РОССИЙСКАЯ ФЕДЕРАЦИЯ

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ

ГОУ ВПО ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ

Моор П.К., Моор А.П.

БАЗЫ ДАННЫХ. ПРАКТИКУМ

Учебное пособие

Издательство Тюменского государственного университета

Тюмень - 2007

П.К. Моор, А.П. Моор. БАЗЫ ДАННЫХ. ПРАКТИКУМ: Учебное пособие. Тюмень: Издательство Тюменского государственного университета, 2007. 130 с.

В учебном пособии рассматриваются некоторые теоретические вопросы построения баз данных и содержатся задания для практических занятий по дисциплине «Базы данных», которая имеет целью обучить студентов принципам хранения и обработки информации в автоматизированных системах. Учебное пособие предназначено для студентов специальности «351400-Прикладная информатика в экономике». Практикум нацелен на выработку у студентов навыков практической работы по построению моделей баз данных, создания баз данных и обработки информации в системах управления базами данных.

Ответственный редактор

Рецензенты: Т.И. Чалкова, канд. техн. наук, доцент кафедры математических методов в экономике ТГНГУ
Н.М. Гаврилова, канд. физ.- мат. наук, доцент кафедры программного обеспечения ТюмГУ

Ответственный за выпуск А.Г. Ивашко, д-р техн. наук, профессор

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Дисциплина “Базы данных” имеет целью обучить студентов принципам хранения, обработки и передачи информации в автоматизированных системах, показать им, что концепция баз данных стала определяющим фактором при создании эффективных систем автоматизированной обработки информации. Дисциплина является продолжением изучения информатики, методов программирования, основ вычислительной техники, объектно-ориентированного и визуального программирования, теории информационных систем. Знания и практические навыки, полученные в курсе “Базы данных” используются в дальнейшем при изучении ряда дисциплин, при разработке курсовых и дипломных работ.

Задачи дисциплины - дать основы проектирования баз данных и приложений баз данных.

В результате изучения дисциплины студенты должны:

- иметь представление о системе управления базами данных как об одной из основных составляющих эффективных систем автоматизированной обработки информации;
- знать области применения систем управления базами данных, этапы проектирования баз данных, характеристики и типы систем баз данных, средства поддержания целостности в базах данных;
- уметь выделять сущности и связи предметной области, отображать предметную область на конкретную модель данных, разрабатывать модель базы данных на основе процесса нормализации отношений, разрабатывать приложения баз данных на высокоуровневых языках программирования;
- иметь навыки работы в различных СУБД, работы со средствами поддержания интерфейса, разработчика баз данных.

Тема 1. РАЗРАБОТКА БАЗ ДАННЫХ И ПРИЛОЖЕНИЙ В СУБД MICROSOFT ACCESS

Задание 1.1. Накладная

Разработать базу данных и приложение Microsoft Access, автоматизирующее учет оптовой продажи товаров.

Учет товаров, отпускаемых со склада, осуществляется на основании накладной:

Накладная №	157
Дата	10/01/03
Покупатель	
Код	345
Наименование	ОАО «Сфера»
ИНН	072 001 000 6789
Адрес	г. Тюмень, ул. Ленина, 120

№	Товар		Количество	Ед. изм.		Цена	Стоимость
	наименование	код		код	наим.		
1	Сахар	0012	100	12	кг	15.00	1 500
2	Макароны	0015	200	23	пачка	13.00	2 600
3	Тушенка	0022	300	24	банка	25.00	7 500
	Всего		600				11 600
Кладовщик					Петрова		

При разработке базы данных считать, что номер накладной – уникален, для его формирования может быть использован счетчик.

Накладная выписывается на конкретного клиента.

В накладной строки нумеруются последовательно 1, 2, …. Каждая накладная содержит, по крайней мере, одну строку товаров.

Цена отпускаемого товара и единица измерения определяется кодом товара.

Требования к приложению:

1. Пользователь должен иметь доступ только к формам и отчетам (обеспечить при открытии базы данных автоматический вызов главной кнопочной формы).
2. Главная форма должна иметь кнопки вызова форм:

- «Товары», «Клиенты», «Кладовщики» и «Единицы измерения» для редактирования соответствующей информации и кнопки вызова форм «Новый товар» и «Новый клиент» и так далее, с помощью которых можно ввести информацию о новых товарах и клиентах и т. д.;
- «Накладные» - для отображения списка накладных с итоговыми показателями «количество строк в накладной» и «общая сумма»). Обеспечить возможность сортировки накладных по номеру накладной или клиентам (рис 1.1). В этой форме предусмотреть вызов форм:

для просмотра текущей накладной. В этой форме запретить редактирование любых данных;

для оформления новой накладной;

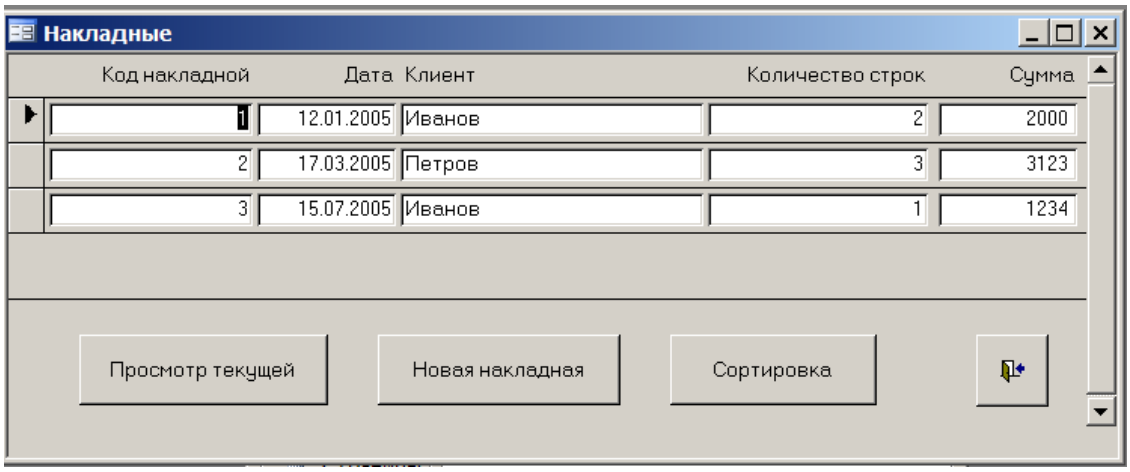


Рис. 1.1. Пример формы «Накладные»

- «По клиентам», «По товарам» и «По кладовщикам» (с подчиненными) – для отображения накладных по соответствующим значениям атрибутов. В формах подвести итоги (по количеству строк, количеству товаров, сумме).

3. Приложение должно допускать изменение текущей цены товара, при этом в накладных, оформленных ранее, цена на товар изменяться не должна. При оформлении новой накладной цена товара должна выбираться автоматически при выборе наименования товара из списка товаров, в качестве даты выписки накладной предлагать текущую дату.

4. Разработать отчеты:

   за указанный период времени,

   по клиентам,

   по товарам,

   по кладовщикам.

   В отчетах выполнить группировку данных и подвести итоги: сумма, среднее.

5. Таблица, в которой осуществляется хранение информации о купленных товарах, должна называться фамилией исполнителя.

Задание 1.2. Учет работы автотранспорта

Разработать базу данных и приложение Microsoft Access для автоматизации учета работы автотранспорта, осуществляющего доставку товаров в магазины.

Учет работы автотранспорта, осуществляется на основании путевого листа.

Путевой лист №	157		Цена бензина	17,00
Дата	10/01/05			
Водитель			Автомобиль	
Код	345		Код	333
Наименование	Фамилия И.О.		Гос. Номер	Т345АИ
Категория	С		Расход бензина на 1 км (л)	1

№	Объект		Расстояние	Расход бензина (л)	Стоимость горючего
	наименование	код			
1	Магазин № 2	0012			
2	Магазин № 5	0015			
	Всего				
Диспетчер				Иванов	

При разработке базы данных считать, что номер путевого листа – уникален, для его формирования может быть использован счетчик.

Путевой лист выписывается на одного водителя.

В путевом листе строки нумеруются последовательно 1, 2, …. Каждый путевой лист содержит, по крайней мере, одну строку.

Расстояние до объектов фиксировано.

Водитель может работать на разных автомобилях.

Расход горючего на 1 км пути определяется кодом автомобиля.

Цена 1 л бензина – 17 руб. Она может в некоторый момент измениться, но в путевых листах, оформленных ранее, она не должна меняется.

Требования к приложению:

6. Пользователь должен иметь доступ только к формам и отчетам (обеспечить при открытии базы данных автоматический вызов главной кнопочной формы).
7. Главная форма должна иметь кнопки вызова форм:

- «Водители», «Автомобили», «Объекты», «Диспетчеры», «Цена» для редактирования соответствующей информации и кнопки вызова форм «Добавить», с помощью которых можно ввести информацию о новых водителях, и т.д.;
- «Путевые листы» - для отображения списка путевых листов с итоговыми показателями «количество строк» «общий расход бензина», «общая стоимость бензина». Обеспечить возможность сортировки их по коду путевого листа или водителям. В этой форме предусмотреть вызов форм:

для просмотра текущего (выделенного) путевого листа. В этой форме запретить редактирование любых данных;

для оформления нового путевого листа;

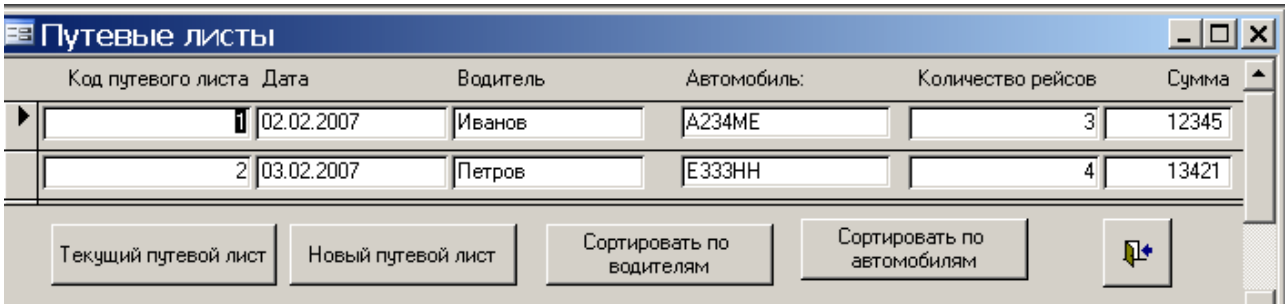


Рис. 1.2. Пример формы «Путевые листы»

- «По Водителям», «По автомобилям» и «По объектам» (с подчиненными) – для отображения информации по соответствующим значениям атрибутов. В формах подвести итоги.

