каждой нормальной форме соответствует определенный набор ограничений, и отношение находится в некоторой нормальной форме, если удовлетворяет свойственному ей набору ограничений.

2.3.1. Функциональные зависимости

В процессе нормализации рассматриваются различные функциональные зависимости. Функциональные зависимости определяют не текущее состояние БД, а все возможные ее состояния, то есть они отражают те связи между атрибутами, которые присущи реальному объекту, моделируемые в БД.

Функциональная зависимость. Атрибут Y некоторого отношения функционально зависит от X (атрибуты могут быть составными), если в любой момент времени каждому значению X соответствует одно значение Y . Функциональная зависимость обозначается $X \rightarrow Y$.

Избыточная функциональная зависимость – это зависимость, заключающая в себе такую информацию, которая может быть получена на основе других зависимостей, имеющих в базе данных.

Полная функциональная зависимость. Неключевой атрибут функционально полно зависит от составного ключа если он функционально зависит от всего ключа в целом, но не находится в функциональной зависимости от какого-либо из входящих в него атрибутов.

Транзитивная функциональная зависимость. Пусть X, Y, Z – три атрибута некоторого отношения. При этом $X \rightarrow Y$ и $Y \rightarrow Z$, но обратное соответствие отсутствует, т.е. $Z \not\rightarrow Y$ и $Y \not\rightarrow X$. Тогда Z транзитивно зависит от X .

Многозначная зависимость. Пусть X, Y, Z – три атрибута отношения R . В отношении R существует многозначная зависимость $R.X \twoheadrightarrow R.Y$ только в том случае, если множество значений Y , соответствующее паре значений X и Z , зависит только от X и не зависит от Z .

В общем случае необходимо проводить нормализацию к пятой нормальной форме (5НФ). На практике зачастую оказывается достаточным приведение к третьей нормальной форме (3НФ).

2.3.2. Первая нормальная форма

Первая нормальная форма (1НФ): отношение находится в 1НФ, если значения всех его атрибутов атомарны.

Иначе можно сказать, что в каждой позиции пересечения столбца и строки таблицы расположено в точности одно значение, а не набор значений. Отношения в 1НФ часто называются просто нормализованными отношениями.

Под атомарностью понимается степень структурирования и детализации информации в БД. Глубина структурирования определяется практической необходимостью при манипулировании данными. Примером является глубина структурирования адреса. Можно хранить в одном поле весь адрес (город, улица, дом, квартира). Данный атрибут будет атомарным, если нет необходимости манипулировать отдельными городами или улицами, в противном случае этот атрибут не является атомарным и необходимо его дальнейшее разбиение на отдельные атрибуты (город), (улица, дом, квартира).

Пример ненормализованного и нормализованного (в 1НФ) отношений приведен на Рис. 2-16.

2.3.3. Вторая нормальная форма

Вторая нормальная форма (2НФ): Отношение (таблица) находится во 2НФ, если оно находится в 1НФ, и каждый неключевой атрибут функционально полно зависит от всего ключа.

Если какой-либо атрибут зависит от части составного первичного ключа, то необходимо:

- создать новое отношение, атрибутами которого будут:
- часть составного ключа (первичный ключ нового отношения)
- атрибут, зависящий от нового ключа
- из исходного отношения исключить атрибут, включенный в новое отношение

То есть, если имеется отношение $R(\underline{k1, k2}, a1, a2)$, находящееся в 1НФ, где $k1, k2$ – составной первичный ключ, а $a1$ и $a2$ – неключевые атрибуты отношения R , и имеются функциональные зависимости:

$k1, k2 \rightarrow a1$ (атрибут $a1$ функционально полно зависит от первичного ключа $k1, k2$),

$k1 \rightarrow a2$ (атрибут $a2$ зависит от части первичного ключа $k1$, т.е. имеется неполная функциональная зависимость)

Для приведения отношения R к 2НФ, это отношение декомпозируется на два отношения: $R1(\underline{k1}, a2)$ и $R2(\underline{k1, k2}, a1)$. Отношения $R1$ и $R2$ будут иметь связь один-ко-многим по атрибуту $k1$.

Пример: Дано отношение *Поставки*(КодПоставщика, КодПродукта, ЕдиницаИзмерения). Поставщик может поставлять различные продукты, один и тот же продукт может поставляться разными поставщиками. Тогда первичным ключом отношения будут атрибуты **КодПоставщика** и **КодПродукта**. Значит, существует функциональная зависимость:

КодПоставщика, КодПродукта \rightarrow ЕдиницаИзмерения

С другой стороны, какой бы поставщик не поставит продукт, единица измерения от этого не изменится (например, цельное молоко измеряется литрами независимо от поставщика, а соль – килограммами). Т.е. существует еще одна функциональная зависимость (неключевой атрибут зависит от части первичного ключа):

КодПродукта \rightarrow ЕдиницаИзмерения

После исключения неполной функциональной зависимости получим отношения:

Поставки(КодПоставщика, КодПродукта) и *Продукты*(КодПродукта, ЕдиницаИзмерения)

При неполной функциональной зависимости возникают *аномалии*:

- *включения* (пока поставщиком не будет поставлен продукт, нельзя указать единицу измерения)
- *удаления* (исключение поставщика может привести к потере единицы измерения продукта)
- *обновления* (при изменении единицы измерения продукта, приходится менять данные везде, где встречается данный продукт)

Данные виды аномалий возникают при любой избыточной функциональной зависимости.

2.3.4. Третья нормальная форма

Третья нормальная форма (3НФ): Отношение находится в 3НФ, если оно находится во 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

То есть, если имеется отношение $R(\underline{k1}, a1, a2)$, находящееся в 2НФ, где $k1$ – первичный ключ, а $a1$ и $a2$ – неключевые атрибуты отношения R , и имеются функциональные зависимости:

$k1 \rightarrow a1$

$a1 \rightarrow a2$

тогда атрибут $a2$ транзитивно зависит от $k1$.

Для приведения отношения R к 3НФ, это отношение декомпозируется на два отношения: $R1(k1, a1)$ и $R2(a1, a2)$. Отношения $R1$ и $R2$ будут иметь связь многие-к-одному по атрибуту $a1$.

Пример: Дано отношение *Группы*(Группа, Специальность, Факультет) с первичным ключом **Группа**. Группа однозначно определяет специальность, а специальность однозначно определяет факультет. Т.е. существуют следующие функциональные зависимости:

Группа \rightarrow Специальность (и наоборот, Специальность \rightarrow Группа)

Специальность \rightarrow Факультет (Факультет \rightarrow Специальность)

После исключения транзитивной функциональной зависимости получим отношения:

Группы(Группа, Специальность) и *Специальности*(Специальность, Факультет)

2.3.5. Нормальная форма Бойса-Кодда

Ситуация, когда отношение будет находиться в 3НФ, но не в нормальной форме Бойса-Кодда (НФБК), возникает при условии, что отношение имеет два (или более) возможных ключа, которые являются составными и имеют общий атрибут. Заметим, что на практике такая ситуация встречается достаточно редко, для всех прочих отношений 3НФ и НФБК эквивалентны.

То есть, если имеется отношение $R(a_1, a_2, a_3, a_4)$, находящееся в 3НФ, где a_1, a_2 – возможный ключ, a_3 – возможный ключ, а a_4 – неключевой атрибут отношения R , и имеются функциональные зависимости:

$a_1 \rightarrow a_3$

$a_3 \rightarrow a_1$

$a_1, a_2 \rightarrow a_4$

$a_2, a_3 \rightarrow a_4$

Для приведения отношения R к НФБК, это отношение декомпозируется на два отношения:

$R_1(a_1, a_3)$ и $R_2(a_1, a_2, a_4)$

или $R_1(a_3, a_1)$ и $R_2(a_2, a_3, a_4)$.

Пример: Дано отношение *Экзамен*(№ зачетки, № паспорта, Дисциплина, Дата, Оценка).

Возможными ключами будут атрибуты: *№ зачетки, Дисциплина, Дата* и *№ паспорта, Дисциплина, Дата*. Имеются следующие функциональные зависимости:

№ зачетки, Дисциплина, Дата \rightarrow Оценка

№ паспорта, Дисциплина, Дата \rightarrow Оценка

№ зачетки \rightarrow *№ паспорта*

№ паспорта \rightarrow *№ зачетки*

После приведения отношения к НФБК могут быть получены отношения:

Студент(*№ зачетки*, *№ паспорта*), *Экзамен*(*№ зачетки, Дисциплина, Дата*, Оценка)

или

Студент(*№ паспорта, № зачетки*), *Экзамен*(*№ паспорта, Дисциплина, Дата*, Оценка)

2.3.6. Четвертая нормальная форма

Четвертая нормальная форма (4НФ): Отношение находится в 4НФ, если оно находится в НФБК, и в нем отсутствуют многозначные зависимости, не являющиеся функциональными зависимостями.

или

Отношение R находится в 4НФ в том случае, если в случае существования многозначной зависимости $A \twoheadrightarrow B$ все остальные атрибуты R функционально зависят от A .

То есть, если имеется отношение $R(a_1, a_2, a_3)$, находящееся в НФБК и имеются функциональные зависимости:

- зависимость множества значений атрибута **a_2** от множества значений атрибута **a_1** ($a_1 \twoheadrightarrow a_2$)
- зависимость множества значений атрибута **a_3** от множества значений ключевого атрибута **a_1** ($a_1 \twoheadrightarrow a_3$)

Для приведения отношения R к 4НФ, это отношение декомпозируется на два отношения: $R_1(a_1, a_2)$ и $R_2(a_1, a_3)$.

Пример: Дано отношение *Книги*(ISBN, Название, Автор, Область знаний). Книга имеет уникальный идентификатор ISBN, книга может быть написана коллективом авторов, книга может относиться к нескольким областям знаний (Таблица 2-6).

Таблица 2-6

ISBN	Название	Автор	Область знаний
5-123-12345-1	Информатика для экономистов	Иванов А.В.	Информатика
5-123-12345-1	Информатика для экономистов	Иванов А.В.	Экономика
5-123-12345-1	Информатика для экономистов	Петров С.М.	Информатика
5-123-12345-1	Информатика для экономистов	Петров С.М.	Экономика

Существуют следующие функциональные зависимости:

ISBN \rightarrow Название

ISBN \twoheadrightarrow Автор

ISBN \twoheadrightarrow Область знаний

После приведения отношения к 4НФ будут получены отношения:

Книги(ISBN, Название)

АвторыКниг(ISBN, Автор)

ОбластиЗнанийКниг(ISBN, Область знаний)

2.3.7. Пятая нормальная форма (нормальная форма проекции-соединения)

Зависимость соединения. Отношение $R(X, Y, \dots, Z)$ удовлетворяет зависимости соединения $*(X, Y, \dots, Z)$ в том и только в том случае, когда R восстанавливается без потерь путем соединения своих проекций на X, Y, \dots, Z . Где X, Y, \dots, Z – наборы атрибутов отношения R .

Пятая нормальная форма (5НФ): Отношение R находится в 5НФ в том и только в том случае, когда любая зависимость соединения в R следует из существования некоторого возможного ключа в R .

То есть, если имеется отношение $R(k_1, k_2, k_3)$, находящееся в 4НФ, где k_1, k_2, k_3 – составной первичный ключ, и имеется зависимость соединения:

$*(\{k_1, k_2\}, \{k_1, k_3\}, \{k_2, k_3\})$

Для приведения отношения R к 5НФ, это отношение декомпозируется на три отношения: $R_1(k_1, k_2)$, $R_2(k_1, k_3)$ и $R_3(k_2, k_3)$.

5НФ редко используется на практике. Очень тяжело определить само наличие зависимостей «проекции-соединения», потому что утверждение о наличии такой зависимости делается для всех возможных состояний БД, а не только для текущего экземпляра отношения R .

Модуль 3. Реализация реляционной модели в среде выбранной СУБД

Глава 3.1. Реализация реляционной модели в среде выбранной СУБД (MS Access)

В этой главе мы рассмотрим процесс реализации реляционной модели учета продажи продуктов в магазине, разработанной в параграфе 2.2.4, в СУБД MS Access. Создаваемую базу данных назовем БД «Магазин».

В СУБД MS Access терминология несколько отличается от терминологии реляционных моделей. Поскольку мы рассматриваем работу в конкретной СУБД, то и термины будем использовать соответствующие. Таблица 0-1 содержит соответствие терминов реляционной модели и СУБД Access.

Таблица 0-1

Термины реляционной модели	Термины СУБД MS Access
Отношение	Таблица
Атрибут	Поле
Кортеж	Запись
Связь	Связь

3.1.1. Создание таблиц

Правила именования таблиц и полей

Имена таблиц (и других объектов Access¹) и полей должны подчиняться следующим правилам:

- имя должно содержать не более 64 символов;
- имя может включать любую комбинацию букв, цифр, пробелов и специальных символов за исключением точки (.), восклицательного знака (!), надстрочного символа (^) и квадратных скобок ([]);
- не должно начинаться с символа пробела;
- не должно включать управляющие символы (с кодами ASCII от 0 до 31);
- не должно включать прямые кавычки (") в именах таблиц, представлений и хранимых процедур в проекте MS Access.

¹ Объектами базы данных MS Access являются таблицы, запросы, формы, отчеты, модули, макросы

Хотя пробелы внутри имен полей, элементов управления и объектов являются допустимыми, в большинстве примеров в документации MS Access имена полей записываются без пробелов. Пробелы в именах могут, при некоторых обстоятельствах, вызывать конфликты в программах Visual Basic.

Создание таблицы в режиме конструктора

Одним из способов создания таблиц в MS Access является создание таблиц в режиме конструктора. Последовательность действий при создании таблицы:

1. Запустить режим конструктора таблиц (Рис. 3-1)
2. В столбце «**Имя поля**» ввести имя очередного поля таблицы
3. В столбце «**Тип данных**» ввести или выбрать из списка тип данных поля:
 - **Текстовый** (Text) – для хранения текста, комбинации букв и цифр (тип данных по умолчанию), максимальный размер 255 символов
 - **Поле МЕМО** – для хранения длинного текста или чисел, например, примечания или описания., максимальный размер 65 535 символов
 - **Числовой** (Number) – для хранения числовых данных, используемых для математических вычислений, за исключением финансовых расчетов, размеры поля:

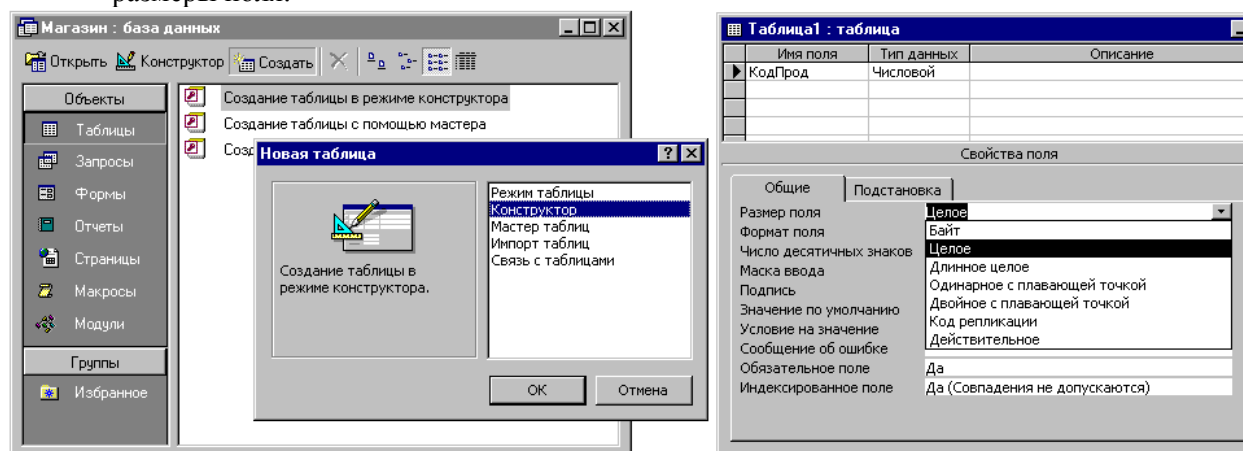


Рис. 3-1. Создание новой таблицы в режиме конструктора


- **Байт** (Byte) – 1 байт (от 0 до 255)
- **Целое** (Integer) – 2 байта (от -32 768 до 32 767)
- **Длинное целое** (Long Integer) – 4 байта (от -2 147 483 648 до 2 147 483 647)
- **Одинарное с плавающей точкой** (Single) – 4 байта (от -3.402823E38 до -1.401298E-45 для отрицательных чисел и от 1.401298E-45 до 3.402823E38 для положительных чисел)
- **Двойное с плавающей точкой** (Double) – 8 байт (от -1.79769313486231E308 до -4.94065645841247E-324 для отрицательных чисел и от 1.79769313486231E308 до 4.94065645841247E-324 для положительных чисел)
- **Код репликации** (Replication ID) – 16 байт (глобальный уникальный идентификатор – GUID)
- **Действительное** (Decimal) – 12 байт (числа с точностью до 28 знаков)
- **Дата/время** (Date/Time) – хранение дат и/или времени от 100 года до 9999 года (8 байт)
- **Денежный** (Currency) – хранение значения валют. Денежный тип используется для предотвращения округлений во время вычислений. Предполагает до 15 символов в целой части числа и 4 - в дробной. 8 байт
- **Счетчик** (AutoNumber) – для автоматической нумерации. Поле счетчика может генерировать три типа чисел: последовательно возрастающие на единицу, случайные числа, а также коды репликации. Созданный для записи номер уже не может быть удален или изменен (удалить можно только всю запись)
- **Логический** (Yes/No) – может принимать одно из двух значений: Да или Нет, Истина или Ложь, Вкл или Выкл – размер 1 бит
- **Поле объекта OLE** (OLE Object) – объекты (например, документы Microsoft Word, электронные таблицы Microsoft Excel, рисунки, звуки и другие двоичные данные),

созданные в других программах, использующих протокол OLE. Объекты могут быть связанными или внедренными в таблицу Microsoft Access.


- **Гиперссылка** (Hyperlink) – поле для хранения гиперссылок. Гиперссылка может иметь вид пути UNC, либо URL-адреса. Адрес гиперссылки может состоять максимум из 4-х частей, каждая из которых может содержать до 2048 символов
- **Мастер подстановок** (Lookup Wizard)– создает поле, позволяющее выбрать значение из другой таблицы или из списка значений, используя поле со списком. При выборе данного параметра в списке типов данных запускается мастер для автоматического определения этого поля

4. В столбце «**Описание**» ввести пояснение назначения поля (не обязательно)

5. В нижней части конструктора расположены две вкладки: **Общие** – для задания различных свойств поля, **Подстановка** – для организации подстановки значений в поле из списка или из другой таблицы. На вкладке **Общие** необходимо уточнить свойства поля:

- **Размер поля** – ввести или выбрать из списка размер поля
- **Формат поля** – ввести или выбрать из списка, для отображения данных в постоянном формате. Например, если свойство **Формат поля** для полей типа «Дата/время» установлено на **Краткий формат даты**, то все вводимые данные будут отображаться в следующем формате: 25.02.04. Если же пользователь базы данных введет число в виде 25-фев-04 (или в другом допустимом виде), то при сохранении записи формат даты будет преобразован в **Краткий формат даты**
- **Число десятичных знаков** – ввести или выбрать из списка (для числовых типов данных)
- **Маска ввода** – нажмите кнопку построителя , чтобы запустить мастер масок ввода, и следуйте инструкциям диалоговых окон мастера. Маска ввода используется для форматирования данных и управления вводимыми значениями. В основном маски ввода используются в текстовых полях и полях даты/времени, а также в числовых и денежных.
- **Значение по умолчанию** – можно ввести значение (при необходимости), которое будет присваиваться полю каждый раз при вводе новой записи
- **Условие на значение** – можно задать условие, которому должно удовлетворять вводимое значение поля например, срок хранения продуктов задается положительным числом, тогда условие на значение будет иметь вид: >0
- **Сообщение об ошибке** – сообщение, выдаваемое при нарушении условия на значение поля для условия из предыдущего примера, сообщение об ошибке будет: «Срок хранения не может быть отрицательным!»
Примечание: свойства поля «Условие на значение» и «Сообщение об ошибке» можно использовать для задания **семантической целостности БД**
- **Обязательное поле** – Да/Нет
Нет – допустимость неопределенных значений (Null-значений)
Да – недопустимость неопределенных значений
- **Индексированное поле** – индекс служит для ускорения поиска и сортировки полей, но замедляет обновление. Могут быть установлены значения:
Нет – нет индекса
Да (Допускаются совпадения) – индексированное поле, разрешены дублирующие значения поля в разных записях
Да (Совпадения не допускаются) – индексированное поле, значения поля уникальны для всей таблицы.
- Обычно устанавливается для первичного ключа, состоящего из одного поля (для составного первичного ключа уникальным является не значение одного поля, а сочетание значений всех полей ключа)

6. Повторить п.2-5 для создания всех полей таблицы

7. Установить первичный ключ таблицы. Для этого выделить нужное поле (или несколько полей для составного первичного ключа) и нажать кнопку 

8. Сохранить таблицу и выйти из режима конструктора

Создание таблиц БД «Магазин»

В соответствии с разработанной реляционной моделью (Рис. 3-28), необходимо создать все таблицы (отношения), входящие в эту модель.

Создавая реляционную модель, мы подробно рассмотрели, из каких атрибутов состоит каждое отношение, какие типы данных должны иметь атрибуты, какие атрибуты являются первичным ключом отношения, а также допустимость Null-значений для атрибутов (Таблица 3-4, Таблица 3-5). Используя данные реляционной модели, создадим в режиме конструктора таблицы: Продукты (Рис. 3-2), Поставщики, Поставки (Рис. 3-3), Заказы, Продажи, Города.

Имя поля	Тип данных	Описание
КодПрод	Числовой	код продукта
Продукт	Текстовый	наименование продукта
ЕдИзм	Текстовый	единицы измерения (кг, л, шт)
СрокХран(дней)	Числовой	срок хранения продукта (в днях)
УсловияХран	Текстовый	условия хранения (температура, влажность ...)

Свойства поля

Общие | Подстановка

Размер поля: Целое
 Формат поля:
 Число десятичных знаков: Авто
 Маска ввода:
 Подпись:
 Значение по умолчанию:
 Условие на значение:
 Сообщение об ошибке:
 Обязательное поле: Да
 Индексированное поле: Да (Совпадения не допускаются)

Имя поля может состоять из 64 знаков с учетом пробелов. Для справки по именам полей нажмите клавишу F1.

Рис. 3-2. Создание таблицы Продукты в режиме конструктора

Имя поля	Тип данных	Описание
ДатаПоставки	Дата/время	
КодПост	Числовой	
КодПрод	Числовой	
КоличествоП	Числовой	
ЦенаПоставки	Денежный	
ДатаИзгот	Дата/время	

Свойства поля

Общие | Подстановка

Формат поля: Краткий формат даты
 Маска ввода: 99.99.99
 Подпись:
 Значение по умолчанию:
 Условие на значение: >#01.01.95#
 Сообщение об ошибке: Дата поставки должна быть позднее 1 яне
 Обязательное поле: Да
 Индексированное поле: Нет

Выражение, накладывающее ограничение на значения, которые вводятся в данное поле. Чтобы получить справку по условиям на значения, нажмите клавишу F1.



Рис. 3-3. Создание таблицы Поставки в режиме конструктора

3.1.2. Построение схемы данных. Задание ограничений целостности

После того, как все таблицы БД созданы, необходимо установить связи между таблицами и задать ограничения **ссылочной целостности** в окне схемы данных.

Построение схемы данных

Последовательность действий для построения схемы данных:

1. Открыть окно схемы данных, нажав кнопку  на панели инструментов (или выбрать пункты меню *Сервис, Схема данных...*)
2. Открыть окно для добавления таблиц на схему данных (Рис. 3-4), нажав кнопку  на панели инструментов (или выбрать пункты меню *Связи, Добавить таблицу...*)

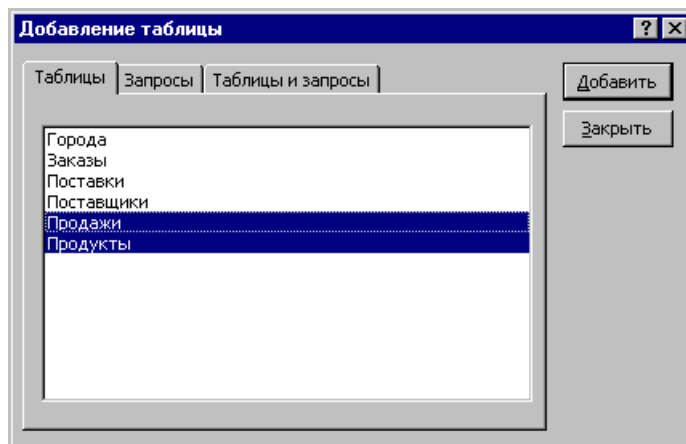


Рис. 3-4. Диалоговое окно «Добавление таблицы»

3. Добавить таблицы на схему данных, выделив нужные таблицы в списке таблиц и нажав кнопку Добавить, закрыть окно «Добавление таблицы»
4. Установить связи и правила ссылочной целостности между таблицами
 - Связь устанавливается между полями таблиц (между первичным ключом основной таблицы и внешним ключом подчиненной таблицы). Для этого перетащите мышкой поле первичного ключа из основной таблицы на поле внешнего ключа подчиненной таблицы (или наоборот внешний ключ на первичный), в результате будет открыто окно изменения связей (Рис. 3-5).

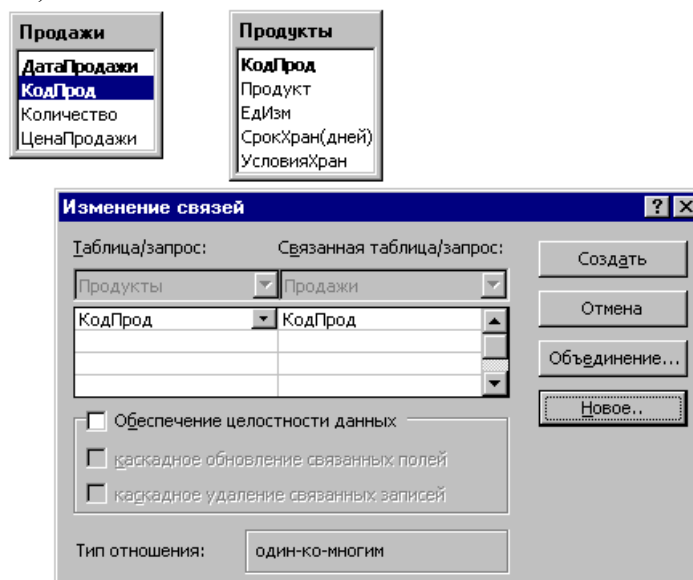


Рис. 3-5. Создание связей и установка ссылочной целостности

- Задайте правила ссылочной целостности путем установки соответствующих флагов (Рис. 3-6):
- обеспечение целостности данных
- каскадное обновление связанных полей
- каскадное удаление связанных полей

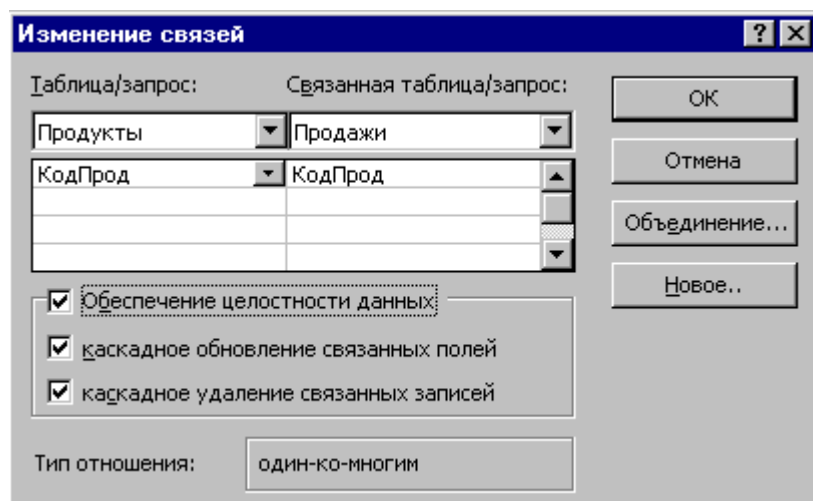


Рис. 3-6. Диалоговое окно установки обеспечения ссылочной целостности

- Для обеспечения режима ОГРАНИЧЕНИЯ обновления (и/или удаления) необходимо установить флаг:
 - обеспечение целостности данных
- Для обеспечения режима КАСКАДИРОВАНИЯ обновления (и/или удаления) необходимо установить флаги:
 - обеспечение целостности данных
 - каскадное обновление (и/или удаление) связанных полей
- Для подтверждения установленных режимов целостности и создания связи между таблицами нажмите кнопку **ОК** (Рис. 3-7)



Рис. 3-7. Связь N:1 между таблицами

5. Повторить п.2-4 для создания связей между остальными таблицами

Построение схемы данных БД «Магазин»

Рассмотрим, какие правила ссылочной целостности необходимо установить в БД «Магазин» (Таблица 3-2), реляционная модель которой приведена на Рис. 3-28.

Таблица 3-2

Связь между таблицами	Обновление	Удаление
Продукты – Продажи	Каскадировать	Каскадировать
Продукты – Поставки	Каскадировать	Ограничить
Продукты – Заказы	Каскадировать	Каскадировать
Поставщики – Поставки	Каскадировать	Ограничить
Поставщики – Заказы	Каскадировать	Каскадировать
Города – Поставщики	Каскадировать	Ограничить

Окончательная схема данных БД «Магазин» приведена на Рис. 3-8.

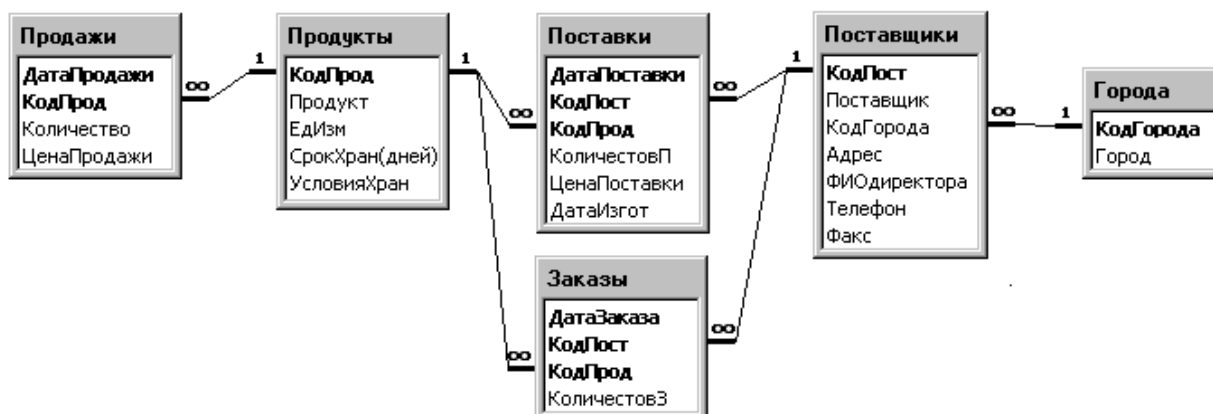


Рис. 3-8. Схема данных БД «Магазин»

Глава 3.2. Табличный язык запросов QBE

Табличный язык запросов QBE (сокращение от **Q**uery-**B**y-**E**xample или Запросы по образцу), наряду с языком **SQL**, используется для создания различных запросов к **реляционным БД**. Язык QBE является более наглядным и простым для понимания по сравнению с SQL, хотя и более ограниченным в возможностях. Поэтому мы начнем изучение построения запросов именно с языка QBE (тем более, что СУБД MS Access поддерживает автоматическое преобразование QBE-запросов в формат SQL).

Для иллюстрации работы с языком QBE рассмотрим работу продуктового магазина. **Схема данных** БД «Магазин» приведена на Рис. 3-8.

Для иллюстрации работы запросов мы будем использовать данные, приведенные в нижеследующих таблицах (Таблица 3-3 – Таблица 3-9).