8. Приложение должно допускать изменение текущей цены бензина, при этом в путевых листах, оформленных ранее, цена изменяться не должна. При оформлении нового путевого листа должна использоваться текущая цена и текущая дата.
9. Разработать отчеты:
   за указанный период времени,
   по водителям,
   по объектам.
   В отчетах выполнить группировку данных и подвести итоги: сумма, среднее.
10. Таблица, в которой осуществляется хранение информации о рейсах, должна называться фамилией исполнителя.

Задание 1.3. Учет работы сотрудников

Разработать базу данных и приложение Microsoft Access для учета работы сотрудников организации и начисления заработной платы (ЗП).

Учет работы сотрудников и начисление ЗП, осуществляется на основании ведомости:

Ведомость №	157	Дата	3/02/05
За период	Год	Месяц	Норматив (час.)
	2005	01	180
Бригада			
Код	345		
Наименование	Конструкторский		

№	Сотрудник		Должность			Отработано часов	Начислено ЗП
	Фамилия И.О.	Таб. номер	код	название	оклад		
1	Иванов И.И.	0012	1	мастер	12000	180	
2	Петров П.П.	0015	2	слесарь	10000	160	
3	Сидорова С.С.	0022	2	слесарь	10000	165	
	Всего						

При разработке базы данных считать, что номер ведомости – уникален, для его формирования может быть использован счетчик.

Ведомость оформляется один раз в месяц на каждую бригаду. В ведомости строки нумеруются последовательно 1, 2, …. Каждая ведомость содержит, по крайней мере, одну строку.

Состав бригады может меняться, он формируется из полного списка сотрудников.

Оклад определяется должностью (кодом).

Заработная плата вычисляется на основании оклада пропорционально отработанным часам.

Норматив рабочего времени устанавливается для каждого периода (год, месяц).

Требования к приложению:

1. Пользователь должен иметь доступ только к формам и отчетам (обеспечить при открытии базы данных автоматический вызов главной кнопочной формы).
2. Главная форма должна иметь кнопки вызова форм:

- «Сотрудники», «Бригады», «Должности», … для редактирования соответствующей информации и кнопки вызова форм «Добавить», с помощью которых можно ввести информацию о новых сотрудниках, должностях и т.д.;
- «Ведомости» - для отображения списка ведомостей с итоговыми значениями «количество сотрудников в ведомости», «общее отработанное время», суммарная заработная плата» (рис. 1.3). Обеспечить возможность сортировки накладных по номеру ведомости или бригадам.

В этой форме предусмотреть вызов форм:

для просмотра текущей (выделенной) ведомости. В этой форме запретить редактирование любых данных;

для оформления новой ведомости;

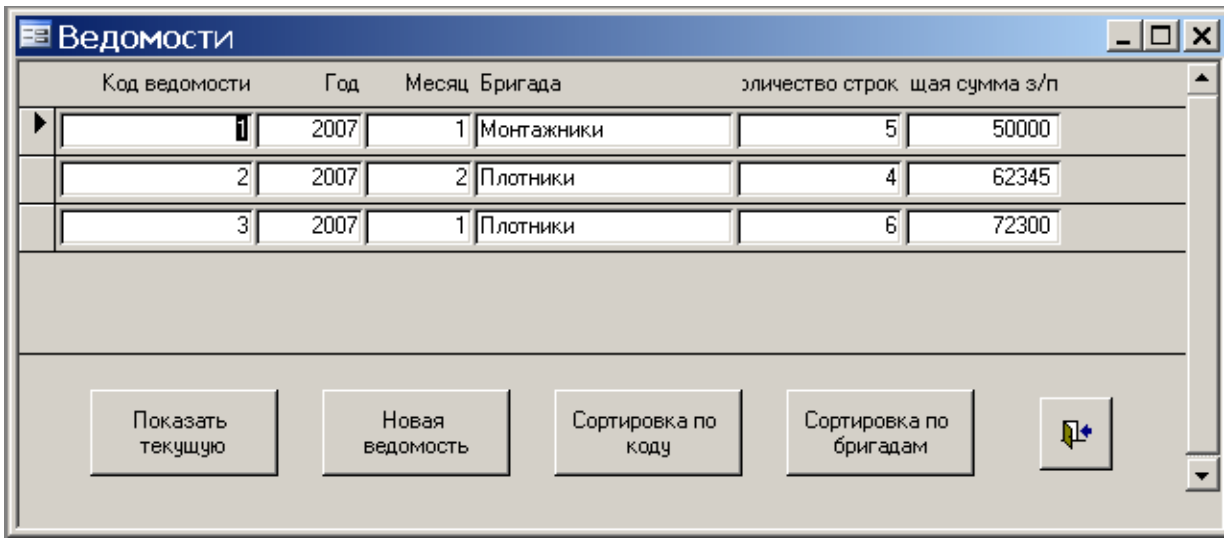


Рис. 1.3. Пример формы «Ведомости»

- «По сотрудникам», «По бригадам» и «По должностям» (с подчиненными) – для отображения ведомостей по соответствующим значениям атрибутов. В формах подвести итоги (по количеству сотрудников в бригаде, количеству отработанных часов, суммарной заработной плате).

3. Приложение должно допускать изменение окладов, при этом в ведомостях, оформленных ранее, заработная плата изменяться не должна. При оформлении новой ведомости должны использоваться текущие для должности оклады. При офрмлении новой ведомости в качестве даты предлагать текущую дату.
4. Разработать отчеты:

   за указанный период времени,

   по бригадам,

   по сотрудникам

   по должностям.

   В отчетах выполнить группировку данных, подвести итоги: сумма, среднее.
5. Таблица, в которой осуществляется хранение информации о работе сотрудников, должна называться фамилией исполнителя.

Задание 1.4. Учет нагрузки преподавателей

Разработать базу данных и приложение Microsoft Access для учета учебной нагрузки преподавателей вуза.

Учет нагрузки преподавателей, осуществляется на основании карточки:

Код преподавателя	157		Код кафедры	21
Фамилия И.О.	Иванов И.И.		Кафедра	ИС
Код должности	3			
Должность	доцент			
За период (уч. год)	2006/2007			

№	Дисциплина		Часов			Всего
	Код	Наименование	лекций	практ.	прочее	
1	111	Информатика	36	36	40	112
2	321	Базы данных				
3	121	Основы ВМ				
	Всего					

При разработке базы данных считать, что код карточки – уникален, для его формирования может быть использован счетчик.

Карточка оформляется для преподавателя каждый учебный год (то есть на одного преподавателя оформляется несколько карточек).

В карточке строки нумеруются последовательно 1, 2, …. Каждая карточка содержит, по крайней мере, одну строку.

Количество часов определяется кодом дисциплины и может различаться в разные годы.

Требования к приложению:

6. Пользователь должен иметь доступ только к формам и отчетам (обеспечить при открытии базы данных автоматический вызов главной кнопочной формы).
7. Главная форма должна иметь кнопки вызова форм:

- «Преподаватели», «Кафедры», «Должности», … для редактирования соответствующей информации и кнопки вызова форм «Добавить», с

помощью которых можно ввести информацию о новых дисциплинах, преподавателяхи т.д.;

- «Карточки» - для отображения списка карточек с возможностью сортировки их по номеру, по преподавателям или по кафедрам (рис. 1.4). В этой форме предусмотреть вызов форм:

для просмотра текущей (выделенной) карточки. В этой форме запретить редактирование любых данных;

для оформления новой карточки;

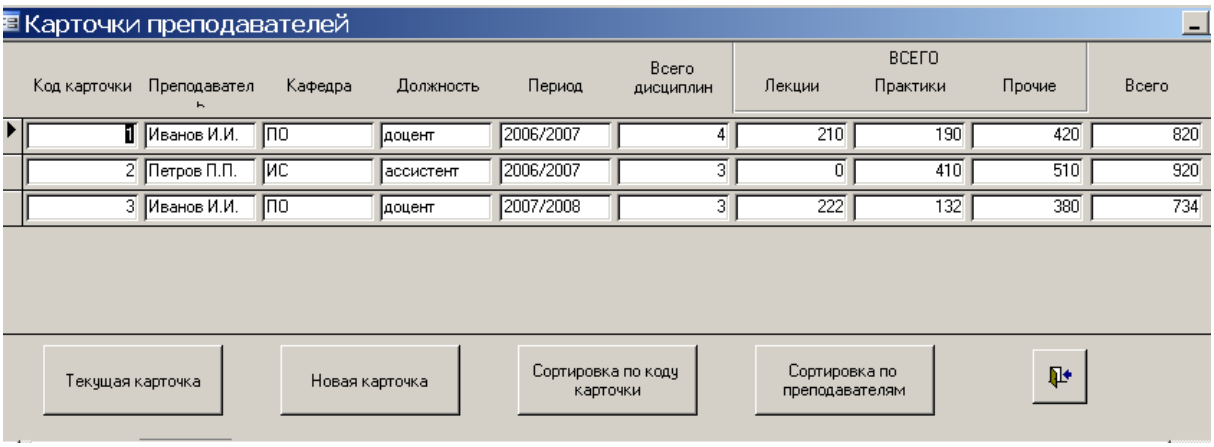


Рис. 1.4. Пример формы «Карточки»

- «По преподавателям», «По кафедрам» и «По должностям» (с подчиненными) – для отображения ведомостей по соответствующим значениям атрибутов. В формах подвести итоги (по количеству дисциплин, количеству часов по видам учебной нагрузки и общей сумме)

8. Приложение должно допускать изменение часов, отводимых на дисциплину, при этом в карточках, оформленных ранее, эти данные изменяться не должны. При оформлении новой карточки должны использоваться текущие для дисциплины значения часов.
9. Разработать отчеты:

   за указанный период времени,

   по кафедрам,

   по преподавателям.