Таблица 3-3. Продукты

КодПрод	Продукт	ЕдИзм	СрокХран(дней)	УсловияХран
1	Говядина	кг	30	
2	Судак	кг	30	
3	Масло	л	60	
4	Майонез	кг	90	
5	Яйца	шт	30	
6	Сметана	кг	10	
7	Молоко	л	3	
8	Творог	кг	3	
9	Морковь	кг	30	
10	Лук	кг	90	
11	Помидоры	кг	30	
12	Укроп	кг	10	
13	Рис	кг	300	
14	Мука	кг	300	
15	Яблоки	кг	90	
16	Сахар	кг	730	
17	Кофе	кг	300	
18	Сливки	л	3	
19	Сок яблочный	л	90	
20	Огурцы	кг	30	
21	Соль	кг	730	в сухом месте

Таблица 3-4. Поставщики

КодПост	Поставщик	КодГорода	Адрес	ФИОдиректора	Телефон	Факс
1	ОАО "Приморье"	3	Русская, 3	Иванов П.Л.	223344	223345
2	ПБОЮЛ Свиридова А.Н.	4	Садовая, 27	Свиридова А.Н.	265493	
3	Овощебаза №8	1	Новая, 17	Тимофеева Н.П.	94238	
4	ООО "ДВ продукты"	3	Фокина, 3	Виноградов О.Н.	222222	
5	Уссурийский МЖК	5	Песчаная, 19	Борисов Л.Л.	26748	
6	ПБОЮЛ Алексеев И.В.	3	Калинина, 8	Алексеев И.В.	284732	
7	ПБОЮЛ Авдеев С.С.	6	Ленинская, 15	Авдеев С.С.	32212	
8	ООО "Урожай"	2	Строительная, 64	Ясенева Т.И.	45789	
9	ООО "Выпечка"	3	Снеговая, 7	Демьянова В.М.	453212	

Таблица 3-5. Заказы

ДатаЗаказа	КодПост	КодПрод	КоличествоЗ
01.03.03	4	5	70
01.03.03	5	3	250
01.03.03	5	4	50
03.03.03	7	16	50
05.03.03	1	15	170
06.03.03	1	13	40
06.03.03	2	6	80
09.03.03	6	12	10
09.03.03	7	1	70
10.03.03	2	11	90
18.03.03	2	9	20
01.01.04	1	9	50
01.01.04	2	1	300
01.01.04	2	3	70
01.01.04	2	5	100
01.01.04	4	10	130
01.01.04	4	16	200
01.01.04	5	4	50
02.01.04	3	12	20
02.01.04	4	14	100
03.01.04	2	8	20
04.01.04	4	17	5
06.01.04	4	13	150
07.01.04	6	11	40
08.01.04	7	7	15
09.01.04	3	15	200
12.01.04	6	2	50
15.01.04	4	2	10
23.01.04	7	6	140
02.02.04	3	9	75
05.02.04	2	14	70
09.02.04	7	8	150
12.02.04	1	11	50
12.02.04	1	12	10
14.02.04	4	7	200
16.02.04	6	10	90
16.02.04	8	11	100

Таблица 3-6. Поставки

ДатаПоставки	КодПост	КодПрод	КоличествоП	ЦенаПоставки	ДатаИзгот
02.03.03	4	5	70	2,00р.	
02.03.03	5	3	250	40,00р.	
02.03.03	5	4	50	20,00р.	
04.03.03	7	16	50	15,00р.	
07.03.03	1	13	40	4,00р.	
07.03.03	1	15	170	13,00р.	
07.03.03	2	6	80	39,00р.	
10.03.03	6	12	10	80,00р.	
10.03.03	7	1	70	60,00р.	
11.03.03	2	11	90	8,00р.	
19.03.03	2	9	20	3,00р.	
02.01.04	1	9	50	4,00р.	
02.01.04	2	1	300	65,00р.	
02.01.04	2	3	70	50,00р.	
02.01.04	4	10	130	5,00р.	
02.01.04	4	16	200	17,00р.	
02.01.04	5	4	50	30,00р.	
03.01.04	3	12	20	80,00р.	
03.01.04	4	14	100	20,00р.	
04.01.04	2	8	20	30,00р.	
05.01.04	4	17	5	290,00р.	
07.01.04	4	13	150	8,00р.	
08.01.04	6	11	40	10,00р.	
09.01.04	7	7	15	6,00р.	
10.01.04	3	15	200	15,00р.	
14.01.04	6	2	50	85,00р.	
16.01.04	4	2	10	85,00р.	
25.01.04	7	6	140	44,00р.	
04.02.04	3	9	75	4,00р.	
06.02.04	2	14	70	20,00р.	
10.02.04	7	8	150	30,00р.	
13.02.04	1	11	50	10,00р.	
13.02.04	1	12	10	80,00р.	
15.02.04	4	7	200	8,00р.	
17.02.04	6	10	90	5,00р.	
17.02.04	8	11	100	10,00р.	

Таблица 3-7. Города

КодГорода	Город
1	Арсеньев
2	Большой Камень
3	Владивосток
4	Находка
5	Уссурийск
6	Фокино
7	Петропавловск-Камчатский

Таблица 3-8. Продажи

ДатаПродажи	КодПрод	Количество	ЦенаПродажи
25.03.03	3	15	60,00р.
25.03.03	7	15	22,00р.
25.03.03	10	15	10,00р.
02.01.04	1	20	75,00р.
02.01.04	4	10	38,00р.
02.01.04	6	40	60,00р.
02.01.04	8	10	60,00р.
02.01.04	12	40	100,00р.
02.01.04	13	30	10,00р.
02.01.04	15	10	28,00р.
02.01.04	16	10	20,00р.
02.01.04	19	5	25,00р.
18.02.04	4	4	38,00р.
18.02.04	6	12	60,00р.
18.02.04	8	11	60,00р.
18.02.04	9	10	10,00р.
18.02.04	12	2	100,00р.
18.02.04	18	8	50,00р.

Таблица 3-9. НовыеПродукты

КодПрод	Продукт	ЕдИзм	СрокХран(дней)	УсловияХран
21	Соль	Кг	730	в сухом месте
31	Колбаса "Докторская"	Кг	7	2-5 град С
32	Колбаса "Любительская"	Кг	7	2-5 град С

3.2.1. Запросы с использованием одной таблицы

1. Выбрать полную информацию о продуктах с сортировкой по алфавиту (Рис. 3-9). Обратите внимание, что для поля Продукт знак вывода на экран не установлен, чтобы это поле не выводилось в запросе дважды

а)

Продукты	
*	
КодПрод	
Продукт	
ЕдИзм	
СрокХран(дней)	
УсловияХран	

Поле:	Продукты.*	Продукт
Имя таблицы:	Продукты	Продукты
Сортировка:		по возрастанию
Вывод на экран:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Условие отбора:		
или:		

б)

КодПрод	Продукт	ЕдИзм	СрокХран(дней)	УсловияХран
1	Говядина	кг	30	
17	Кофе	кг	300	
10	Лук	кг	90	
4	Майонез	кг	90	
3	Масло	л	60	
7	Молоко	л	3	
9	Морковь	кг	30	
14	Мука	кг	300	
20	Огурцы	кг	30	
11	Помидоры	кг	30	
13	Рис	кг	300	
16	Сахар	кг	730	
18	Сливки	л	3	
6	Сметана	кг	10	
19	Сок яблочн	л	90	
21	Соль	кг	730	в сухом месте

Рис. 3-9. Выборка всех продуктов с сортировкой по алфавиту

2. Исключение дубликатов (выдать перечень проданных продуктов без повторений)
 этот запрос является аналогом операции проекции реляционной алгебры;
 для исключения дубликатов необходимо установить свойство **Уникальные значения: Да** (Рис. 3-10)

а)

Продажи	
*	
ДатаПродажи	
КодПрод	
Количество	
ЦенаПродажи	

Поле:	КодПрод
Имя таблицы:	Продажи
Сортировка:	
Вывод на экран:	<input checked="" type="checkbox"/>
Условие отбора:	
или:	

б)

Свойства запроса	
Общие	
Описание	
Вывод всех полей	Нет
Набор значений	Все
Уникальные значения	Да
Уникальные записи	Нет
При запуске предоставляются права ..	Пользователя
База данных-источник	(текущая)
Строка подключения-источник	
Блокировка записей	Отсутствует
Тип набора записей	Динамический наб
Время ожидания ODBC	60
Фильтр	
Порядок сортировки	

КодПрод
1
3
4
6
7
8
9
10
12
13
15
16
18
19

Рис. 3-10. Исключение дубликатов

3. Задание условия отбора (выбрать поставщиков, в названии которых есть сокращение ООО). В запросе используется предикат Like для отбора части значения поля, символ * в данном контексте соответствует любому количеству цифр или символов

а)

Поставщики	
*	
КодПост	
Поставщик	
КодГорода	
Адрес	

Поле:	Поставщик
Имя таблицы:	Поставщики
Сортировка:	
Вывод на экран:	<input checked="" type="checkbox"/>
Условие отбора:	Like "*ООО*"
или:	

б)

Поставщик
ООО "ДВ продукты"
ООО "Урожай"
ООО "Выпечка"

4. *Задание нескольких условий отбора* (выбрать поставщиков, в названии которых есть сокращение ООО, ОАО или ЗАО). Условия отбора, объединенные оператором ИЛИ, можно записывать каждое в своей строке условия отбора (Рис. 3-11, а) или в виде одного общего условия (Рис. 3-11, б), в любом случае результат запроса будет одинаковым (Рис. 3-11, в)
5. *Задание нескольких условий отбора* (выбрать поставщиков, в названии которых есть сокращение ООО и находящихся во Владивостоке (т.е. у которых КодГорода = 3)). Условия отбора, объединенные оператором И, должны быть записаны в одной строке условия отбора. Если условия относятся к одному и тому же полю, то условие можно записать в виде общего выражения (например, для выбора заказов за период с 1.02.04 по 31.03.04 условие можно записать так: >=#01.02.04# And <=#31.03.04#)

а) **Поставщики**

*
КодПост
Поставщик
КодГорода
Адрес

Поле: Поставщик
Имя таблицы: Поставщики
Сортировка:
Вывод на экран: ☒
Условие отбора: Like "*ООО*"
или: Like "*ОАО*"
Like "*ЗАО*"

б) **Поставщики**

*
КодПост
Поставщик
КодГорода
Адрес

Поле: Поставщик
Имя таблицы: Поставщики
Сортировка:
Вывод на экран: ☒
Условие отбора: Like "*ООО*" Or Like "*ОАО*" Or Like "*ЗАО*"
или:

в) **Поставщик**

ОАО "Приморье"
ООО "ДВ продукты"
ООО "Урожай"
ООО "Выпечка"

Рис. 3-11. Задание условий ИЛИ

а) **Поставщики**

*
КодПост
Поставщик
КодГорода
Адрес

Поле: Поставщик
Имя таблицы: Поставщики
Сортировка:
Вывод на экран: ☒
Условие отбора: Like "*ООО*"
или:

б) **Поставщик** **КодГорода**

ООО "ДВ продукты"	3
ООО "Выпечка"	3

Поле: Поставщик КодГорода
Имя таблицы: Поставщики Поставщики
Сортировка:
Вывод на экран: ☒ ☒
Условие отбора: Like "*ООО*" 3
или:

Рис. 3-12. Задание условий И

6. *Задание диапазонов в запросах* (выбрать поставки продуктов, цена поставки которых попадает в диапазон от 15 до 50 руб. включительно). Диапазон можно задать, используя конструкцию Between ... And ... (находится в интервале от ... до ...) (Рис. 3-13, а), или используя операторы сравнения >=, >, <, <= (Рис. 3-13, б).

а) **Поставки**

*
ДатаПоставки
КодПост
КодПрод
КоличествоП
ЦенаПоставки
ДатаИзгот

Поле: Поставки.* ЦенаПоставки
Имя таблицы: Поставки Поставки
Сортировка:
Вывод на экран: ☒ ☐
Условие отбора: Between 15 And 50
или:

б) **Поставки**

*
ДатаПоставки
КодПост
КодПрод
КоличествоП
ЦенаПоставки
ДатаИзгот

Поле: Поставки.* ЦенаПоставки
Имя таблицы: Поставки Поставки
Сортировка:
Вывод на экран: ☒ ☐
Условие отбора: >=15 And <=50
или:

в)

ДатаПоставки	КодПост	КодПрод	КоличествоП	ЦенаПоставки	ДатаИзгот
02.01.04	2	3	70	50,00р.	
02.01.04	4	16	200	17,00р.	
02.01.04	5	4	50	30,00р.	
03.01.04	4	14	100	20,00р.	
04.01.04	2	8	20	30,00р.	
10.01.04	3	15	200	15,00р.	
25.01.04	7	6	140	44,00р.	
06.02.04	2	14	70	20,00р.	
10.02.04	7	8	150	30,00р.	
02.03.03	5	4	50	20,00р.	
02.03.03	5	3	250	40,00р.	
04.03.03	7	16	50	15,00р.	
07.03.03	2	6	80	39,00р.	

Рис. 3-13. Условия для диапазонов значений

3.2.2. Возможности совместной обработки нескольких таблиц, связывание таблиц

1. **Декартово произведение.** Декартово произведение может потребоваться для получения всех сочетаний значений таблиц. Получим все возможные сочетания поставщиков и продуктов, т.е. ВСЕ поставщики поставляют ВСЕ продукты. Для получения декартова произведения двух таблиц необходимо разместить в запросе две несвязанные таблицы (Рис. 3-14, а). На Рис. 3-14, б приведены только первые записи результата, т.к. количество результирующих записей при декартовом произведении составляет $n*m$, где n и m – количество записей исходных таблиц (в нашем случае 189).

а) **Поставщики**

*
КодПост
Поставщик
КодГорода
Адрес

Продукты

*
КодПрод
Продукт
ЕдИзм
СрокХран(дней)

Поле: Поставщики.* Продукты.*
Имя таблицы: Поставщики Продукты
Сортировка:
Вывод на экран: ☒ ☒
Условие отбора:
или:

б)

КодПост	Поставщик	КодГоро	Адрес	ФИОдиректора	Телефон	Факс	КодПрод	Продукт	ЕдИзм	СрокХран	УсловияХран
1	ОАО "Приморье"	3	Русская, 3	Иванов П.Л.	223344	223345	21	Соль	кг	730	в сухом месте
2	ПБОЮЛ Свиридова А.Н.	4	Садовая, 27	Свиридова А.Н.	265493		21	Соль	кг	730	в сухом месте
3	Овощебаза №8	1	Новая, 17	Тимофеева Н.П.	94238		21	Соль	кг	730	в сухом месте
4	ООО "ДВ продукты"	3	Фокина, 3	Виноградов О.Н.	222222		21	Соль	кг	730	в сухом месте
5	Уссурийский МЖК	5	Песчаная, 19	Борисов Л.Л.	26748		21	Соль	кг	730	в сухом месте
6	ПБОЮЛ Алексеев И.В.	3	Калинина, 8	Алексеев И.В.	284732		21	Соль	кг	730	в сухом месте
7	ПБОЮЛ Авдеев С.С.	6	Ленинская, 15	Авдеев С.С.	32212		21	Соль	кг	730	в сухом месте
8	ООО "Урожай"	2	Строительная, 64	Ясенева Т.И.	45789		21	Соль	кг	730	в сухом месте
9	ООО "Выпечка"	3	Снеговая, 7	Демьянова В.М.	453212		21	Соль	кг	730	в сухом месте
1	ОАО "Приморье"	3	Русская, 3	Иванов П.Л.	223344	223345	1	Говядина	кг	30	
2	ПБОЮЛ Свиридова А.Н.	4	Садовая, 27	Свиридова А.Н.	265493		1	Говядина	кг	30	
3	Овощебаза №8	1	Новая, 17	Тимофеева Н.П.	94238		1	Говядина	кг	30	
4	ООО "ДВ продукты"	3	Фокина, 3	Виноградов О.Н.	222222		1	Говядина	кг	30	

Рис. 3-14. Декартово произведение

2. *Естественное соединение.* Получить список продаж с характеристиками продуктов. Поскольку продажи продуктов хранятся в таблице Продажи, а информация о продуктах – в таблице Продукты, то для получения необходимого результата в запросе нужно использовать обе таблицы, связанные по полю КодПрод (Рис. 3-15)

а)

Поле:	Продажи.*	Продукт	ЕдИзм	СрокХран(дней)	УсловияХран
Имя таблицы:	Продажи	Продукты	Продукты	Продукты	Продукты
Сортировка:					
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:					
или:					

б)

ДатаПродажи	КодПрод	Количество	ЦенаПродажи	Продукт	ЕдИзм	СрокХран(дней)	УсловияХран
25.03.03	10	15	10,00р.	Лук	кг	90	
25.03.03	3	15	60,00р.	Масло	л	60	
25.03.03	7	15	22,00р.	Молоко	л	3	
02.01.04	13	30	10,00р.	Рис	кг	300	
02.01.04	4	10	38,00р.	Майонез	кг	90	
• • •							
18.02.04	4	4	38,00р.	Майонез	кг	90	
18.02.04	8	11	60,00р.	Творог	кг	3	
18.02.04	6	12	60,00р.	Сметана	кг	10	

Рис. 3-15. Естественное соединение

3. *Условное соединение.* Получить названия и вес продуктов, проданных 2 января 2004г. В отличие от предыдущего запроса здесь добавляется условие отбора (Рис. 3-16)

а)

Поле:	Продажи.*	Продукт	Количество	ЕдИзм
Имя таблицы:	Продажи	Продукты	Продажи	Продукты
Сортировка:				
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:	#02.01.04#			
или:				

б)

ДатаПродажи	Продукт	Количество	ЕдИзм
02.01.04	Говядина	20	кг
02.01.04	Майонез	10	кг
02.01.04	Сметана	40	кг
02.01.04	Творог	10	кг
02.01.04	Укроп	40	кг
02.01.04	Рис	30	кг
02.01.04	Яблоки	10	кг
02.01.04	Сахар	10	кг
02.01.04	Сок яблочный	5	л

Рис. 3-16. Условное соединение

4. *Внешнее соединение.* Получить поставщиков, ни разу не поставивших продукты (Рис. 3-18). Т.е. таких поставщиков, которые есть в таблице **Поставщики** и, которых нет в таблице **Поставки**. Для установки этого вида соединения выберите пункт меню **Вид/ Параметры объединения** (Рис. 3-17) и в появившемся диалоговом окне установите нужный вид соединения (Рис. 3-18). Для поставщиков, у которых не оказалось соответствующей записи в таблице **Поставки**, поле **КодПост** (из таблицы **Поставки**) будет принимать неопределенное значение (Null-значение). Для выбора таких поставщиков необходимо указать условие отбора **Is Null** в соответствующем поле.

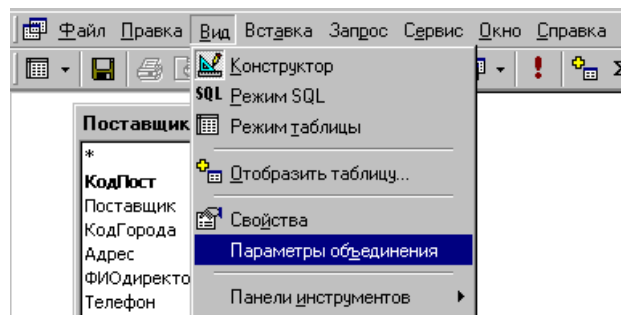


Рис. 3-17. Пункт меню «Параметры объединения»

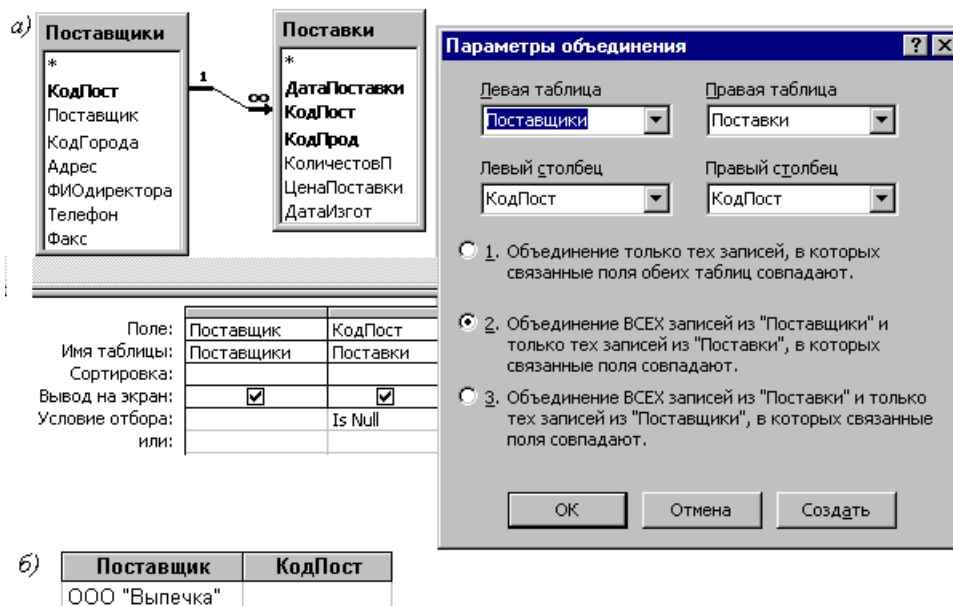
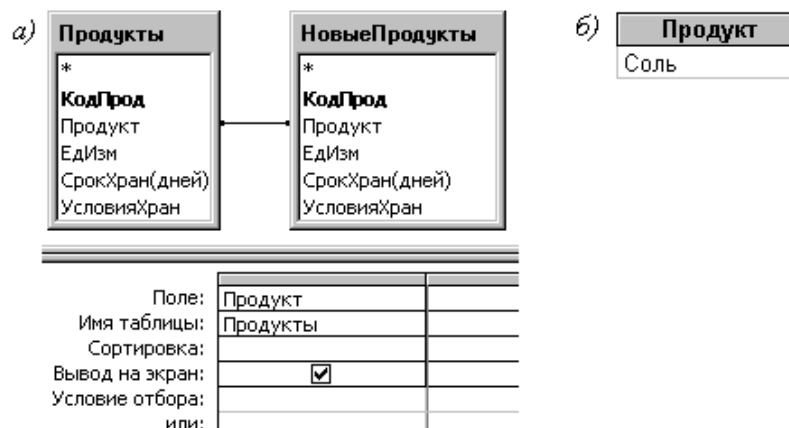


Рис. 3-18. Левое внешнее соединение

5. *Пересечение таблиц.* Найти продукты, информация о которых есть в обеих таблицах Продукты и НовыеПродукты



6. *Соединение таблицы со своей копией.* Выбрать из таблицы Продукты дубликаты по названию продуктов. Размещаемая в запросе копия таблицы получает новое название с префиксом _n, где n – номер копии таблицы в запросе. Необходимо найти дубликаты по названию продукта, т.е. такие записи, которые имеют одинаковые названия (связь между таблицами по полю Продукт) и разные коды продукта.



Рис. 3-19. Поиск дубликатов

3.2.3. Вычисляемые поля

1. **Переименование полей.** Иногда для удобства работы требуется переименовать некоторые поля в запросе (например, при наличии одноименных полей в разных таблицах). При переименовании полей используется следующий синтаксис:
НовоеИмя: СтароеИмя (Рис. 3-20)

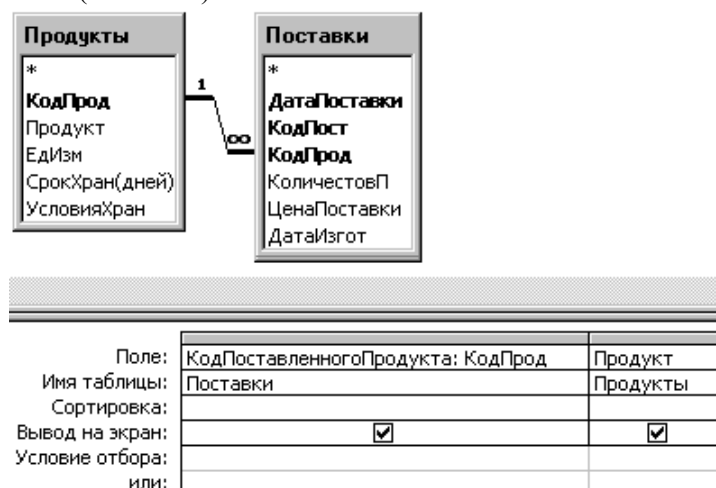



Рис. 3-20. Переименование полей

2. **Вычисляемые поля** Вычислить дату окончания срока хранения продуктов (Рис. 3-21)
 В БД «Магазин» хранятся: дата изготовления продукта (поле *ДатаИзгот* таблицы *Поставки*) и срок хранения продукта (в днях) (поле *СрокХран(дней)* таблицы *Продукты*). Если к дате прибавить количество дней, то получится новая дата. В нашем случае
 а) **ДатаОкончХран: [ДатаИзгот] + [СрокХран(дней)]** или

- б) **ДатаОкончХран: [Поставки].[ДатаИзгот] + [Продукты].[СрокХран(дней)]**

Заметим, что:

- в вычисляемых полях имена полей указываются в квадратных скобках - [ИмяПоля] (скобки можно опустить, если в имени поля используются только буквы и/или цифры и нет ссылки на таблицу)
- на имена полей можно ссылаться
 - только по имени, если нет одноименных полей в других таблицах запроса: [ИмяПоля]
 - по имени таблицы и имени поля: [ИмяТаблицы].[ИмяПоля]
- для записи вычисляемых полей можно использовать построитель выражений. Для вызова построителя выражений нажмите кнопку  на панели инструментов.

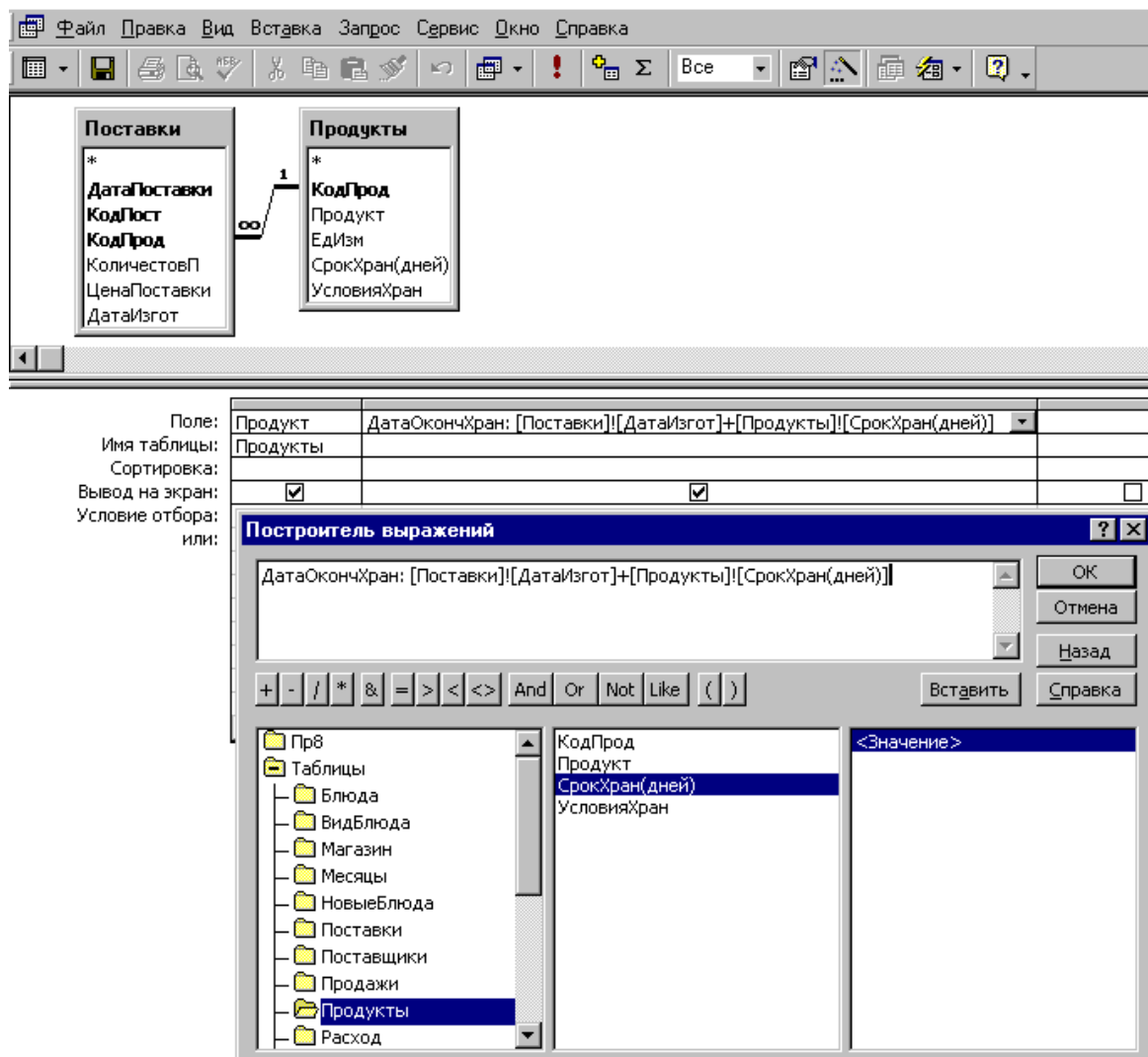


Рис. 3-21. Вычисляемые поля в запросе

3.2.4. Возможности группировки данных. Использование агрегатных функций


Довольно часто требуется не просто сделать выборку каких-либо данных, а подвести некоторые итоги. Например, посчитать, на какую сумму было продано продуктов за текущий день, или найти общее количество продуктов в заказе, или найти максимальное количество продукта, проданного за один раз и т.п. В таких случаях необходимо использовать запросы с группировкой данных и агрегатными функциями.

Наиболее часто используются следующие агрегатные функции:

- Count – подсчет количества записей, возвращаемых запросом
- Sum – вычисление суммы набора значений, содержащихся в заданном поле запроса
- Avg – вычисление арифметического среднего набора чисел, содержащихся в указанном поле запроса
- Max – вычисление максимального значения из набора значений, содержащихся в указанном поле запроса
- Min – вычисление минимального значения из набора значений, содержащихся в указанном поле запроса

Группировка используется для тех полей, по значениям которых нужно сгруппировать записи таблицы, а затем произвести нужные вычисления для каждой группы значений. Например, для расчета количества (номенклатуры) проданных продуктов за каждый день необходимо:

- Сгруппировать данные продажи продуктов по дате продажи
- Задать функцию для подсчета количества продуктов (Count)

Для подключения групповых операций в запросе на выборку нажмите кнопку  или выберите пункты меню *Вид, Групповые операции*. В бланке запроса появится дополнительная строка «Групповая операция» (Рис. 3-22).

Рассмотрим несколько примеров:

1. Рассчитать количество проданных продуктов за каждый день (Рис. 3-22, а)
Обратите внимание, что поле, для которого установлена агрегатная функция, автоматически получает новое имя (Рис. 3-22, б).

а)

Продажи	Продукты
*	*
ДатаПродаж	КодПрод
КодПрод	Продукт
Количество	ЕдИзм
ЦенаПродажи	СрокХран(дней)
	УсловияХран

б)

ДатаПродажи	Count-Продукт
25.03.03	3
02.01.04	9
18.02.04	6

Поле:	ДатаПродажи	Продукт
Имя таблицы:	Продажи	Продукты
Групповая операция:	Группировка	Count
Сортировка:		
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:		
или:		

Рис. 3-22. Расчет количества значений

2. Рассчитать количество проданных продуктов за текущий день
В отличие от предыдущего запроса здесь нет необходимости группировать данные по всем дням. Достаточно выбрать записи, относящиеся к текущему дню, а затем посчитать количество продуктов. Т.е. в строке «Групповая операция» вместо значения *Группировка* выберем *Условие* (обратите внимание, что поле с условием не выводится на экран) (Рис. 3-23). Для задания текущего дня можно использовать функцию Date(), возвращающую каждый день новую дату – текущую. В этом случае запрос станет более универсальным чем, если бы мы задавали конкретную дату.

а)

Продажи	Продукты
*	*
ДатаПродаж	КодПрод
КодПрод	Продукт
Количество	ЕдИзм
ЦенаПродажи	СрокХран(дней)
	УсловияХран

б)

Count-Продукт
6

Поле:	Продукт	ДатаПродажи
Имя таблицы:	Продукты	Продажи
Групповая операция:	Count	Условие
Сортировка:		
Вывод на экран:	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Условие отбора:		Date()
или:		

Рис. 3-23. Расчет количества значений с использованием условия

3. Рассчитать ежедневную сумму продаж продуктов, с учетом того, что ЦенаПродажи – это цена за единицу продукта. В этом запросе необходимо сгруппировать данные по дате продажи, создать вычисляемое поле: СуммаПродажи: Количество*ЦенаПродажи, а затем установить функцию Sum для вычисляемого поля (Рис. 3-24)

а)

Продажи
*
ДатаПродажи
КодПрод
Количество
ЦенаПродажи

б)

ДатаПродажи	СуммаПродажи
25.03.03	1380
02.01.04	9785
18.02.04	2232

Поле:	ДатаПродажи	СуммаПродажи: [Количество]*[ЦенаПродажи]
Имя таблицы:	Продажи	
Групповая операция:	Группировка	Sum
Сортировка:		
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:		
или:		

Рис. 3-24. Расчет суммы

3.2.5. Вложенные запросы

Язык QBE позволяет строить вложенные запросы. То есть можно создать первый запрос, а затем построить второй запрос, вызвав первый запрос в качестве источника данных. В случае использования вложенных запросов, вычисление начинается с самого внутреннего запроса.