В отчетах выполнить группировку данных и подвести итоги: сумма, среднее.

10. Таблица, в которой осуществляется хранение информации о нагрузке преподавателя по дисциплинам, должна называться фамилией исполнителя.

Тема 2. АРХИТЕКТУРА СИСТЕМ БАЗ ДАННЫХ

2.1. Системы баз данных

База данных – совместно используемое единое хранилище для некоторого набора логически связанных данных по определенной предметной области.

Система баз данных (database system) - компьютеризированная система хранения специально организованной информации по определенной предметной области.

Четыре главных компонента системы: **данные, аппаратное обеспечение, программное обеспечение** и **пользователи.**

Данные. Обычно данные в БД называют «постоянными». Под словом «постоянные» подразумеваются данные, которые в отличие от других, более изменчивых, таких как промежуточные результаты, результаты запросов и т.п.

Аппаратное обеспечение. К аппаратному обеспечению относятся: накопители для хранения информации, контроллеры устройств, память компьютера и процессор (или процессоры), которые используются для поддержки работы ПО.

Программное обеспечение. Система управления базами данных (СУБД) обеспечивает взаимодействие пользовательских систем и физических хранилищ данных. Основные функции, выполняемые СУБД – предоставление пользователю БД возможности работы с ней. Все запросы пользователя на доступ к БД обрабатываются СУБД. Наряду с СУБД в программное обеспечение входят также другие компоненты. Например: утилиты, средства разработки приложений, средства проектирования, генераторы отчетов.

Пользователи. Системы баз данных бывают однопользовательскими (одновременный доступ не более одного пользователя) и многопользовательские (возможен одновременный доступ к данным нескольких пользователей).

Пользователей можно разделить на четыре группы:

- администраторы данных;
- администраторы баз данных;
- прикладные программисты, которые отвечают за разработку прикладных программ, использующих базу данных;
- пользователи, которые работают с системами баз данных непосредственно через рабочую станцию или терминал;

2.2. Распределение обязанностей в системах баз данных

Администраторы данных (АД) отвечает за управление данными, включая планирование базы данных, разработку и сопровождение стандартов, бизнес-правил и деловых процедур, а также за концептуальное и логическое проектирование базы данных, контролирует соответствие общего направления развития БД корпоративным целям.

Администратор баз данных (АБД) отвечает за физическую реализацию базы данных, включая физическое проектирование и воплощение проекта, за обеспечение безопасности и целостности данных, за обеспечение максимальной производительности приложений.

Прикладные программисты занимаются разработкой приложений, предоставляющих пользователям необходимые функциональные возможности.

Пользователи являются потребителями БД. Можно условно их разделить на две подгруппы:

«опытные пользователи» - знакомы со структурой БД, могут использовать языки, создавать собственные приложения;

«наивные пользователи» обращаются к базе данных с помощью специальных приложений, используя меню и простейшие команды.

2.3. Трехуровневая архитектура ANSI – SPARC

Разработанный Национальным Институтом Стандартизации США (ANSI) и Комитетом планирования стандартов и норм (SPARC) стандарт терминологии и архитектуры БД признал необходимость использования трехуровневого

подхода. В соответствии с этим архитектура системы баз данных включает три уровня:

- внешний – уровень, на котором БД воспринимается пользователем (зависит от средства доступа к БД);
- концептуальный – «промежуточный» между внутренним и внешним;
- внутренний – это уровень наиболее близкий к физическому хранению, такой ее воспринимает операционная система и СУБД (но не физический уровень).

Внешний уровень – индивидуальный уровень пользователя, который имеет некоторый язык доступа к БД.

Концептуальный уровень – представление содержимого всей БД в абстрактной форме, его можно рассматривать как обобщение всех внешних уровней. Он описывает хранящиеся в БД данные, связи между ними и их ограничения, меры обеспечения доступа, безопасности и целостности данных.

Внутренний уровень – представление БД, в котором описывается размещение БД на внешнем носителе, типы хранимых записей, индексы, физическая последовательность хранимых записей, и т.п. Ниже внутреннего находится физический уровень, который контролируется операционной системой.

Основные соображения в пользу выбора трехуровневой модели архитектуры:

- каждый пользователь должен иметь возможность обращаться к данным, используя свое собственное представление о них (например, выбор языка программирования);
- пользователи не должны непосредственно иметь дело с техническими вопросами физического хранения данных (индексирования, сжатия и т.п.);
- администратор БД должен иметь возможность изменять структуру БД без влияния на пользователей;

- внутренняя структура не должна зависеть от изменений физических аспектов хранения информации, таких как, например, смена устройства;

При работе с БД осуществляется отображение концептуального уровня на внешний и внешнего на концептуальный (рис. 2.1).

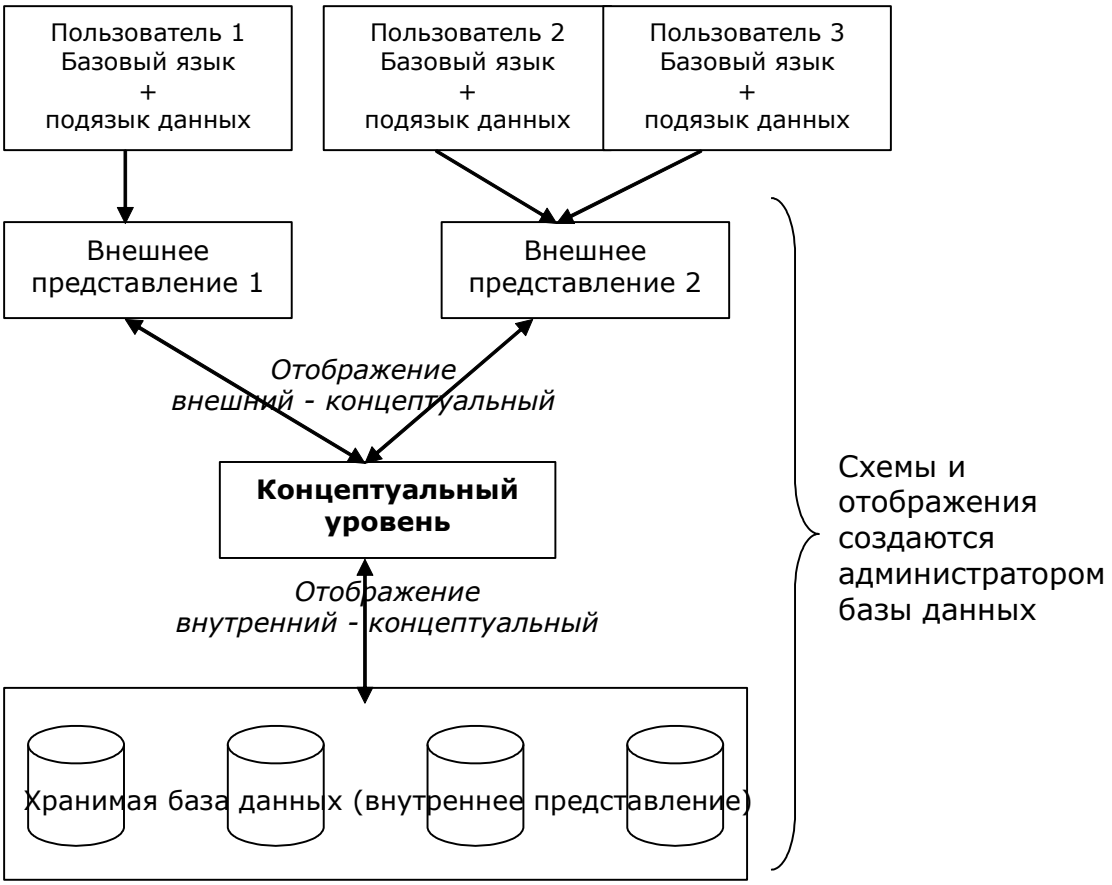


Рис. 2.1. Уровни архитектуры баз данных

Пример трех уровней БД о персонале компании приведён на рис. 2.2.