1. Получить названия и вес продуктов, проданных 2 января 2004г.

Этот пример мы рассмотрели в п.3.2.2. как условное соединение Рис. 3-16. Рассмотрим теперь этот пример как вложенный запрос. На первом шаге выберем продажи продуктов за 2.01.04, запрос сохраним под именем «ПродажаЗаДень» (Рис. 3-25). На втором шаге соединим запрос «ПродажаЗаДень» с таблицей Продукты для подстановки названий и единиц измерений продуктов (Рис. 3-26).

а)

Microsoft Access - [ПродажаЗаДень : запрос на выборку]

Продажи
*
ДатаПродажи
КодПрод
Количество
ЦенаПродажи

Поле:	ДатаПродажи	КодПрод	Количество
Имя таблицы:	Продажи	Продажи	Продажи
Сортировка:			
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:	#02.01.04#		
или:			

б)

ДатаПродажи	КодПрод	Количество
02.01.04	1	20
02.01.04	4	10
02.01.04	6	40
02.01.04	8	10
02.01.04	12	40
02.01.04	13	30
02.01.04	15	10
02.01.04	16	10
02.01.04	19	5

Рис. 3-25. Первая часть вложенного запроса

а)

ПродажаЗаДень
*
ДатаПродажи
КодПрод
Количество

Продукты
*
КодПрод
Продукт
ЕдИзм
СрокХран(дней)
УсловияХран

Поле:	ДатаПродажи	Продукт	Количество	ЕдИзм
Имя таблицы:	ПродажаЗаДень	Продукты	ПродажаЗаДень	Продукты
Сортировка:				
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:				
или:				

б)

ДатаПродажи	Продукт	Количество	ЕдИзм
02.01.04	Говядина	20	кг
02.01.04	Майонез	10	кг
02.01.04	Сметана	40	кг
02.01.04	Творог	10	кг
02.01.04	Укроп	40	кг
02.01.04	Рис	30	кг
02.01.04	Яблоки	10	кг
02.01.04	Сахар	10	кг
02.01.04	Сок яблочный	5	л

Рис. 3-26. Вторая часть вложенного запроса

2. Какие продукты имеют максимальный срок хранения?

На первом шаге найдем максимальный срок хранения продуктов (Рис. 3-27, а, б).

На втором шаге выберем продукты, соединив поле из запроса, с найденным максимальным сроком хранения, с полем *СрокХранения(дней)* таблицы Продукты (Рис. 3-27, в, г).

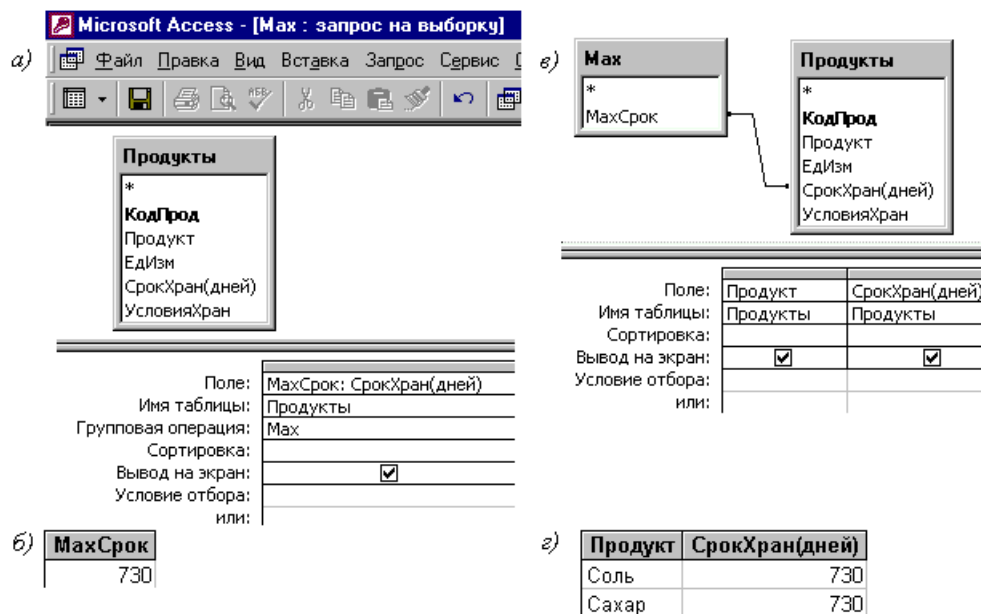


Рис. 3-27. Пример вложенного запроса

3.2.6. Корректирующие запросы

К корректирующим запросам относятся запросы:

- на обновление записей
- на добавление записей
- на удаление записей

При создании таких запросов в конструкторе необходимо изменить вид запроса (по умолчанию установлен запрос на выборку) используя пункт меню *Запрос* или кнопку *Тип запроса* на панели инструментов

1. Запрос на обновление

Изменить написание единицы измерения «л» на «литр» для продуктов, имеющих эту единицу измерения (Рис. 3-28)

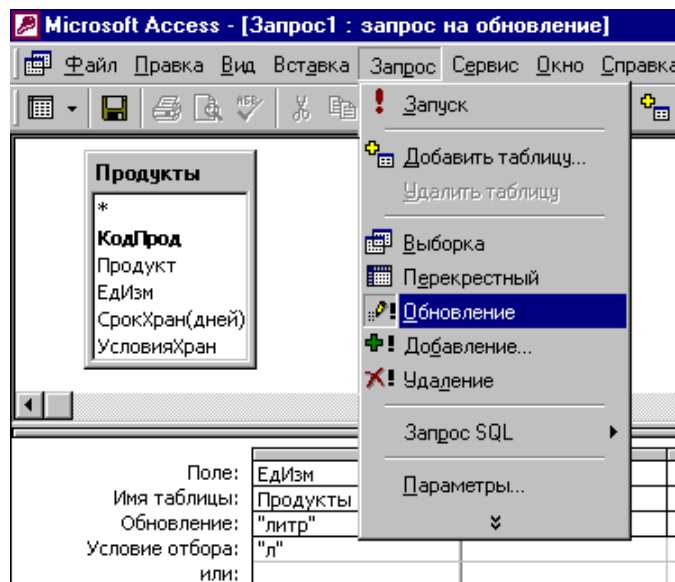


Рис. 3-28. Запрос на обновление

Новое значение, которое должно быть записано в поле ЕдИзм вместо старого, указывается в строке «Обновление».

Строка «Обновление» может содержать выражения, например, увеличить цену продажи на 5%: [ЦенаПродажи]*1,05

2. Запрос на добавление записей в таблицу

Добавить в таблицу Продукты новые продукты из таблицы НовыеПродукты.

При установке вида запроса на добавление необходимо указать (в диалоговом окне «Добавление»), в какую таблицу будут добавляться данные (Рис 3-29)

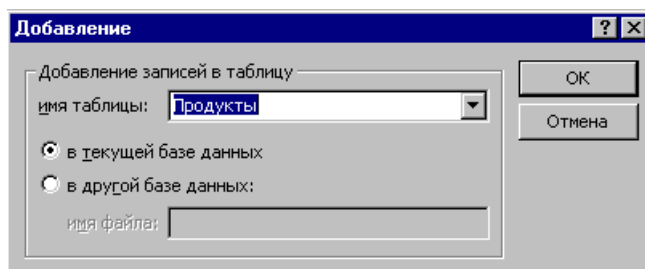


Рис. 3-29. Диалоговое окно «Добавление»

Запрос на добавление продуктов с кодом >21 (т.к. продукты с кодами от 1 до 21 уже есть в таблице Продукты) из таблицы НовыеПродукты в таблицу Продукты приведен на Рис. 3-30. Строка «Добавление» содержит ссылки на имена полей таблицы Продукты.

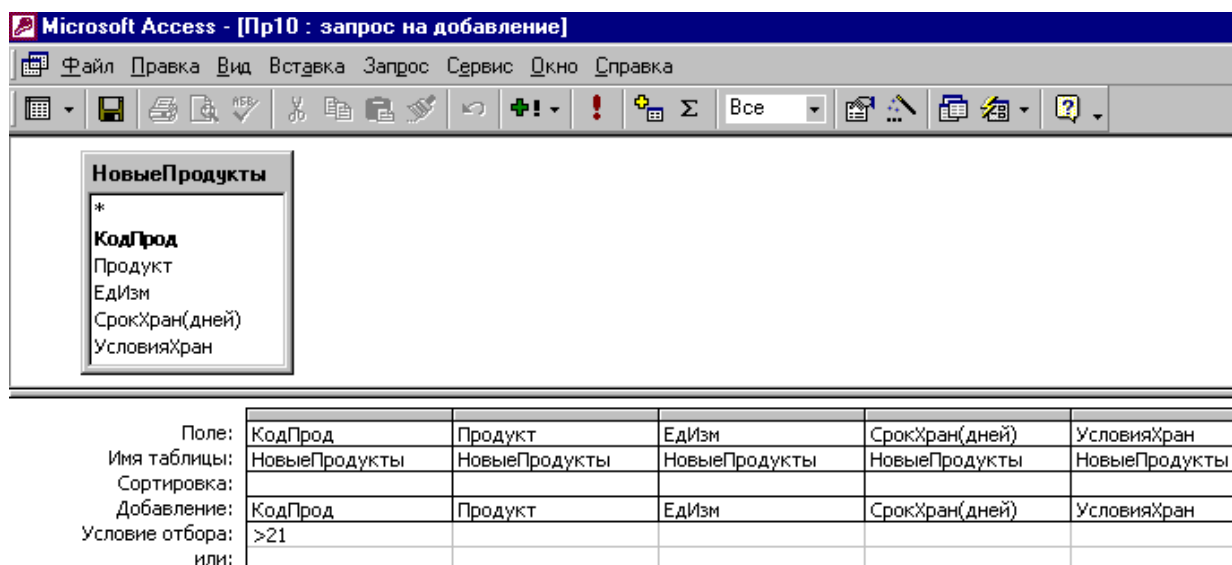


Рис. 3-30. Запрос на добавление записей

3. Запрос на удаление записей

Удалить все записи из таблицы НовыеПродукты (Рис. 3-31)

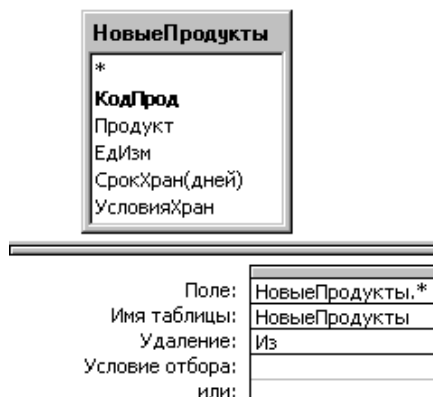


Рис. 3-31. Запрос на удаление всех записей таблицы

Удалить заказы за период с 1 января 2004г. по 31 января 2004г. (Рис. 3-32)

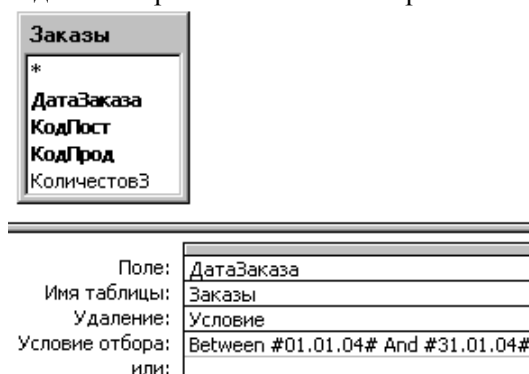


Рис. 3-32. Удаление выборочных записей таблицы

3.2.7. QBE как «построитель» SQL-запросов

В режиме конструктора запросов есть возможность автоматического перевода QBE-запроса в запрос на языке SQL. Используйте для этого пункты меню *Вид, Режим SQL* (Рис. 3-33).

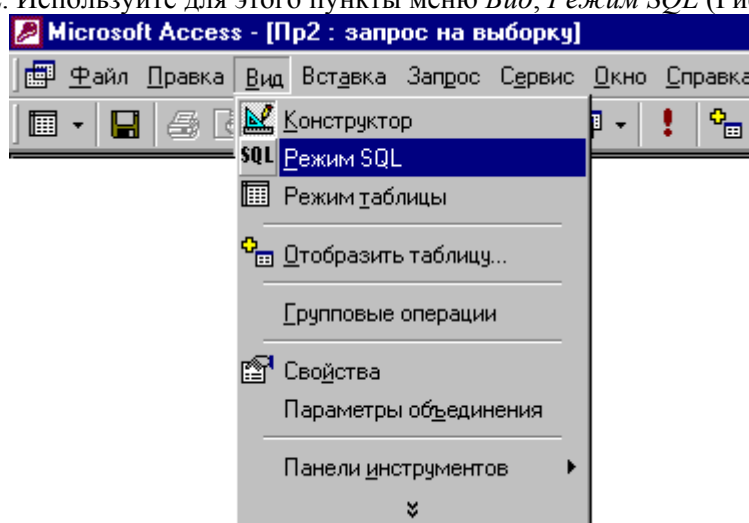


Рис. 3-33. Перевод QBE-запроса в SQL-запрос

Преобразуем запрос, приведенный на Рис. 3-10 в SQL-запрос:

```
SELECT DISTINCT Продажи.КодПрод
FROM Продажи;
```

Модуль 4. Язык SQL

SQL (Structured Query Language) – Структурированный Язык Запросов – стандартный язык запросов по работе с **реляционными БД** [4]. Этот язык был разработан в компании IBM Research в начале 1970-х годов, впоследствии он был реализован в многочисленных коммерческих продуктах как компании IBM, так и других изготовителей.

В 1989 г. был принят первый международный стандарт языка SQL, в 1992 г. – второй стандарт SQL, являющий более полным и точным по сравнению с первым стандартом. В настоящий момент большинство СУБД в той или иной мере соответствуют этому стандарту.

Для иллюстрации работы с языком SQL рассмотрим работу продуктового магазина. Схема данных БД «Магазин» приведена на Рис. 3-8.

Для иллюстрации работы запросов мы будем использовать те же данные, что и для запросов на QBE, приведенные в таблицах (Таблица 3-3 – Таблица 3-9).

Глава 4.1. Оператор выбора SELECT

Все запросы на получение практически любого количества данных из одной или нескольких таблиц выполняются с помощью единственного предложения SELECT. В общем случае результатом реализации предложения SELECT является другая таблица. К этой новой (рабочей) таблице может быть снова применена операция SELECT и т.д., т.е. такие операции могут быть вложены друг в друга. Оператор SELECT заменяет все операторы реляционной алгебры.

4.1.1. Синтаксис оператора SELECT

Предложение SELECT может использоваться как:

- самостоятельная команда на получение и вывод строк таблицы, сформированной из столбцов и строк одной или нескольких таблиц (представлений);
- элемент WHERE- или HAVING-условия (сокращенный вариант предложения, называемый "вложенный запрос");
- фраза выбора в командах CREATE VIEW, DECLARE CURSOR или INSERT;
- средство присвоения глобальным переменным значений из строк сформированной таблицы (INTO-фраза).