Внешнее представление 1:

Sno – личный номер сотрудника;

Fname – имя;

LName – фамилия;

Age – возраст;

Salary – заработная плата;

Внешнее представление 1:

S_N – личный номер сотрудника;

LName – фамилия;

Bno – номер отделения компании

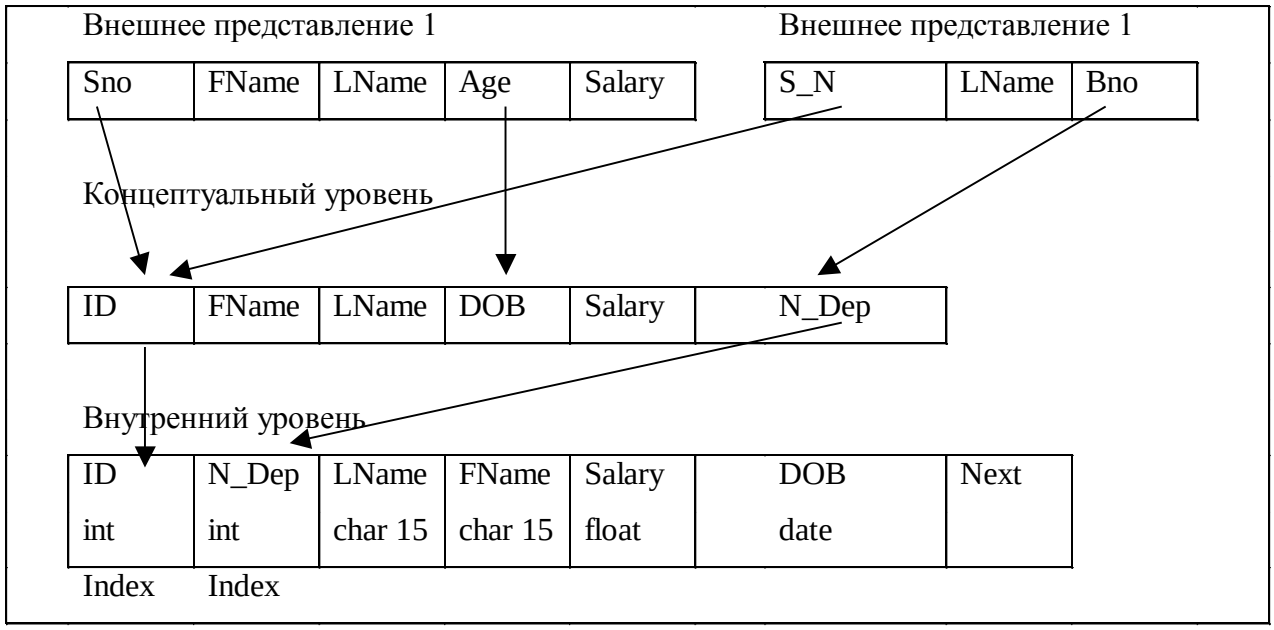


Рис. 2.2. Три уровня представления данных

Примечание. DOB – дата рождения, преобразуется в возраст.

На внутреннем уровне имеется указатель Next, связывающий физически все записи БД в единую цепочку. Последовательность полей может не совпадать.

С концептуальной схемой БД, в первую очередь, имеет дело администратор баз данных (АБД), который отвечает за общее управление системой и в его функции входит:

1. Определение концептуальной схемы;
2. Определение внутренней схемы;
3. Взаимодействие с пользователями;
4. Определение правил безопасности и целостности;
5. Определение процедур резервного копирования и восстановления;
6. Управление производительностью и реагирование на изменяющиеся требования.

Задания

Приведите примеры внутреннего, концептуального и внешних уровней для следующих баз данных:

1. «Студенты вуза»;
2. «Преподаватели вуза»;
3. «Отдел кадров»;
4. «Аренда объектов недвижимости»;
5. «Издательство книг».

Тема 3. ПЛАНИРОВАНИЕ, ПРОЕКТИРОВАНИЕ И АДМИНИСТРИРОВАНИЕ БАЗ ДАННЫХ

Структурный подход к разработке программного обеспечения называется **жизненным циклом информационных систем.** Этапы жизненного цикла.

Этап	Содержание
Планирование разработки базы данных	Подготовительные действия, позволяющие с максимально возможной эффективностью реализовать этапы жизненного цикла приложений баз данных.
Определение требований к системе	Определение диапазона действия и границ приложения базы данных, состава его пользователей и областей применения.
Сбор и анализ требований пользователей	Сбор и анализ информации о той части организации, работа которой будет поддерживаться с помощью создаваемого приложения базы данных, а также использование этой информации для определения требований пользователей к создаваемой системе
Проектирование базы данных	Процесс создания проекта базы данных, предназначенный для автоматизации функционирования предприятия и способствующий достижению его целей.
Выбор целевой СУБД	Выбор СУБД подходящего типа, предназначенной для поддержки создаваемого приложения базы данных.
Разработка приложений	Проектирование интерфейса пользователя и прикладных программ, предназначенных для работы с базой данных.
Создание прототипа	Создание рабочей модели БД и приложения баз данных.
Реализация	Физическая реализация базы данных и разработанных приложений.
Конвертирование данных	Перенос существующих данных в новую базу данных, загрузка и модификация существующих приложений с целью организации совместной работы с новой базой

Этап	Содержание
	данных.
Тестирование	Процесс выполнения прикладных программ с целью поиска ошибок
Эксплуатация и сопровождение	Наблюдение за системой и поддержка ее нормального функционирования по окончании развертывания

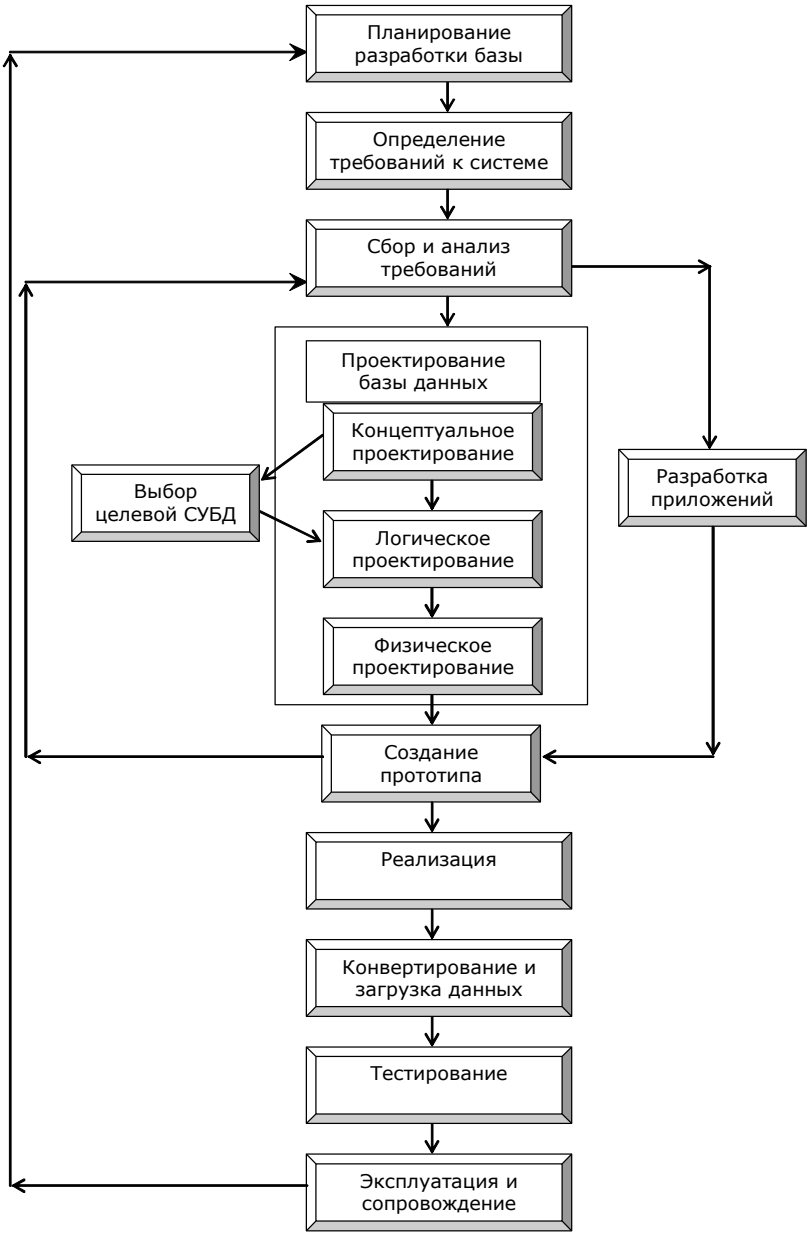


Рис. 3.1 Жизненный цикл приложения баз данных

Задания

Опишите этапы жизненного цикла информационных систем:

1. Информационной системы учета успеваемости студентов;
2. Информационной системы учета выдачи книг в библиотеке;
3. Информационной системы учета нагрузки преподавателей;
4. Информационной системы учета пациентов поликлиники;
5. Информационной системы учета продаж товаров;
6. Информационной системы учета работы студентов в компьютерных кабинетах;
7. Информационной системы «Расписание занятий в вузе»;
8. Информационной системы учета средств вычислительной техники.

Тема 4. РЕЛЯЦИОННАЯ МОДЕЛЬ БАЗ ДАННЫХ

4.1. Терминология

Отношение	Плоская таблица, состоящая из столбцов и строк В реляционной СУБД предполагается, что пользователь воспринимает БД как набор таблиц (и никак иначе). Отметим, что это восприятие относится только к внешнему и концептуальному уровню.
Атрибут	Поименованный столбец отношения
Домен	Набор допустимых значений для одного или нескольких атрибутов
Кортеж	Строка отношения
Степень	Количество атрибутов в отношении
Кардинальность, кардинальное число	Количество кортежей в отношении
Заголовок (содержание) отношения	Описание структуры отношения вместе с спецификацией доменов и другими ограничениями значений атрибутов
Реляционная база данных	Набор нормализованных отношений

4.2. Математические определение отношения

Пусть D_1, D_2, … D_n некоторые множества.

Декартовым произведением $D_1 \times D_2 \times \ldots \times D_n$ называется множество всех возможных n-ок , в которых первый элемент берется из D_1, второй – из D_2, … n – ый из D_n:

$$D_1 \times D_2 \times \ldots \times D_n = \{(x_1, x_2, \ldots) \mid x_1 \in D_1, x_2 \in D_2, \ldots x_n \in D_n\}$$

Любое подмножество $R \subset D_1 \times D_2 \times \ldots \times D_n$ является отношением.

Например, n=2, D_1={2,4} и D2={1,3,5},

$D_1 \times D_2$={(2,1), (2,3), (2,5), (4,1), (4,3), (4,5)},

R={(2,1), (4,1)}.

Подмножество может быть задано некоторым условием, например:

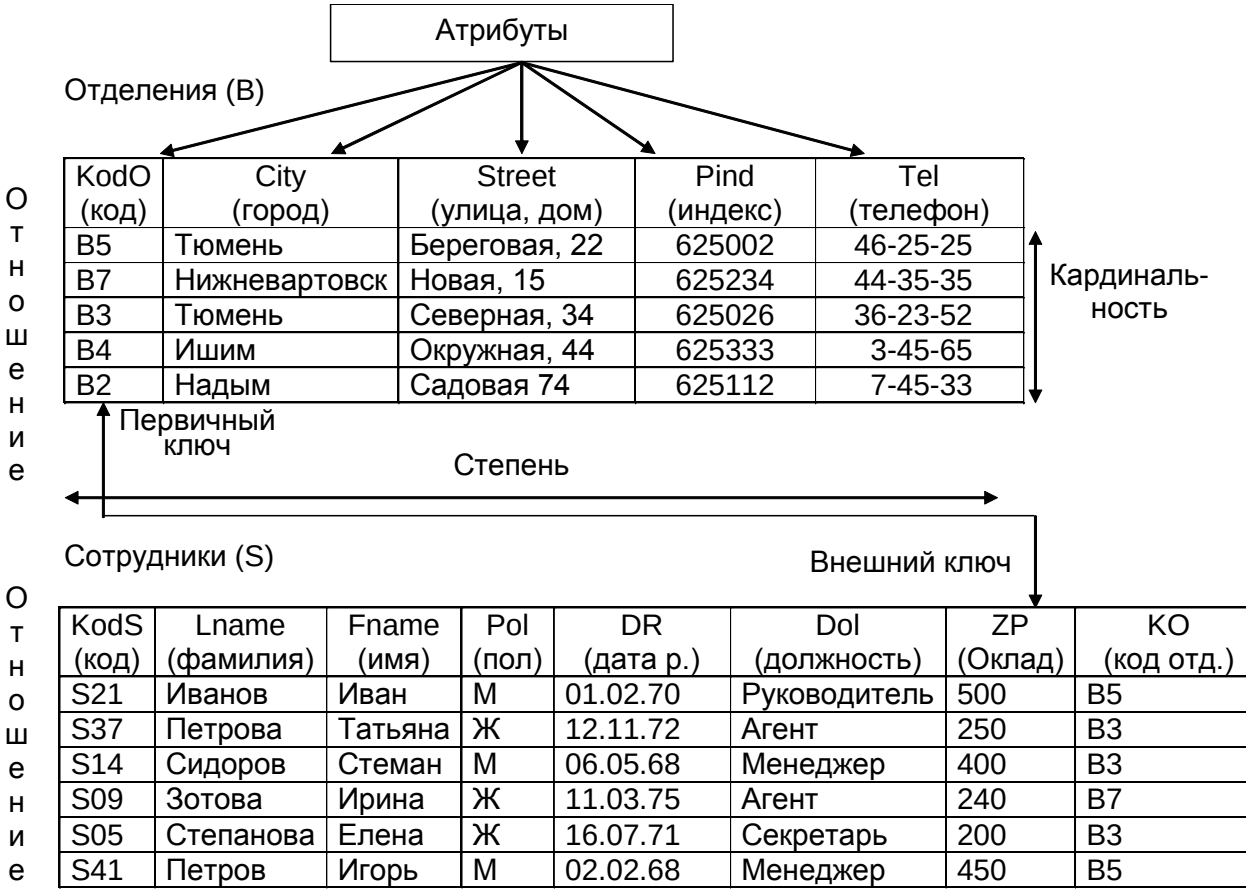
$R=\{(x_1,x_2,\ldots) \mid x_1 \in D_1, x_2 \in D_2, x_2 =1\}$.

Пусть имеются имена атрибутов A_1, A_2, … A_n с доменами D_1, D_2, … D_n, тогда отношение будем записывать в виде:

$R=\{A_1: D_1, A_2: D_2, \ldots A_n: D_n,\}$.

Свойства отношений:

- отношение имеет уникальное имя;
- каждый атрибут имеет уникальное имя (в отношении);
- каждая ячейка отношения содержит только атомарное значение и нет повторяющихся групп;
- порядок следования атрибутов не имеет никакого значения;
- каждый кортеж является уникальным;
- порядок следования кортежей произвольный.



KodO (код)	City (город)	Street (улица, дом)	Pind (индекс)	Tel (телефон)
B5	Тюмень	Береговая, 22	625002	46-25-25
B7	Нижневартовск	Новая, 15	625234	44-35-35
B3	Тюмень	Северная, 34	625026	36-23-52
B4	Ишим	Окружная, 44	625333	3-45-65
B2	Надым	Садовая 74	625112	7-45-33

KodS (код)	Lname (фамилия)	Fname (имя)	Pol (пол)	DR (дата р.)	Dol (должность)	ZP (Оклад)	KO (код отд.)
S21	Иванов	Иван	М	01.02.70	Руководитель	500	B5
S37	Петрова	Татьяна	Ж	12.11.72	Агент	250	B3
S14	Сидоров	Стеман	М	06.05.68	Менеджер	400	B3
S09	Зотова	Ирина	Ж	11.03.75	Агент	240	B7
S05	Степанова	Елена	Ж	16.07.71	Секретарь	200	B3
S41	Петров	Игорь	М	02.02.68	Менеджер	450	B5

Рис. 4.1. Иллюстрация терминов реляционной теории

4.3. Реляционные ключи

Реляционные ключи служат для уникальной идентификации каждого кортежа.

Суперключ	Атрибут или совокупность атрибутов, которые единственным образом идентифицирует кортеж.
Потенциальный ключ	Суперключ, который не содержит подмножества, образующего суперключ. Свойства потенциального ключа: уникальность и неприводимость. Отношение может иметь несколько потенциальных ключей.
Составной ключ	Ключ, составленный из нескольких атрибутов.
Первичный ключ	Потенциальный ключ, который выбран для уникальной идентификации кортежей внутри отношения.
Альтернативные ключи	Остальные потенциальные ключи.
Внешний ключ	Атрибут или совокупность атрибутов, которые соответствуют потенциальному ключу некоторого (может быть, того же) отношения.

4.4. Реляционная целостность

Определитель Null	Указывает, что значение атрибута не определено.
Целостность сущностей	В базовом отношении ни один атрибут первичного ключа не может содержать Null.
Ссылочная целостность	Если в отношении существует внешний ключ, то значение внешнего ключа должно соответствовать значению потенциального ключа, либо быть не определенным (Null).

4.5. Представления

Базовое	Поименованное отношение, соответствующее сущности в

отношение	концептуальной схеме, кортежи которого физически хранятся в базе данных
Представление	Динамический результат одной или нескольких реляционных операций над базовыми отношениями. Представление является виртуальным отношением, которое в базе данных реально не существует, но которое создается по требованию пользователя в момент поступления этого требования.

Назначение представлений. Использование представлений:

- предоставляет гибкий механизм защиты БД за счет сокрытия некоторой ее части от определенных пользователей (например, при просмотре списка сотрудников скрыть их заработные платы, запретить обычному пользователю изменение кода отделения);
- позволяет организовать доступ пользователей к данным наиболее удобным для них способом (в частности, выбирать атрибуты и имена атрибутов, удобные пользователю) Например, отображать не код отделения, а его название;
- позволяет упрощать сложные операции с базовыми отношениями. Например, каждому сотруднику предоставлять информацию об объектах недвижимости, которые он обслуживает.

4.6. Реляционная алгебра

Реляционная алгебра	Теоретический язык операций, который на основе одного или нескольких отношений позволяют создавать другое отношение без изменения самих исходных отношений.

Результат операции, таким образом, может использоваться в качестве операнда для другой операции, что позволяет создавать вложенные выражения (замкнутость реляционной алгебры).

Реляционная алгебра является языком, в котором все кортежи обрабатываются одной командой.

Определим пять основных операций:

- выборка,
- проекция,
- декартово произведение,
- объединение

На основе этих операций могут быть получены другие:

- соединение,
- пересечение,
- деление.

Операции выборки и проекции – унарные операции, остальные операции – бинарные.

Выборка

$\sigma_{предикат}(R)$	Операция выборки определяет отношение, которое содержит только те кортежи отношения R, которые удовлетворяют заданному условию (предикату). В предикате могут использоваться знаки логических операций $\wedge$ (And), $\vee$ (Or), ~ (Not).

Пример. Получить список всех сотрудников с окладом более 300

$\sigma_{ZP>300}(S)$

KodS	Lname	Fname	Pol	DR	Dol	ZP	KO
S21	Иванов	Иван	M	01.02.70	Руководитель	500	B5
S14	Сидоров	Стеман	M	06.05.68	Менеджер	400	B3
S41	Петров	Игорь	M	02.02.68	Менеджер	450	B5

Проекция

$\Pi_{атр1,\ldots,атрn}(R)$	Операция проекции определяет отношение, атрибутами которого являются $атр_1$, … $атр_n$ и содержащее только

	уникальные кортежи (из результата исключаются строки-дубликаты)

Пример. Получить список всех городов, в которых имеются отделения.

$\Pi_{City}(B)$

Результат:

City
Тюмень
Нижневартовск
Ишим
Надым

Декартово произведение

$R \times S$	Операция декартового произведения определяет новое отношение, которое определяется результатом конкатенации (сцепления) каждого кортежа из отношения R с каждым кортежем из отношения S

Пример. Получить список всех арендаторов, осмотревших объекты недвижимости.

Рассмотрим декартово произведение

$(\Pi_{KodR,Name}(R)) \times (\Pi_{KodR,KodP,Comment}(V))$

Результат:

KodR	Name	V.KodR	KodP	Comment
R76	Саблев Иван	R56	P14	Мала
R76	Саблев Иван	R76	P04	Далеко
R76	Саблев Иван	R56	P04	
R76	Саблев Иван	R62	P14	Дорого
R76	Саблев Иван	R56	P36	
R56	Рубин Степан	R56	P14	Мала
	И т.д.			