Здесь в синтаксических конструкциях используются следующие обозначения:

- звездочка (*) для обозначения "все" - употребляется в обычном для программирования смысле, т.е. "все случаи, удовлетворяющие определению";
- квадратные скобки ([]) – означают, что конструкции, заключенные в эти скобки, являются необязательными (т.е. могут быть опущены);
- фигурные скобки ({}) – означают, что конструкции, заключенные в эти скобки, должны рассматриваться как целые синтаксические единицы, т.е. они позволяют уточнить порядок разбора синтаксических конструкций, заменяя обычные скобки, используемые в синтаксисе SQL;
- многоточие (...) – указывает на то, что непосредственно предшествующая ему синтаксическая единица факультативно может повторяться один или более раз;
- прямая черта (|) – означает наличие выбора из двух или более возможностей. Например обозначение ASC|DESC указывает, можно выбрать один из терминов ASC или DESC; когда же один из элементов выбора заключен в квадратные скобки, то это означает, что он выбирается по умолчанию (так, [ASC]|DESC означает, что отсутствие всей этой конструкции будет восприниматься как выбор ASC);
- точка с запятой (;) – завершающий элемент предложений SQL;
- запятая (,) – используется для разделения элементов списков;
- пробелы () – могут вводиться для повышения наглядности между любыми синтаксическими конструкциями предложений SQL;
- прописные жирные латинские буквы и символы – используются для написания конструкций языка SQL и должны (если это специально не оговорено) записываться в точности так, как показано;
- строчные буквы – используются для написания конструкций, которые должны заменяться конкретными значениями, выбранными пользователем, причем для определенности отдельные слова этих конструкций связываются между собой символом подчеркивания (_);
- термины таблица, столбец, ... – заменяют (с целью сокращения текста синтаксических конструкций) термины имя_таблицы, имя_столбца, ..., соответственно;
- термин таблица – используется для обобщения таких видов таблиц, как базовая_таблица, представление или псевдоним; здесь псевдоним служит для временного (на момент выполнения запроса) переименования и (или) создания рабочей копии базовой_таблицы (представления).

Предложение SELECT (выбрать) имеет следующий формат:

```
подзапрос [UNION [ALL] подзапрос] ...  
[ORDER BY {[таблица.]столбец | номер_элемента_SELECT} [[ASC] |  
DESC]  
[, {[таблица.]столбец | номер_элемента_SELECT} [[ASC] | DESC]] ...;
```

и позволяет объединить (UNION) а затем упорядочить (ORDER BY) результаты выбора данных, полученных с помощью нескольких "подзапросов". При этом упорядочение можно производить в порядке возрастания - ASC (ASCending) или убывания DESC (DESCending), а по умолчанию принимается ASC.

В этом предложении подзапрос позволяет указать условия для выбора нужных данных и (если требуется) их обработки

SELECT

(выбрать) данные из указанных столбцов и (если необходимо) выполнить перед выводом их преобразование в соответствии с указанными выражениями и (или) функциями

FROM

(из) перечисленных таблиц, в которых расположены эти столбцы

WHERE

(где) строки из указанных таблиц должны удовлетворять указанному перечню условий отбора строк

GROUP BY

(группируя по) указанному перечню столбцов с тем, чтобы получить для каждой группы единственное агрегированное значение, используя во фразе SELECT SQL-функции SUM (сумма), COUNT (количество), MIN (минимальное значение), MAX (максимальное значение) или AVG (среднее значение)

HAVING

(имея) в результате лишь те группы, которые удовлетворяют указанному перечню условий отбора групп

и имеет формат

```
SELECT      [[ALL] | DISTINCT]{ * | элемент_SELECT [,элемент_SELECT]
...}
FROM        {базовая_таблица | представление} [псевдоним]
            [, {базовая_таблица | представление} [псевдоним]] ...
[WHERE      фраза]
[GROUP BY  фраза [HAVING фраза]];
```

Элемент_SELECT - это одна из следующих конструкций:

[таблица.]* | значение | SQL_функция | системная_переменная

где значение – это:

[таблица.]столбец | (выражение) | константа | переменная

Синтаксис выражений имеет вид

({ [[+] | -] {значение | функция_СУБД} [+ | - | * | **] } ...)

а синтаксис SQL_функций – одна из следующих конструкций:

{SUM|AVG|MIN|MAX|COUNT} ([[ALL]|DISTINCT] [таблица.]столбец)

{SUM|AVG|MIN|MAX|COUNT} ([ALL] выражение)

COUNT (*)

Фраза WHERE включает набор условий для отбора строк:

WHERE [NOT] WHERE_условие [[AND|OR][NOT] WHERE_условие]...

где WHERE_условие – одна из следующих конструкций:

значение { = | <> | < | <= | > | >= } { значение | (подзапрос) }

значение_1 [NOT] BETWEEN значение_2 AND значение_3

значение [NOT] IN { (константа [,константа]...) | (подзапрос) }

значение IS [NOT] NULL

[таблица.]столбец [NOT] LIKE 'строка_символов' [ESCAPE 'символ']

EXISTS (подзапрос)

Кроме традиционных операторов сравнения (= | <> | < | <= | > | >=) в WHERE фразе используются условия BETWEEN (между), LIKE (похоже на), IN (принадлежит), IS NULL (не определено) и EXISTS (существует), которые могут предваряться оператором NOT (не). Критерий

отбора строк формируется из одного или нескольких условий, соединенных логическими операторами:

AND

- когда должны удовлетворяться оба разделяемых с помощью AND условия;

OR

- когда должно удовлетворяться одно из разделяемых с помощью OR условий;

AND NOT

- когда должно удовлетворяться первое условие и не должно второе;

OR NOT

- когда или должно удовлетворяться первое условие или не должно удовлетворяться второе, причем существует приоритет AND над OR (сначала выполняются все операции AND и только после этого операции OR). Для получения желаемого результата WHERE условия должны быть введены в правильном порядке, который можно организовать введением скобок.

При обработке условия числа сравниваются алгебраически - отрицательные числа считаются меньшими, чем положительные, независимо от их абсолютной величины. Строки символов сравниваются в соответствии с их представлением в коде, используемом в конкретной СУБД, например, в коде ASCII. Если сравниваются две строки символов, имеющих разные длины, более короткая строка дополняется справа пробелами для того, чтобы они имели одинаковую длину перед осуществлением сравнения.

Наконец, синтаксис фразы GROUP BY имеет вид

GROUP BY [таблица.]столбец [, [таблица.]столбец] ... [HAVING фраза]

GROUP BY инициирует перекомпоновку формируемой таблицы по группам, каждая из которых имеет одинаковое значение в столбцах, включенных в перечень GROUP BY. Далее к этим группам применяются агрегирующие функции, указанные во фразе SELECT, что приводит к замене всех значений группы на единственное значение (сумма, количество и т.п.).

С помощью фразы HAVING (синтаксис которой почти не отличается от синтаксиса фразы WHERE)

HAVING [NOT] HAVING_условие [[AND|OR][NOT] HAVING_условие]...

можно исключить из результата группы, не удовлетворяющие заданным условиям:

значение { = | < > | <= | >= } { значение | (подзапрос)

| SQL_функция }

{значение_1 | SQL_функция_1} [NOT] BETWEEN

{значение_2 | SQL_функция_2} AND {значение_3 | SQL_функция_3}

{значение | SQL_функция} [NOT] IN { (константа [,константа]...)

| (подзапрос) }

{значение | SQL_функция} IS [NOT] NULL

[таблица.]столбец [NOT] LIKE 'строка_символов' [ESCAPE 'символ']

EXISTS (подзапрос)

4.1.2. Запросы с использованием одной таблицы

1. *Простая выборка* (выбрать полную информацию о продуктах с сортировкой по алфавиту)

SELECT *

FROM Продукты

ORDER BY Продукт;

Здесь "звездочка" (*) служит кратким обозначением всех имен полей в таблице, указанной во фразе FROM. При этом порядок вывода полей соответствует порядку, в котором эти поля определялись при создании таблицы.

Запрос выдает результат, указанный на Рис. 3-9, б.

2. *Исключение дубликатов* (выдать перечень проданных продуктов без повторов)

Этот запрос является аналогом операции *проекции* реляционной алгебры.

Для исключения дубликатов и одновременного упорядочения перечня необходимо дополнить запрос ключевым словом DISTINCT (различный, различные), как показано в следующем примере:

SELECT DISTINCT КодПрод

FROM Продажи;

Результат запроса приведен на Рис. 3-10, б.

3. *Задание условия отбора* (выбрать поставщиков, в названии которых есть сокращение ООО, ОАО или ЗАО).

Для задания условия отбора используется фраза WHERE

В запросе используется условие Like для отбора части значения поля, символ * в данном контексте соответствует любому количеству цифр или символов

```
SELECT Поставщик
FROM Поставщики
WHERE Поставщик Like '*ООО*' Or Поставщик Like '*ОАО*' Or Поставщик
Like '*ЗАО*';
```

Результат запроса приведен на Рис. 3-11, в.

4. *Задание диапазонов в запросах* (выбрать заказы за период с 1.02.04 по 31.03.04).

Диапазон можно задать, используя конструкцию Between ... And ... (находится в интервале от ... до ...)

```
SELECT ДатаЗаказа, КодПост, КодПрод, КоличествЗ
FROM Заказы
WHERE ДатаЗаказа Between #2-1-2004# And #3-31-2004#;
```

Результат запроса приведен на Рис. 4-1.

ДатаЗаказа	КодПост	КодПрод	КоличествЗ
02.02.04	3	9	75
05.02.04	2	14	70
09.02.04	7	8	150
12.02.04	1	11	50
12.02.04	1	12	10
14.02.04	4	7	200
16.02.04	6	10	90
16.02.04	8	11	100

Рис. 4-1. Результат запроса на выбор заказов за период

4.1.3. *Возможности совместной обработки нескольких таблиц*

1. *Декартово произведение*

Декартово произведение может потребоваться для получения всех сочетаний значений таблиц. Получим все возможные сочетания поставщиков и продуктов, т.е. ВСЕ поставщики поставляют ВСЕ продукты.

```
SELECT *
FROM Поставщики, Продукты;
Или
SELECT Поставщики.*, Продукты.*
FROM Поставщики, Продукты;
```

Результат запроса приведен на Рис. 3-14, б.

2. *Естественное соединение*

Получить список продаж с характеристиками продуктов. Поскольку продажи продуктов хранятся в таблице Продажи, а названия – в таблице Продукты, то для получения необходимого результата в запросе нужно использовать обе таблицы. В результат должны быть включены те записи, для которых код продукта из таблицы Продукты совпадает с кодом продукта из таблицы Продажи.

```
SELECT Продажи.*, Продукт, ЕдИзм, [СрокХран(дней)], УсловияХран
FROM Продукты, Продажи
WHERE Продукты.КодПрод = Продажи.КодПрод;
```

Результат запроса приведен на Рис. 3-15, б.

3. *Условное соединение*

Получить названия и вес продуктов, проданных 2 января 2004г. В отличие от предыдущего запроса здесь к условию связи таблиц по поля код продукта добавляется условие отбора даты продажи

```

SELECT ДатаПродажи, Продукт, Количество, ЕдИзм
FROM Продукты, Продажи
WHERE ДатаПродажи = #1-2-2004# AND Продукты.КодПрод =
Продажи.КодПрод;

```

Результат запроса приведен на Рис. 3-16, б.

4. *Объединение* двух таблиц содержит те записи, которые есть либо в первой, либо во второй, либо в обеих таблицах. Объединить записи таблиц **Продукты** и **НовыеПродукты**. Поскольку таблицы имеют эквивалентные схемы, то в запрос можно включить все поля:

```

SELECT Продукты.*
FROM Продукты
UNION SELECT НовыеПродукты.*
FROM НовыеПродукты;

```

4.1.4. Вычисляемые поля

1. Переименование полей

Иногда для удобства работы требуется переименовать некоторые поля в запросе (например, при наличии одноименных полей в разных таблицах).

Выдать список всех поставленных продуктов (кодов продуктов и названий продуктов) без повторений.

```

SELECT DISTINCT Поставки.КодПрод AS КодПоставленногоПродукта,
Продукт

```

```

FROM Поставки, Продукты

```

```

WHERE Поставки.КодПрод = Продукты.КодПрод;

```

Результат запроса приведен на Рис. 4-2.

КодПоставленногоПродукта	Продукт
1	Говядина
2	Судак
3	Масло
4	Майонез
5	Яйца
6	Сметана
7	Молоко
8	Творог
9	Морковь
10	Лук
11	Помидоры
12	Укроп
13	Рис
14	Мука
15	Яблоки
16	Сахар
17	Кофе

Рис. 4-2. Запрос на переименование полей с исключением дубликатов

2. Выборка вычисляемых значений

Выдать список поставленных продуктов (дата поставки, продукт). Вычислить дату окончания срока хранения продуктов. Отсортировать полученный результат по дате поставки.

В БД «Магазин» хранятся: дата изготовления продукта (поле *ДатаИзгот* таблицы **Поставки**) и срок хранения продукта (в днях) (поле *СрокХран(дней)* таблицы **Продукты**). Если к дате прибавить количество дней, то получится новая дата.

```

SELECT ДатаПоставки, Продукт, [ДатаИзгот] + [СрокХран(дней)] AS
ДатаОкончХран

```

```

FROM Продукты, Поставки

```

```

WHERE Продукты.КодПрод = Поставки.КодПрод

```

```

ORDER BY ДатаПоставки;

```

Глава 4.2. Применение агрегатных функций и вложенных запросов в операторе выбора

4.2.1. SQL-функции

В SQL существует ряд специальных стандартных функций (SQL-функций). Кроме специального случая COUNT(*) каждая из этих функций оперирует совокупностью значений поля некоторой таблицы и создает единственное значение, определяемое так:

- COUNT – подсчет количества записей, содержащихся в заданном поле запроса
- SUM – вычисление суммы набора значений, содержащихся в заданном поле запроса
- AVG – вычисление арифметического среднего набора чисел, содержащихся в указанном поле запроса
- MAX – вычисление максимального значения из набора значений, содержащихся в указанном поле запроса
- MIN – вычисление минимального значения из набора значений, содержащихся в указанном поле запроса

Для функций SUM и AVG рассматриваемый столбец должен содержать числовые значения.

Следует отметить, что здесь поле – это поле виртуальной таблицы, в которой могут содержаться данные не только из поля базовой таблицы, но и данные, полученные путем функционального преобразования и (или) связывания символами арифметических операций значений из одного или нескольких полей. При этом выражение, определяющее поле такой таблицы, может быть сколь угодно сложным, но не должно содержать SQL-функций (вложенность SQL-функций не допускается). Однако из SQL-функций можно составлять любые выражения.

Аргументу всех функций, кроме COUNT(*), может предшествовать ключевое слово DISTINCT (различный), указывающее, что избыточные дублирующие значения должны быть исключены перед тем, как будет применяться функция.

Специальная функция COUNT(*) служит для подсчета всех без исключения записей в таблице (включая дубликаты).

1. Посчитать количество поставщиков

```
SELECT Count(*) AS Количество  
FROM Поставщики;
```

Количество
9

Результат запроса

Если не используется фраза GROUP BY, то в перечень элементов SELECT можно включать лишь SQL-функции или выражения, содержащие такие функции. Другими словами, нельзя иметь в списке столбцы, не являющихся аргументами SQL-функций.

2. Выдать данные о массе творога (КодПрод=8), поставленного поставщиками, и указать количество этих поставок

```
SELECT Sum(КоличествоП) AS Вес, Count(КоличествоП) AS Количество  
FROM Поставки  
WHERE КодПрод = 8;
```

Вес	Количество
170	2

Результат запроса

3. Рассчитать ежедневную сумму продаж продуктов и вес проданных продуктов

```
SELECT ДатаПродажи, Sum([Количество]*[ЦенаПродажи]) AS СуммаПродажи  
FROM Продажи  
GROUP BY ДатаПродажи;
```

ДатаПродажи	СуммаПродажи
25.03.03	1380
02.01.04	9785
18.02.04	2232

Результат запроса

Фраза GROUP BY (группировать по) инициирует перекомпоновку указанной во FROM таблицы по группам, каждая из которых имеет одинаковые значения в поле, указанном в GROUP BY. В рассматриваемом примере записи таблицы Продажи группируются так, что в одной группе содержатся все записи с датой продажи ДатаПродажи = 25.03.03, в другой с датой продажи ДатаПродажи = 02.01.04 и т.д. (см. Таблицу 3-6). Далее к каждой группе применяется фраза SELECT. Каждое выражение в этой фразе должно принимать единственное значение для группы, т.е. оно может быть либо значением поля, указанного в GROUP BY, либо арифметическим выражением, включающим это значение, либо константой, либо одной из SQL-функций, которая оперирует всеми значениями поля в группе и сводит эти значения к единственному значению (например, к сумме).

4.2.2. Вложенные подзапросы

Виды вложенных подзапросов

Вложенный подзапрос - это подзапрос, заключенный в круглые скобки и вложенный в WHERE (HAVING) фразу предложения SELECT или других предложений, использующих WHERE фразу. Вложенный подзапрос может содержать в своей WHERE (HAVING) фразе другой вложенный подзапрос и т.д. Вложенный подзапрос создан для того, чтобы при отборе записей таблицы, сформированной основным запросом, можно было использовать данные из других таблиц.