В таком виде отношение содержит много лишней информации. Для получения искомого списка необходимо применить операцию выборки с предикатом P.KodR = V.KodR:

$\sigma_{KodR = V.KodR}((\Pi_{KodR,Name}(R)) \times (\Pi_{KodR,KodP,Comment}(V)))$

Результат этих операций:

KodR	Name	V.KodR	KodP	Comment
R76	Саблев Иван	R76	P04	Далеко
R56	Рубин Степан	R56	P14	Мала
R56	Рубин Степан	R56	P04	
R56	Рубин Степан	R56	P36	
R62	Зимина Елена	R62	P14	Дорого

Объединение

$R \cup S$	Операция объединения определяет новое отношение, в которое включены все кортежи из R и S, с удалением дублирующих кортежей. При этом отношения R и S должны быть **совместимы по объединению** – имеют одинаковые атрибуты с совпадающими доменами

Пример. Получить список городов, в котором у агентства имеется отделение или объект недвижимости.

$\Pi_{City}(B) \cup \Pi_{City}(P)$

Результат:

City
Тюмень
Нижневартовск
Ишим
Надым
Сургут

Разность

R-S	Операция разности определяет новое отношение, в которое включены кортежи из R, которые отсутствуют в отношении S. При этом отношения R и S должны быть **совместимы по объединению**

Пример. Получить список городов, в котором у агентства имеется недвижимость, но нет отделения.

$\Pi_{City}(P) - \Pi_{City}(B)$

Результат:

City
Сургут

Операции соединения

Как правило, пользователя интересует лишь некоторая часть кортежей декартового произведения: удовлетворяющие заданному предикату. Для этого используется одна из самых важных операций – соединения. Существует несколько различных операций соединения:

- тета – соединение;
- естественное соединение;
- внешнее соединение;
- полусоединение.

Тета-соединение.

$R \mid><\mid_F S$	Операция тета-соединение определяет отношение, которое содержит кортежи из их декартового произведения, удовлетворяющие предикату F. Предикат F имеет вид $R.a_i \Theta S.b_i$, где Θ - один из знаков < | <= … Если предикат F содержит только =, то соединение называется **соединением по эквивалентности.**

Отношение соединения можно записать на основе базовых операций:

$R \mid><\mid_F S = \sigma_F(R \times S)$

Пример. Получить список всех арендаторов, осмотревших объекты недвижимости

$(\Pi_{KodR,Name}(R)) \mid><\mid_{Renter.KodR= V.KodR} (\Pi_{KodR,KodP,Comment}(V))$

Естественное соединение.

$R \mid><\mid S$	Операция естественного соединения – операция соединения по эквивалентности, выполненное по всем общим атрибутам, из результатов которого исключаются по одному экземпляру каждого общего атрибута

Пример. Получить список всех арендаторов, осмотревших объекты недвижимости

$(\Pi_{KodR,Name}(R))$ |><| $(\Pi_{KodR,KodP,Comment}(V))$

Результат:

KodR	Name	KodP	Comment
R76	Саблев Иван	P04	Далеко
R56	Рубин Степан	P14	Мала
R56	Рубин Степан	P04	
R56	Рубин Степан	P36	
R62	Зимина Елена	P14	Дорого

Внешнее соединение

$R \supset\!\lhd S$	Левым внешним соединением называется соединение, при котором кортежи отношения R, не имеющие совпадающих значений общих атрибутов в отношении S, также включаются в результирующее отношение. Для отсутствующих значений используется Null

Преимущество: при внешнем соединении сохраняется вся исходная информация из отношения R.

Аналогично можно определить также **правое внешнее соединение** и **полное внешнее соединение.**

Пример. Создать отчет о проведении осмотров объектов недвижимости.

$\Pi_{KodR,Street,City}(P) \supset \lhd (V))$

Результат:

KodP	KodR	Дата	(Заключение)
P14	R56	24.05.01	Мала
P04	R76	20.04.01	Далеко
P04	R56	26.05.01	
P14	R62	14.05.01	Дорого
P36	R56	28.04.01	

Полусоединение

$R \mid>_F S$	Операция полусоединения определяет отношение, которое содержит те кортежи отношения R, которые входят в соединение отношений R и S

Операцию полусоединения можно определить с помощью операторов проекции и соединения:

$R \mid>_F S = \Pi_A (R \mid><\mid_F S)$,

где A – набор всех атрибутов в отношении R.

Пример. Создать отчет, содержащий полную информацию о всех сотрудниках, работающих в отделении, расположенном в г. Тюмени.

$S \mid>_{S.KodO = B.KodO \text{ and } B.City='Тюмень'} B$

Результат:

KodS (код)	Lname (фамилия)	Fname (имя)	Pol (пол)	DR (дата р.)	Dol (должность)	ZP (Оклад)	KodO
S21	Иванов	Иван	М	01.02.70	Руководитель	500	B5
S37	Петрова	Татьяна	Ж	12.11.72	Агент	250	B3
S14	Сидоров	Стеман	М	06.05.68	Менеджер	400	B3
S05	Степанова	Елена	Ж	16.07.71	Секретарь	200	B3
S41	Петров	Игорь	М	02.02.68	Менеджер	450	B5

Пересечение

$R \cap S$	Операция пересечения определяет отношение, которое содержит кортежи, присутствующие как в отношении R, так и в отношении S. Отношения R и S должны быть совместны по объединению.

Операцию пересечения можно записать через базовые операции:

$R \cap S=R-(R-S)$

Деление

Определение 11. Пусть даны отношения $A(X_1,X_2,\ldots X_n,Y_1,Y_2,\ldots Y_m)$ и $B(Y_1,Y_2,\ldots Y_m)$, причем атрибуты $Y_1,Y_2,\ldots Y_m$ - общие для двух отношений. ***Делением отношений*** A на B называется отношение с заголовком $(X_1,X_2,\ldots X_n)$ и телом, содержащим множество кортежей $x_1,x_2,\ldots x_n$, таких, что для всех кортежей $y_1,y_2,\ldots y_m \in B$ в отношении A найдется кортеж $(x_1,x_2,\ldots x_n,y_1,y_2,\ldots y_m)$.

Пусть $R(A_1,A_2,\ldots,A_n,\ B_1,B_2,\ldots,B_m)$, $S(B_1,B_2,\ldots,B_m)$, причем атрибуты $B_1,B_2,\ldots,B_m$ – общие.

$R \div S$	Операция деления определяет отношение с атрибутами $A_1,A_2,\ldots,A_n$, которое содержит множество кортежей $(x_1,x_2,\ldots x_n)$, таких, что для всех кортежей $(y_1,y_2,\ldots y_m)$ из S в отношении R найдется кортеж $(x_1,x_2,\ldots x_n,y_1,y_2,\ldots y_m)$.

Операцию деления можно записать через базовые операции:

$T1=\Pi_C(R)$

$T2=\Pi_C((S\times T1)-R)$

$R \div S=T1-T2$

Пример. Создать список арендаторов, осмотревших объекты недвижимости с 4 комнатами.

$(\Pi_{kodR,KodP}(V)) \div (\Pi_{kodP}(\sigma_{Rooms=4}(P)))$

Результат:

KodR	KodP
R76	P04
R56	P04

Задания 4

Задание 4.1.

Пусть имеется отношение S (студенты) с первичным ключом NZ:

S (студенты)

Фамилия ИО	№ зач. книжки	Группа	Специальность	Курс
F	NZ	G	SP	K
Иванов И.И	345001	345	ПИЭ	3
Петров П.П.	346002	346	МОАИС	3
Сидоров С.С.	345012	345	ПИЭ	3
Сидоров С.В.	346003	346	МОАИС	3
Степанов М.К.	355017	355	ПИЭ	2
Десятова Н.Н.	355019	356	МОАИС	2

С помощью операций реляционной алгебры получить отношения, в которых будет содержаться следующая информация:

1. Список фамилий студентов и номера их зачетных книжек;
2. Список групп;
3. Список специальностей;
4. Список групп;
5. Список специальностей и групп;
6. Список студентов (F, NZ, G) второго курса;
7. Список студентов (F, NZ, G) специальности ‘ПИЭ’;
8. Список студентов (F, NZ, K) 345 группы;
9. Список студентов (F, NZ, G) третьего курса специальности ‘ПИЭ’;
10. Список студентов групп 345 и 355.

Привести полученные отношения. Сколько кортежей будет в полученных отношениях?

Задание 4.2.

Пусть имеются отношения:

S(F,NZ,G,SP) – список всех студентов;

S1(F,NZ,G,SP) – список студентов 1 курса;

S2(F,NZ,G,SP) – список студентов 2 курса;

S34(F,NZ,G,SP) – список студентов 3 и 4 курсов;

S34(F,NZ,G,SP) – список студентов 4 и 5 курсов.

Структура всех отношений одинаковая:

S

Фамилия ИО	№ зач. книжки	Группа	Специальность
F	NZ	G	SP

Всего имеется 5 курсов

С помощью операций реляционной алгебры получить отношения, в которых будет содержаться следующая информация:

1. Список фамилий студентов и номера их зачетных книжек студентов первого и второго курсов;
2. Список фамилий студентов и номера их зачетных книжек студентов, которые обучаются на 2, 3, 4 или 5 курсе;
3. Список фамилий студентов и номера их зачетных книжек студентов, которые обучаются на 3, 4 или 5 курсе;
4. Список студентов 4 курса.

Задание 4.3.

Пусть имеются отношения:

S(F,NZ,G,KODSP) – список студентов;

SP(F,NZ,G,SP) – список специальностей.

S

Фамилия ИО	№ зач. книжки	Группа	Код специальности
F	NZ	G	KODSP

SP

Код специальности	Специальность
KODSP	SP

С помощью операций реляционной алгебры получить отношение, в котором будет содержаться информация о студенте и специальности (SP), на которой он обучается.

В таблице SP имеется информация о специальностях, на которых не обучается ни одного студента. С помощью операций реляционной алгебры получить отношение, в котором будет содержаться информация о студенте и специальности (SP), на которой он обучается. Если на специальности не обучаются студенты, она должна присутствовать в результирующем отношении. Значения соответствующих студенту атрибутов должны быть не определены.

Задание 4.4.

Пусть имеется отношение S (сотрудники) с первичным ключом TN:

S (сотрудники)

Фамилия ИО	Табельный номер	Отдел	Должность	Стаж работы
F	TN	G	D	ST
Иванов И.И	345001	Участок 1	слесарь	3
Петров П.П.	346002	Участок 2	мастер	3
Сидоров С.С.	345012	Участок 1	сварщик	1
Сидоров С.В.	346003	Участок 2	мастер	3
Степанов М.К.	355017	Участок 1	слесарь	2
Десятова Н.Н.	355019	Участок 2	слесарь	2

С помощью операций реляционной алгебры получить отношения, в которых будет содержаться следующая информация:

1. Список фамилий сотрудников и их табельные номера;
2. Список отделов;
3. Список должностей;
4. Список отделов и должностей;
5. Список сотрудников (F, TN, G), имеющих стаж работы более 2 лет;
6. Список сотрудников (F, TN, G) отдела ‘Участок 1’;
7. Список сотрудников (F, TN, G), работающих в должности ‘мастер’;
8. Список сотрудников (F, TN, G) отдела ‘Участок 1’, работающих в должности ‘слесарь’;

Привести полученные отношения. Сколько кортежей будет в полученных отношениях?

Задание 4.5.

Пусть имеется отношения:

S(F,TN,G,D,ST) – список всех сотрудников;

S1(F,TN,G,D,ST) – список всех сотрудников участка 1;

S2(F,TN,G,D,ST) – список всех сотрудников участка 2;

S23(F,TN,G,D,ST) – список всех сотрудников участка 2 и участка 3;

S34(F,TN,G,D,ST) – список всех сотрудников участка 3 и участка 4.

Всего на предприятии имеется 4 участка.

Структура всех отношений одинакова: они совместимы по объединению:

S (сотрудники)

Фамилия ИО	Табельный номер	Отдел	Должность	Стаж работы
F	TN	G	D	ST

С помощью операций реляционной алгебры получить отношения, в которых будет содержаться следующая информация:

1. Список фамилий и их табельные номера сотрудников, работающих в отделах «Участок 1» и «Участок 2»;
2. Список фамилий и их табельные номера сотрудников, работающих в отделе «Участок 3»;
3. Список фамилий и их табельные номера сотрудников, работающих в отделе «Участок 4»;

Задание 4.6.

Пусть имеются отношения:

S(F,TN, KODO,D,ST) – список всех сотрудников предприятия;

OTD(KODO, OTDEL) – список отделов предприятия.

S (сотрудники)

Фамилия ИО	Табельный номер	Код отдела	Должность	Стаж работы
F	TN	KODO	D	ST

OTD (отделы)

Код отдела	Отдел
KODO	OTDEL

С помощью операций реляционной алгебры получить отношение, в котором будет содержаться информация о сотруднике, его табельном номере и отделе (OTDEL), в котором он работает.

В таблице OTD имеется информация об отделах, в которых не работает ни одного сотрудника. С помощью операций реляционной алгебры получить отношение, в котором будет содержаться информация о сотрудниках и отделах (OTDEL), в которых они работают. Если в отделе не работает ни одного сотрудника, отдел должен присутствовать в результирующем отношении. Значения соответствующих сотруднику атрибутов должны быть не определены.

ТЕМА 5. ПОСТРОЕНИЕ МОДЕЛИ БАЗЫ ДАННЫХ НА ОСНОВЕ НОРМАЛИЗАЦИИ

5.1. Цель нормализации

Для разработки логической модели данных, создания точного представления данных, связей между ними и требуемых ограничений (в т.ч. целостности) необходимо определить, прежде всего, подходящий набор отношений. Для решения этой задачи используется специальный метод, который называется **нормализацией.**

Нормализация	Метод создания набора отношений с заданными свойствами на основе заданных требований к данным

Процесс нормализации был впервые предложен Э. Ф. Коддом в 1972г. Нормализация осуществляется в виде последовательности тестов для отношения с целью проверки его соответствия (или несоответствия) требованиям заданной нормальной формы. Сначала были предложены три вида нормальных форм: первая (1НФ), вторая (2НФ) и третья (3НФ). Затем Р. Бойсом и Э. Ф. Коддом было сформулировано более строгое определение третьей нормальной формы, которое получило название нормальной формы Бойса-Кодда (НФБК). Все эти нормальные формы основаны на функциональных зависимостях, существующих между атрибутами отношения.

Вслед за НФБК появились определения четвертой (4НФ) и пятой (5НФ) нормальных форм. Однако на практике эти нормальные формы более высоких порядков используются крайне редко.

Процесс нормализации и является формальным методом распределения атрибутов по отношениям с образованием реляционной схемы, он позволяет идентифицировать отношения на основе их первичных ключей или потенциальных ключей и функциональных зависимостей, существующих между их атрибутами.

Проектировщики баз данных используют процесс нормализации в виде наборов тестов, применяемых к отдельным отношениям с целью нормализации

реляционной схемы до заданной конкретной формы, что позволяет предотвратить избыточность данных и аномалий обработки данных.

5.2. Избыточность данных и аномалии обработки данных

Избыточность данных и возможные аномалии обработки данных, можно проиллюстрировать, сравнив отношения Сотрудники (S) и Отделения (B), с отношением S_B, показанным в табл. 5.1.

Таблица 5.1

Отношение B

KodO	Address
B5	Тюмень, Береговая, 22
B7	Нижневартовск, Новая, 15
B3	Тюмень, Северная, 34
B4	Ишим, Окружная, 44
B2	Надым, Садовая 74

Отношение S

KodS	Name	Dol	ZP	KodO
S21	Иванов	Руководитель	500	B5
S37	Петрова	Агент	250	B3
S14	Сидоров	Менеджер	400	B3
S09	Зотова	Агент	240	B7
S05	Степанова	Секретарь	200	B3
S41	Петров	Менеджер	450	B5

Отношение S_B

KodS	Name	Dol	ZP	KodO	Address
S21	Иванов	Руководитель	500	B5	Тюмень, Береговая, 22
S37	Петрова	Агент	250	B3	Тюмень, Северная, 34
S14	Сидоров	Менеджер	400	B3	Тюмень, Северная, 34
S09	Зотова	Агент	240	B7	Нижневартовск, Новая, 15
S05	Степанова	Секретарь	200	B3	Тюмень, Северная, 34
S41	Петров	Менеджер	450	B5	Тюмень, Береговая,22

Избыточность данных

В отношении S_B содержатся избыточные данные, поскольку сведения об отделении компании повторяются в записях, относящихся к каждому сотруднику данного отделения. В противоположность этому, в отношении B сведения об отделении содержатся только в одной строке, а в отношении S повторяется только номер отделения компании (KodO), который представляет собой место работы каждого сотрудника.

Аномалии вставки

Аномалии вставки проявляются в том, что при необходимости внести в базу данных некоторую информацию нам не удается это сделать, не нарушая целостность данных.

Например, для вставки в отношении S_B сведений о новом отделении компании, которое еще не имеет собственных сотрудников, потребуется присвоить значение NULL всем атрибутам описания персонала, включая и личный номер. Однако, поскольку этот атрибут является первичным ключом отношения S_B, то попытка ввести значение NULL вызовет нарушение целостности сущностей и потому будет отклонена. Структура отношений, представленных в S и B, позволяет избежать возникновения этой проблемы, поскольку сведения об отделениях компании вводятся в отношение B независимо от ввода сведений о сотрудниках.

Аномалии удаления

Аномалии вставки проявляются в том, что при удалении из базы данных некоторой информации одновременно будет удалена и другая информация.

Например, при удалении из отношения S_B строки с информацией о последнем сотруднике некоторого отделения компании, будут полностью удалены из базы данных и сведения об этом отделении. Структура отношений S и B позволяет избежать возникновения этой проблемы, поскольку строки со

сведениями об отделениях компании хранятся отдельно от строк со сведениями о сотрудниках.

Аномалии обновления

Аномалии обновления проявляются в том, что при необходимости изменения значения одной характеристики сущности необходимо произвести изменение данных в нескольких кортежах.

Например, при изменения значения одного из атрибутов для некоторого отделения компании в отношении S_B (например, адреса) необходимо обновить соответствующие значения в строках для всех сотрудников этого отделения.

5.3. Функциональные зависимости

Функциональная зависимость (functional dependency) описывает связь между атрибутами и является одним из основных понятий нормализации.

Функциональная зависимость	Описывает связь между атрибутами отношения. Функциональная зависимость является смысловым (или семантическим) свойством атрибутов отношения. Если каждое значение атрибута (или группы атрибутов) A связано только с одним значением атрибута B (или группы атрибутов), то говорят, что атрибут B функционально зависит от атрибута A. Функциональная зависимость обозначается: A→B.
Детерминант	Детерминантом функциональной зависимости называется атрибут или группа атрибутов, расположенная слева от символа стрелки.

Пример. Функциональные зависимости.

В отношении S имеется функциональная зависимость:

KodS→Dol, (но не Dol→KodS);

В отношении S_B имеются функциональные зависимости:

KodS→Name;

KodO→Address.

Все атрибуты, которые не являются частью первичного ключа (потенциального ключа) функционально зависят от него.

5.4. Процесс нормализации

Нормализация — это формальный метод анализа отношений на основе их первичного ключа или потенциальных ключей и существующих функциональных зависимостей. Он включает ряд правил, которые могут использоваться для проверки отдельных отношений таким образом, чтобы вся база данных стала нормализована до желаемой степени. Если некоторое требование не удовлетворяется, то нарушающее данное требование отношение должно быть декомпозировано на отношения, каждое из которых удовлетворяет всем требованиям нормализации.