Существуют простые и коррелированные вложенные подзапросы. Они включаются в WHERE (HAVING) фразу с помощью условий IN, EXISTS или одного из условий сравнения (= | < > | <= | >=). Простые вложенные подзапросы обрабатываются системой "снизу вверх". Первым обрабатывается вложенный подзапрос самого нижнего уровня. Множество значений, полученное в результате его выполнения, используется при реализации подзапроса более высокого уровня и т.д.

Запросы с коррелированными вложенными подзапросами обрабатываются системой в обратном порядке. Сначала выбирается первая строка рабочей таблицы, сформированной основным запросом, и из нее выбираются значения тех столбцов, которые используются во вложенном подзапросе (вложенных подзапросах). Если эти значения удовлетворяют условиям вложенного подзапроса, то выбранная строка включается в результат. Затем выбирается вторая строка и т.д., пока в результат не будут включены все строки, удовлетворяющие вложенному подзапросу (последовательности вложенных подзапросов).

Вложенные подзапросы с предикатом IN

Простые вложенные подзапросы используются для представления множества значений, исследование которых должно осуществляться в каком-либо предикате IN

1. Подзапрос с одним уровнем вложенности

Выдать название и телефон поставщиков продукта с кодом 9, т.е. моркови.

```
SELECT Поставщик, Телефон
FROM Поставщики
WHERE КодПост IN
(SELECT КодПост
FROM Поставки
WHERE КодПрод=9);
```

Как уже отмечалось, при обработке полного запроса система выполняет прежде всего вложенный подзапрос. Этот подзапрос выдает множество номеров поставщиков, которые поставляют продукт с кодом КодПрод = 9, а именно множество (1, 2, 3). Поэтому первоначальный запрос эквивалентен такому простому запросу:

```
SELECT Поставщик, Телефон
FROM Поставщики
WHERE КодПост IN (1,2,3);
```

Результат запроса	Поставщик		Телефон
	ОАО "Приморье"		223344
	ПБОЮЛ Свиридова А.Н.		265493
	Овощебаза №8		94238

2. Подзапрос с несколькими уровнями вложенности

Пусть требуется узнать не поставщиков продукта 9, как это делалось в предыдущем запросе, а поставщиков моркови

```
SELECT Поставщик, Телефон
FROM Поставщики
WHERE КодПост IN
  (SELECT КодПост
   FROM Поставки
   WHERE КодПрод IN
    (SELECT КодПрод
     FROM Продукты
     WHERE Продукт = 'Морковь' ) ) ;
```

В данном случае результатом самого внутреннего подзапроса является только одно значение (9). Как уже было показано выше, подзапрос следующего уровня в свою очередь дает в результате множество (1, 2, 3). Последний, самый внешний SELECT, вычисляет приведенный выше окончательный результат. Вообще допускается любая глубина вложенности подзапросов.

SQL позволяет одни и те же запросы формулировать несколькими способами.

Тот же результат можно получить с помощью следующего запроса:

```
SELECT Поставщик, Телефон
FROM Поставщики, Поставки, Продукты
WHERE Поставщики.КодПост = Поставки.КодПост
      AND Поставки.КодПрод = Продукты.КодПрод
      AND Продукт = 'Морковь' ;
```

При выполнении этого компактного запроса система должна одновременно обрабатывать данные из трех таблиц, тогда как в предыдущем примере эти таблицы обрабатываются поочередно. Естественно, что для их реализации требуются различные ресурсы памяти и времени, однако этого невозможно ощутить при работе с ограниченным объемом данных в иллюстративной БД «Магазин».

3. Пересечение двух таблиц

Найти продукты, которые есть и в таблице Продукты, и в таблице НовыеПродукты

```
SELECT Продукт
FROM Продукты
WHERE Продукт IN
  (SELECT Продукт
   FROM НовыеПродукты);
Или
SELECT Продукты.Продукт
FROM Продукты, НовыеПродукты
WHERE Продукты.Продукт = НовыеПродукты.Продукт;
```

4. Разность двух таблиц

Найти продукты, которые есть в таблице Продукты, но отсутствуют в таблице НовыеПродукты

```
SELECT Продукт
FROM Продукты
WHERE Продукт NOT IN
  (SELECT Продукт
   FROM НовыеПродукты) ;
```

Вложенный подзапрос с оператором сравнения, отличным от IN

Выдать продукты, имеющие ту же единицу измерения, что и молоко.

```
SELECT Продукт
FROM Продукты
WHERE ЕдИзм =
  (SELECT ЕдИзм
   FROM Продукты
   WHERE Продукт = 'Молоко' ) ;
```

Продукт
Масло
Молоко
Сливки
Сок яблочный

Результат запроса

В подобных запросах можно использовать и другие операторы сравнения (<>, <=, <, >= или >), однако, если вложенный подзапрос возвращает более одного значения и не используется оператор IN, будет возникать ошибка.

Глава 4.3. Операторы манипулирования данными

Модификация данных может выполняться с помощью предложений UPDATE (обновить), INSERT (вставить) и DELETE (удалить).

Запросы на обновление

Предложение UPDATE имеет формат:

```
UPDATE {базовая таблица | представление}
SET столбец = значение [, столбец = значение] ...
[WHERE фраза]
```

где значение – это: столбец | выражение | константа | переменная

и может включать столбцы лишь из обновляемой таблицы, т.е. значение одного из столбцов модифицируемой таблицы может заменяться на значение ее другого столбца или выражения, содержащего значения нескольких ее столбцов, включая изменяемый.

При отсутствии WHERE фразы обновляются значения указанных столбцов во всех строках модифицируемой таблицы. WHERE фраза позволяет сократить число обновляемых строк, указывая условия их отбора.

Пример: Изменить написание единицы измерения «л» на «литр» для продуктов, имеющих эту единицу измерения

```
UPDATE Продукты SET ЕдИзм = 'литр'
WHERE ЕдИзм= 'л';
```

Запросы на добавление записей

Предложение INSERT имеет следующий формат:

```
INSERT
INTO {базовая таблица | представление} [(столбец [, столбец] ...)]
подзапрос;
```

Сначала выполняется подзапрос, т.е. по предложению SELECT в памяти формируется рабочая таблица, а потом строки рабочей таблицы загружаются в модифицируемую таблицу. При этом i-й столбец рабочей таблицы (i-й элемент списка SELECT) соответствует i-му столбцу в списке столбцов модифицируемой таблицы.

Пример: Добавить в таблицу Продукты новые продукты из таблицы НовыеПродукты с кодами продукта > 21

```
INSERT INTO Продукты (КодПрод, Продукт, ЕдИзм, [СрокХран (дней)],
УсловияХран)
SELECT КодПрод, Продукт, ЕдИзм, [СрокХран (дней)], УсловияХран
FROM НовыеПродукты
WHERE НовыеПродукты.КодПрод > 21;
```

Запросы на удаление

Предложение DELETE имеет формат

```
DELETE
FROM базовая таблица | представление
[WHERE фраза];
```

и позволяет удалить содержимое всех строк указанной таблицы (при отсутствии WHERE фразы) или тех ее строк, которые выделяются WHERE фразой.

Пример 1: Удаление всех записей таблицы
Удалить все записи из таблицы НовыеПродукты
DELETE НовыеПродукты.*
FROM НовыеПродукты;

Пример 2: *Удаление выборочных записей*
Удалить заказы за период с 1 января 2004г. по 31 января 2004г.
DELETE ДатаЗаказа
FROM Заказы
WHERE ДатаЗаказа Between #1-1-2004# And #1-31-2004#;

Заключение. Направления развития баз данных

Несмотря на всю их привлекательность, классические реляционные системы управления базами данных являются ограниченными. Они идеально подходят для таких традиционных приложений, как системы резервирования билетов или мест в гостиницах, а также банковских систем, но их применение в системах автоматизации проектирования, интеллектуальных системах обучения и других системах, основанных на знаниях, часто является затруднительным. Это, прежде всего, связано с примитивностью структур данных, лежащих в основе реляционной модели данных. Плоские нормализованные отношения универсальны и теоретически достаточны для представления данных любой предметной области. Однако в нетрадиционных приложениях в базе данных появляются сотни, если не тысячи таблиц, над которыми постоянно выполняются дорогостоящие операции соединения, необходимые для воссоздания сложных структур данных, присущих предметной области.

Другим серьезным ограничением реляционных систем являются их относительно слабые возможности по части представления семантики приложения. Самое большее, что обеспечивают реляционные СУБД, - это возможность формулирования и поддержки ограничений целостности данных.

Осознавая эти ограничения и недостатки реляционных систем, исследователи в области баз данных выполняют многочисленные проекты, основанные на идеях, выходящих за пределы реляционной модели данных. По всей видимости, какая-либо из этих работ станет основой систем баз данных будущего.

Объектно-ориентированные БД

Направление объектно-ориентированных баз данных (ООБД) возникло сравнительно давно. Публикации появлялись уже в середине 1980-х. Однако наиболее активно это направление развивается в последние годы. С каждым годом увеличивается число публикаций и реализованных коммерческих и экспериментальных систем.

Возникновение направления ООБД определяется, прежде всего, потребностями практики: необходимостью разработки сложных информационных прикладных систем, для которых технология предшествующих систем БД не была вполне удовлетворительной.

В объектно-ориентированной парадигме предметная область моделируется как множество классов взаимодействующих объектов. Каждый объект характеризуется набором свойств, которые являются как бы его пассивными характеристиками и набором методов работы с этим объектом. Работать с объектом можно только с использованием его методов. Атрибуты объекта могут принимать определенное множество допустимых значений, набор конкретных значений атрибутов объекта определяет его состояние. Множество объектов с одним и тем же набором атрибутов и методов образует класс объектов.

Одной из наиболее перспективных черт объектно-ориентированной парадигмы является принцип наследования. Допускается порождение нового класса на основе уже существующего класса, и этот процесс называется наследованием. В этом случае новый класс, называемый подклассом существующего класса (суперкласса), наследует все атрибуты и методы суперкласса. В подклассе, кроме того, могут быть определены дополнительные атрибуты и методы. Различаются случаи простого и множественного наследования. В первом случае подкласс может определяться только на основе одного суперкласса, во втором случае суперклассов может быть несколько.

Можно считать, что наиболее важным качеством ООБД, которое позволяет реализовать объектно-ориентированный подход, является учет поведенческого аспекта объектов.

В прикладных информационных системах, основанных на реляционных БД, существовал принципиальный разрыв между структурной и поведенческой частями. Структурная часть системы поддерживалась всем аппаратом БД, ее можно было моделировать, верифицировать и т.д., а поведенческая часть создавалась изолированно. В частности, отсутствовали формальный аппарат и системная поддержка совместного моделирования и гарантий согласованности структурной (статической) и поведенческой (динамической) частей. В среде ООБД проектирование, разработка и сопровождение прикладной системы становятся процессом, в котором интегрируются структурный и поведенческий аспекты. Конечно, для этого нужны специальные языки, позволяющие определять объекты и создавать на их основе прикладную систему.

Распределенные БД

Основная задача систем управления распределенными базами данных состоит в обеспечении средства интеграции локальных баз данных, располагающихся в некоторых узлах вычислительной сети, с тем, чтобы пользователь, работающий в любом узле сети, имел доступ ко всем этим базам данных как к единой базе данных.

При этом должны обеспечиваться:

- простота использования системы;
- возможности автономного функционирования при нарушениях связности сети или при административных потребностях;
- высокая степень эффективности.

Возможны однородные и неоднородные распределенные базы данных. В однородном случае каждая локальная база данных управляется одной и той же СУБД. В неоднородной системе локальные базы данных могут относиться даже к разным моделям данных. Сетевая интеграция неоднородных баз данных - это актуальная, но очень сложная проблема. Многие решения известны на теоретическом уровне, но пока не удается справиться с главной проблемой - недостаточной эффективностью интегрированных систем.

Темпоральные БД

Обычные БД хранят мгновенный снимок модели предметной области. Любое изменение в момент времени t некоторого объекта приводит к недоступности состояния этого объекта в предыдущий момент времени. Самое интересное, что на самом деле в большинстве развитых СУБД предыдущее состояние объекта сохраняется в журнале изменений, но возможности доступа со стороны пользователя нет.

Конечно, можно явно ввести в хранимые отношения явный временной атрибут и поддерживать его значения на уровне приложений. Более того, в большинстве случаев так и поступают. Недаром в стандарте SQL появились специальные типы данных `date` и `time`. Но в таком подходе имеются несколько недостатков: СУБД не знает семантики временного поля отношения и не может контролировать корректность его значений; появляется дополнительная избыточность хранения (предыдущее состояние объекта данных хранится и в основной БД, и в журнале изменений); языки запросов реляционных СУБД не приспособлены для работы со временем.

Существует отдельное направление исследований и разработок в области темпоральных БД. В этой области исследуются вопросы моделирования данных, языки запросов, организация данных во внешней памяти и т.д. Основной тезис темпоральных систем состоит в том, что для любого объекта данных, созданного в момент времени t_1 и уничтоженного в момент времени t_2 , в БД сохраняются (и доступны пользователям) все его состояния во временном интервале $[t_1, t_2]$.

Исследования и построения прототипов темпоральных СУБД обычно выполняются на основе некоторой реляционной СУБД. Темпоральная СУБД - это надстройка над реляционной системой. Конечно, это не лучший способ реализации с точки зрения эффективности, но он прост и позволяет производить достаточно глубокие исследования.

Глоссарий

База данных (БД)	именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области
Банк данных (БнД)	это система специальным образом организованных данных – баз данных, программных, технических, языковых, организационно-методических средств, предназначенных для обеспечения централизованного накопления и коллективного многоцелевого использования данных
Система управления базами данных (СУБД)	совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями
Жизненный цикл БД	этапы развития БД, начиная от анализа предметной области, и заканчивая эксплуатацией БД
Данные	это набор конкретных значений, параметров, характеризующих объект, условие, ситуацию или любые другие факторы
Модель данных	это некоторая абстракция, которая, будучи приложима к конкретным данным, позволяет пользователям и разработчикам трактовать их уже как информацию, то есть сведения, содержащие не только данные, но и взаимосвязь между ними
Модель «сущность-связь»	представление предметной области как множество сущностей, обладающих некоторыми свойствами, между которыми существует некоторое множество связей
Сущность	это реальный или представляемый объект, информация о котором должна сохраняться в проектируемой системе
Домен	множество допустимых значений (область определения) атрибута
Атрибут	именованная характеристика, определяющая свойства данной сущности (объекта)
Ключ	минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности
Связь	ассоциация, устанавливаемая между несколькими сущностями, и показывающая как взаимодействуют сущности между собой
Системный анализ предметной области	подробное словесное описание объектов предметной области и реальных связей между описываемыми объектами
Инфологическое (семантическое) моделирование	представление семантики предметной области в концептуальной модели БД, т.е. моделирование структур данных, опираясь на смысл этих данных.
Семантическое моделирование	См. Инфологическое моделирование
Концептуальная модель	обобщенная модель предметной области, для которой создается БД, не зависящая от конкретной СУБД
Фактографическая модель	соответствует представлению информации в виде определенных структур данных (дерево, сеть, таблица и т.п.). К фактографическим моделям относятся: иерархические, сетевые, реляционные, объектно-ориентированные модели.
Документальная модель	соответствуют представлению о слабоструктурированной информации, ориентированной в на свободные форматы документов, текстов на естественном языке
Реляционная БД	БД, воспринимаемая пользователем как набор нормализованных отношений
Целостность данных	правильность данных в любой момент времени при манипулировании данными
Структурная целостность	допустимыми являются только данные, представленные в виде отношений реляционной модели
Языковая целостность	поддержка языков манипулирования данными высокого уровня

Ссылочная целостность

поддержка непротиворечивого состояния БД в процессе
модификации данных

отсутствие несогласованных значений внешних ключей, т.е. для
каждого значения внешнего ключа появляющегося в подчиненном
отношении, в основном отношении должен существовать кортеж с
таким же значением первичного ключа

**Семантическая
целостность**

ограничения, связанные с содержанием БД

**Неопределенное значение
(Null-значение)**

значение, неизвестное на данный момент времени

Схема БД

совокупность схем отношений, адекватно моделирующих
абстрактные объекты предметной области и семантические связи
между этими объектами

SQL

структурированный язык запросов – стандартный язык запросов по
работе с реляционными БД

Список литературы

Основная

1. Дейт К.Дж. Введение в системы баз данных / К.Дж. Дейт. – К.;М.;СПб: Вильямс, 2001. – 1096 с.
2. Карпова Т.С. Базы данных: модели, разработка, реализация / Т.С. Карпова. – СПб.: Питер, 2001. – 304 с.
3. Михеева В.Д. Microsoft Access 2002 / В.Д. Михеева, И.А. Харитоновна – СПб.: БХВ, 2002.

Дополнительная

1. Диго С.М. Проектирование и использование баз данных / С.М. Диго – М.: Финансы и статистика, 1995.
2. Бойко В.В. Проектирование баз данных информационных систем / В.В. Бойко, В.М. Савинков – М., 1989.
3. Джексон Г. Проектирование реляционных баз данных для использования с микроЭВМ / Г. Джексон – М.: Мир, 1991.
4. Кириллов В.В. Структуризованный язык запросов (SQL) / В.В. Кириллов, Г.Ю. Громов – СПб.: ИТМО, 1994 (<http://iclub.nsu.ru/~abstract/docs/SQL/index.shtml>)
5. Мартин Дж. Планирование развития автоматизированных систем / Дж. Мартин – М.: Финансы и статистика, 1984.
6. Мейер М. Теория реляционных баз данных / М. Мейер – М.: Мир, 1987.
7. Тиори Т. Проектирование структур баз данных. В 2 кн. / Т. Тиори, Дж. Фрай – М.: Мир, 1985.
8. Цикритизис Д. Модели данных / Д. Цикритизис, Ф. Лоховски – М.: Финансы и статистика, 1985.

Задания для самостоятельной работы

Модель «сущность-связь»

1. Привести примеры связей между сущностями:
 - 1:1
 - 1:N
 - M:NУказать и пояснить классы принадлежности приведенных сущностей.
2. Привести пример зависимой сущности.
3. Как изменится модель "сущность-связь", рассмотренная в параграфе 2.1.2, если необходимо хранить сведения о покупателях и учитывать покупателей при продаже продуктов?
4. Построить модель "сущность-связь" для предприятия общественного питания.
БД должна:
 - хранить сведения о блюдах, видах блюд (первое, второе, десерт, напиток)
 - хранить сведения о составе и рецептуре блюд
 - хранить сведения о продуктах (в том числе о калорийности продуктов), из которых приготавливаются блюда
 - хранить сведения о поставщиках продуктов
 - учитывать потребление блюд за день
5. Построить модель «сущность-связь» для учета продажи билетов в кинотеатре.
БД должна:
 - хранить сведения о репертуаре на каждый день для каждого кинозала по сеансам
 - в кинотеатре 2 кинозала с 3-мя типами мест
 - стоимость билетов зависит от типа места и времени сеанса
 - учитывать продажу билетов за конкретный день
6. Построить модель «сущность-связь» для учета продажи авиабилетов.
БД должна:
 - хранить сведения о пассажирах
 - хранить сведения о рейсах (откуда, куда, время отправления, продолжительность рейса)
 - стоимость билетов зависит от класса (туристический, бизнес...)
 - учитывать продажу билетов за конкретный день
7. Как изменится модель «сущность-связь» для учета продажи авиабилетов, если необходимо учитывать транзитные рейсы (несколько промежуточных посадок)?

Реляционная модель данных

1. Преобразовать модели «сущность-связь», созданные в предыдущем задании, в реляционные модели
 - Предприятие общественного питания
 - Учет продажи билетов в кинотеатре
 - Учет продажи авиабилетов
2. Задать типы данных для атрибутов отношений, установить первичные и внешние ключи, определить допустимость Null-значений атрибутов.

Реляционная алгебра

1. Выразить операцию пересечения через примитивные операции алгебры
2. Выразить операцию естественного соединения через примитивные операции алгебры
3. Выразить операцию условного соединения через примитивные операции алгебры
4. Выразить операцию деления через примитивные операции алгебры
5. Используя операторы реляционной алгебры, записать выражения для получения отношений
 - 5.1. Даны три отношения с эквивалентными схемами:
Книга1 (ISBN, Автор, Название) – список книг в читальном зале, где ISBN – уникальный шифр книги
Книга2 (ISBN, Автор, Название) – список книг, выдаваемых на абонемент
Книга3 (ISBN, Автор, Название) – список книг на руках у читателей
Часть книг есть только в читальном зале, часть – только на абонементе, на руках у читателей могут находиться книги, выдаваемые на абонемент или взятые в читальном зале.
Выдать список книг:
 - а) имеющихся только в читальном зале

- б) имеющих и в читальном зале, и на абонементе
 - с) имеющих либо только в читальном зале, либо только на абонементе
 - д) имеющих либо только в читальном зале, либо только на абонементе и не взятых читателями
 - е) имеющих в читальном зале и на абонементе и взятых читателями
- 5.2. Даны отношения:
- Руководитель (КодРуководителя, ФИО, Адрес)
 Город (КодГорода, Город, Страна)
 Маршрут (КодМаршрута, Название, Продолжительность, Стоимость)
 СодержаниеМаршрута (КодМаршрута, КодГорода)
 Группа (КодГруппы, КодРуководителя, КодМаршрута, ДатаВыезда)
 Маршрут может включать посещение нескольких городов. Один и тот же город может встречаться в разных маршрутах. Руководитель может сопровождать разные группы по одинаковым или разным маршрутам.
- а) Выбрать названия маршрутов, стоимость которых меньше стоимости маршрута M15
 - б) Выбрать группы, дата выезда которых попадает в период с 25.10.04 по 20.12.04
 - с) Выбрать названия маршрутов, в которые входит посещение города “Харбин”
 - д) Выбрать ФИО руководителей, не работавших ни с одной группой
 - е) Выбрать ФИО руководителей, работавших со всеми маршрутами
 - ф) Выбрать группы, у которых совпадает дата выезда
 - г) Выбрать группы, у которых совпадает дата выезда, а продолжительность маршрута разная

Нормализация отношений

- Определить первичные (и альтернативные) ключи в отношениях
 - Выявить все функциональные зависимости в отношениях
 - Определить в какой нормальной форме находятся отношения
 - Преобразовать отношения к 4НФ
1. **Сотрудник**(ФИО, Адрес, Табельный №, Паспорт, Название проекта, Дата сдачи проекта)
сотрудник может работать с несколькими проектами
 2. **Деталь**(№ детали, Название, Цех, Материал)
одноименные детали могут быть изготовлены из разных материалов, в разных цехах
 3. **Деталь**(№ детали, Название, Завод-изготовитель, Страна завода)
 4. **Изделие**(Название, Вес изделия, Сырье, Вес сырья)
 5. **Дисциплины**(Специальность, Дисциплина, Преподаватель)
дисциплины могут читаться для разных специальностей, специальность включает множество дисциплин, преподаватель может читать разные дисциплины для разных специальностей
 6. **Студенты**(№ зачетки, ФИО, Дисциплина, Специальность)
 7. **Продажа**(Товар, Покупатель, Дата продажи, Склад)
в один день могут быть проданы разные товары разным покупателям
 8. **Билеты**(Дата, Время, Ряд, Место, Фильм, Жанр фильма)
 9. **Рейс**(№ рейса, Дата вылета, Время вылета, Продолжительность рейса, Пункт отправления, Пункт назначения)
рейс совершается не чаще 1 раза в день
 10. **Книга**(Автор, Название, Издательство)
книга может быть написана коллективом авторов
 11. **Читатель**(№ чит. билета, ФИО, Адрес, Дата рождения, Название книги, Дата сдачи)
читатель может брать до 5 книг, одна и та же книга может быть взята несколько раз

Запросы на QBE

1. Для БД предприятие общественного питания, реляционная модель для которого была разработана в предыдущих заданиях, составить запросы:
 - Вычислить калорийность всех блюд
 - Вычислить общий вес продуктов в блюдах
 - Вычислить общую стоимость каждой партии продуктов с 1 января по 23 марта
 - Сколько (количество) продуктов было поставлено в январе?
 - Во сколько блюд входит мука?
 - В состав каких блюд входит сахар?
 - Вывести состав блюда «Рассольник»

- Вывести поставщиков, поставлявших продукты в конкретном месяце
 - Вывести продукты, цена которых не превышает 50 руб (без повторений)
 - Какие виды блюд не заказывались в марте?
2. Для БД учета продажи билетов в кинотеатре, реляционная модель для которого была разработана в предыдущих заданиях, составить запросы:
- Вывести номера свободных мест на последний сеанс на сегодня
 - Вывести цены на билеты на второй сеанс
 - Вывести список фильмов на следующую неделю, отсортированных по дате и по названиям фильмов
 - Сколько билетов было продано 1 марта?
 - Сколько билетов сегодня остались не проданными на первый сеанс в 1-м кинозале?
 - Вычислить общую стоимость проданных билетов за текущий день
 - Билеты на какие типы мест самые дорогие?
 - Найти самый прибыльный и самый убыточный день в январе
 - Сколько фильмов шло в обоих кинозалах за год?
 - Какой фильм самый прибыльный?
3. Для БД учета продажи авиабилетов, реляционная модель для которого была разработана в предыдущих заданиях, составить запросы:
- Выдать список городов (по алфавиту), куда совершаются полеты
 - Выдать список рейсов на сегодня
 - Выдать список транзитных рейсов (рейсы, имеющие промежуточные посадки)
 - Выдать список прямых рейсов (рейсы, не имеющие промежуточные посадки)
 - Найти время вылета самого раннего рейса
 - Найти продолжительность самого долгого рейса
 - Сколько билетов было продано на конкретный рейс (указать рейс)
 - Вычислить стоимость билетов, проданных в марте
 - Найти самый прибыльный рейс за первый квартал текущего года
 - Составить рейтинг рейсов по стоимости проданных билетов за год, отсортированных по убыванию

Запросы на SQL

1. Для БД предприятие общественного питания, реляционная модель для которого была разработана в предыдущих заданиях, составить запросы:
- Выбрать информацию о блюдах с сортировкой названия блюда по алфавиту
 - Выбрать продукты, поставляемые одним из поставщиков (указать конкретного поставщика)
 - Выбрать блюда, которые потреблялись в феврале текущего года с сортировкой по алфавиту
 - Выбрать поставщиков ни разу не поставивших продукты
 - Выбрать блюда, в состав которых входят те же продукты, что и блюдо «Борщ» (можно указать свое название блюда)
2. Для БД учета продажи билетов в кинотеатре, реляционная модель для которого была разработана в предыдущих заданиях, составить запросы:
- Выбрать информацию о фильмах, идущих сегодня в 1-м кинозале
 - Посчитать общее количество проданных билетов за неделю
 - На какой фильм не продан ни один билет?
 - Найти общую стоимость билетов проданных сегодня на 1-й сеанс
 - Выбрать фильм максимальный по продолжительности
3. Для БД учета продажи авиабилетов, реляционная модель для которого была разработана в предыдущих заданиях, составить запросы:
- Выдать список рейсов отсортированных по времени вылета
 - Найти рейс с минимальной продолжительностью полета
 - Какие рейсы были отменены за последний месяц (не продано ни одного билета)
 - Посчитать общую стоимость билетов проданных за период с 1 февраля по 31 августа
 - Найти рейс вылетающий сегодня последним

Проектирование БД «Библиотека»

Пусть требуется разработать информационную систему для автоматизации учета получения и выдачи книг в библиотеке. Система должна предусматривать режимы ведения системного каталога, отражающего перечень областей знаний, по которым имеются книги в библиотеке. Внутри библиотеки области знаний в систематическом каталоге могут иметь уникальный внутренний номер и полное наименование. Каждая книга может содержать сведения из нескольких областей знаний. Каждая книга в библиотеке может присутствовать в нескольких экземплярах. Каждая книга, хранящаяся в библиотеке, характеризуется следующими параметрами:

- *уникальный шифр;*
- *название;*
- *фамилии авторов (могут отсутствовать);*
- *место издания (город);*
- *издательство;*
- *год издания;*
- *количество страниц;*
- *стоимость книги;*
- *количество экземпляров книги в библиотеке.*

Книги могут иметь одинаковые названия, но они различаются по своему уникальному шифру (ISBN).

В библиотеке ведется картотека читателей. На каждого читателя в картотеку заносятся следующие сведения:

- *фамилия, имя, отчество;*
- *домашний адрес;*
- *телефон (будем считать, что у нас два телефона – рабочий и домашний);*
- *дата рождения.*

Каждому читателю присваивается *уникальный номер читательского билета.*

Каждый читатель может одновременно держать на руках не более 5 книг. Читатель не должен одновременно держать более одного экземпляра книги одного названия.

Каждая книга в библиотеке может присутствовать в нескольких экземплярах. Каждый экземпляр имеет следующие характеристики:

- *уникальный инвентарный номер;*
- *шифр книги, который совпадает с уникальным шифром из описания книг;*

В случае выдачи экземпляра книги читателю в библиотеке хранится специальный вкладыш, в котором должны быть записаны следующие сведения:

- *номер билета читателя, который взял книгу;*
- *дата выдачи книги;*
- *дата возврата.*

Предусмотреть следующие ограничения на информацию в системе:

1. Книга может не иметь ни одного автора.
2. В библиотеке должны быть записаны читатели не моложе 17 лет.
3. В библиотеке присутствуют книги, изданные начиная с 1960 по текущий год.
4. Каждый читатель может держать на руках не более 5 книг.
5. Каждый читатель при регистрации в библиотеке должен дать телефон для связи: он может быть рабочим или домашним.
6. Каждая область знаний может содержать ссылки на множество книг, но каждая книга может относиться к различным областям знаний.

С данной информационной системой должны работать следующие группы пользователей:

- библиотекари;
- читатели;
- администрация библиотеки.

При работе с системой **библиотекарь** должен иметь возможность решать следующие задачи:

1. Принимать новые книги и регистрировать их в библиотеке.
2. Относить книги к одной или к нескольким областям знаний.
3. Проводить каталогизацию книг, то есть назначение новых инвентарных номеров вновь принятым книгам.
4. Проводить дополнительную каталогизацию, если поступило несколько экземпляров книги, которая уже есть в библиотеке, при этом информация о книге в предметный каталог не вносится, а каждому новому экземпляру присваивается новый инвентарный номер и для него определяется место на полке библиотеки.
5. Проводить списание старых и не пользующихся спросом книг. Списывать можно только книги, ни один экземпляр которых не находится у читателей. Списание проводится по специальному акту списания,

который утверждается администрацией библиотеки.

6. Вести учет выданных книг читателям, при этом предполагается два режима работы: выдача книг читателю и прием от него возвращаемых им книг обратно в библиотеку. При выдаче книг фиксируется, когда и какой экземпляр книги был выдан данному читателю и к какому сроку читатель должен вернуть этот экземпляр книги. При выдаче книг наличие свободного экземпляра и его конкретный номер могут определяться по заданному уникальному шифру книги или инвентарный номер может быть известен заранее. Не требуется вести «историю» чтения книг, то есть требуется отражать только текущее состояние библиотеки. При приеме книги, возвращаемой читателем, проверяется соответствие возвращаемого инвентарного номера книги выданному инвентарному номеру, и она ставится на свое старое место на полку библиотеки.
7. Проводить списание утерянных читателем книг по специальному акту списания или замены, подписанному администрацией библиотеки.
8. Проводить закрытие абонемента читателя, то есть уничтожение данных о нем, если читатель хочет выписаться из библиотеки и не является ее должником, то есть за ним не числится ни одной библиотечной книги.

Читатель должен иметь возможность решать следующие задачи:

1. Просматривать системный каталог, то есть перечень всех областей знаний, книги по которым есть в библиотеке.
2. По выбранной области знаний получить полный перечень книг, которые числятся в библиотеке.
3. Для выбранной книги получить инвентарный номер свободного экземпляра книги или сообщение о том, что свободных экземпляров книги нет. В случае отсутствия свободных экземпляров книги читатель должен иметь возможность узнать дату ближайшего предполагаемого возврата экземпляра данной книги. Читатель не может узнать данные о том, у кого в настоящий момент экземпляры данной книги находятся на руках (в целях обеспечения личной безопасности держателей требуемой книги).
4. Для выбранного автора получить список книг, которые числятся в библиотеке.

Администрация библиотеки должна иметь возможность получать сведения:

1. сведения о должниках – читателях библиотеки, которые не вернули вовремя взятые книги;
2. сведения о книгах, которые не являются популярными, т. е. ни один экземпляр которых не находится на руках у читателей;
3. сведения о стоимости конкретной книги, для того чтобы установить возможность возмещения стоимости утерянной книги или возможность замены ее другой книгой;
4. сведения о наиболее популярных книгах, то есть таких, все экземпляры которых находятся на руках у читателей.