Нормализация осуществляется в несколько последовательно выполняющихся этапов, каждый из которых соответствует некоторой нормальной форме, обладающей известными свойствами. При работе с реляционной моделью данных важно понимать, что только удовлетворение требований первой нормальной формы (1НФ) обязательно для создания отношений приемлемого качества. Все остальные формы могут использоваться по желанию проектировщиков. Однако для того чтобы избежать аномалий обновления, нормализацию рекомендуется выполнять как минимум до ЗНФ.

5.5. Первая нормальная форма (1НФ)

Ненормализованная форма	Таблица, в которой на пересечении каждой строки и каждого столбца содержит несколько значений или содержащая одну или несколько повторяющихся (ННФ) групп данных.
Первая нормальная	Отношение, в котором на пересечении каждой строки

форма (1НФ)	и каждого столбца содержится только одно значение.

Например, пусть необходимо образовать отношение, в котором будет храниться информация об успеваемости студентов. Атрибуты определены:

Name – на домене D1 – множество студентов;

Dis – на домене D2 – множество дисциплин;

Z – на домене D3 – множество оценок $\{2, 3, 4, 5\}$.

Ненормализованными будут отношения в следствии:

неатомарных значений (вторая оценка – результат пересдачи экзамена):

Name	Dis	Z
Иванов	Математика	2 3
…		

повторяющихся групп:

Name	Dis_1	Z_1	Dis_2	Z_2
Иванов	Математика	5	Информатика	4
…				

или

Name	Dis	Z
Иванов	Математика Информатика	5 4
…		

Процесс нормализации начинается с преобразования данных из формата источника (например, из формата стандартной формы) в формат таблицы со строками и столбцами. На исходном этапе таблица может находиться в ненормализованной форме**.**

Рассмотрим пример приведения отношений к первой нормальной форме данных об аренде помещений. Пусть в агентстве для каждого арендатора заводится форма (карточка) учета аренды помещений.

Форма содержит сведения об объектах недвижимости, арендованных клиентом по имени «Саблев». Для упрощения этого примера предположим, что клиент арендует некоторый объект только один раз.

Арендатор	Код	KodR	R76		
	Фамилия	Name	Саблев		

Код помещения	Адрес	Дата начала	Дата окончания	Стоимость	Код владельца	Фамилия владельца
KodP	Address	RStart	Rfinish	Rent	KodO	OName
P04	Адрес 1	1.06.94	30.08.96	350	C40	Кротова
P16	Адрес 2	1.09.96	1.09.98	450	C93	Зимина

Существует два подхода исключения повторяющихся групп из ненормализованных таблиц.

В первом подходе повторяющиеся группы устраняются путем ввода соответствующих данных в пустые столбцы строк с повторяющимися данными. Полученная в результате этих действий таблица, которая теперь будет называться отношением, содержит атомарные (или единственные) значения на пересечении каждой строки и столбца.

Во втором подходе выбирается атрибут или группа атрибутов, которые образуют первичный ключ ненормализованной таблицы, а затем повторяющиеся группы помещаются в новое отношение вместе с копиями ключа исходной таблицы.

Пример 1НФ. Первая нормальная форма.

Данные об объектах, арендованных двумя клиентами, «Саблев» и «Рубин» преобразуются из формы в таблицу со строками и столбцами, формат которой представлен в табл. 5.2. Эта исходная таблица данных является примером ненормализованной таблицы (ННФ).

Таблица 5.2.

Отношение R

KodR	RName	KodP	Address	RStart	Rfinish	Rent	KodO	OName
R76	Саблев	P04	Адрес 1	1.06.94	30.08.96	350	C40	Кротова
		P16	Адрес 2	1.09.96	1.09.98	450	C93	Зимина
R56	Рубин	P04					C40	Кротова
		P36					C93	Зимина
		P16					C93	Зимина

Первичным ключом в отношении R является атрибут KodR.

В этой таблице имеется повторяющаяся группа атрибутов:

(KodP, Address, RStart, Rfinish, Rent, KodO, Oname)

Первый подход. Создадим таблицу, в которой вместо 2 строк будет 5: каждая повторяющаяся группа образует новый кортеж, недостающие значения дублируются. В результате получим новое отношение

Таблица 5.3.

Отношение R1

KodR	KodP	RName	Address	RStart	RFinish	Rent	KodO	OName
R76	P04	Саблев	Адрес 1	1.06.94	30.08.96	350	C40	Кротова
R76	P16	Саблев	Адрес 2	1.09.96	1.09.98	450	C93	Зимина
R56	P04	Рубин					C40	Кротова
R56	P36	Рубин					C93	Зимина
R56	P16	Рубин					C93	Зимина

В этом отношении все потенциальные ключи – составные:

(KodR, KodP) – если арендатор не арендует один и тот же объект дважды;

(KodR, Rstart) – если арендатор не арендует в один день несколько объектов;

(KodP, Rstart) – если объект не сдается в аренду в один день более одного раза.

Выберем в качестве первичного ключа - (KodR,KodP). Отношение R1 находится в 1НФ, но в нем имеется избыточность данных. Для устранения этого ее необходимо преобразовать в 2НФ.

Второй подход. Повторяющаяся группа удаляется и переносится в другое отношение с копией первичного ключа. Из отношения R получаем два отношения R4 и R5:

Отношение R4

KodR	RName
R76	Саблев
R56	Рубин

Отношение R5

KodR	KodP	Address	RStart	Rfinish	Rent	KodO	OName
R76	P04	Адрес 1	1.06.94	30.08.96	350	C40	Кротова
R76	P16	Адрес 2	1.09.96	1.09.98	450	C93	Зимина
R56	P04					C40	Кротова
R56	P36					C93	Зимина
R56	P16					C93	Зимина

Оба отношения находятся в первой нормальной форме, поскольку на пересечении каждой строки и каждого столбца находится единственное значение. Заметим, что последнее отношение по-прежнему обладает некоторой избыточностью данных, а потому может пострадать от аномалий обновления.

5.6. Вторая нормальная форма (2НФ)

Вторая нормальная форма (2НФ) основана на понятии полной функциональной зависимости.

Полная функциональная зависимость	В некотором отношении атрибут B называется **полностью** функционально зависимым от атрибута (или группы) A, если атрибут B функционально зависит от полного значения атрибута A и не зависит ни от какого подмножества атрибута A.
Частичная функциональная зависимость	Если в функциональной зависимости A→B в группе A есть некий атрибут, при удалении которого эта зависимость сохраняется, то она называется **частичной** функциональной зависимостью

Вторая нормальная форма применяется к отношениям с составными первичными ключами, т.е. к таким отношениям, первичный ключ которых

состоит из двух или больше атрибутов. Отношение с первичным ключом на основе единственного атрибута всегда находится в 2НФ.

Вторая нормальная форма (2НФ)	Отношение, которое находится в первой нормальной форме и каждый атрибут которого, не входящий в состав первичного ключа, характеризуется полной функциональной зависимостью от первичного ключа.

Нормализация 1НФ-отношений с образованием 2НФ-отношений состоит в устранении частичных зависимостей.

Правило приведения отношения к форме 2НФ. Если в отношении между атрибутами существует частичная зависимость, то функционально-зависимые атрибуты удаляются из него и помещаются в новое отношение вместе с копией их детерминанта.

В приведенной ниже таблице приведены все функциональные зависимости, имеющиеся в отношении R1 с первичным ключом (KodR, KodP).

f1	KodR, KodP→RStart, RFinish, …	Первичный ключ
f2	KodR→RName	Частичная зависимость
f3	KodP→ Address, Rent, KodO, OName	Частичная зависимость
f4	KodO→ OName	Транзитивная зависимость
f5	KodR, RStart → KodP, Address, …	Потенциальный ключ
f6	KodP, RStart → KodR, RName, …	Потенциальный ключ

Обнаружение частичных зависимостей в отношении R1 означает, что данное отношение не находится во второй нормальной форме. Для преобразования отношения его в 2НФ необходимо создать два новых отношения: частично зависимые атрибуты, не входящие в первичный ключ, перемещаются в них вместе с копией части первичного ключа, от которой они функционально зависят. Применение этого правила в нашем случае приведет к созданию трех новых отношений — C1, R1 и P1, которые представлены ниже.

C1

KodR	RName

R76	Саблев
R56	Рубин

R2

KodR	KodP	RStart	RFinish
R76	P04	1.06.94	30.08.96
R76	P16	1.09.96	1.09.98
R56	P04	…	…
R56	P36		
R56	P16		

P1

KodP	Address	Rent	KodO	OName
P04	Адрес 1	350	C40	Кротова
P16	Адрес 2	450	C93	Зимина
P36			C93	Зимина

Эти три отношения находятся во второй нормальной форме, так как каждый атрибут, не входящий в первичный ключ полностью функционально зависит от первичного ключа отношения.

6.7. Третья нормальная форма (3НФ)

Хотя 2НФ-отношения в меньшей степени обладают избыточностью данных, чем 1НФ-отношения, в них еще имеются аномалии обновления.

Транзитивная зависимость	Если для атрибутов A, B и C некоторого отношения существуют зависимости вида A→B и B→C, то говорят, что атрибут C транзитивно зависит от атрибута A через атрибут B (при условии, что атрибут A функционально не зависит ни от атрибута B, ни от атрибута C).
Третья нормальная	Отношение, которое находится в первой и второй

форма (3НФ)	нормальных формах и не имеет не входящих в первичный ключ атрибутов, которые находились бы в транзитивной функциональной зависимости от этого первичного ключа.

Приведение отношения к 3НФ состоит в устранении транзитивных функциональных зависимостей.

Правило приведения отношения к 3НФ. Если в отношении существует транзитивная зависимость, то транзитивно-зависимые атрибуты удаляются из него и помещаются в новое отношение вместе с копией их детерминанта.

Рассмотрим функциональные зависимости, существующие в отношениях C1, R2 и P1

Отношение C1	
f2	KodR→Rname
Отношение R2	
f1	KodR, KodP→Rstart, Rfinish
f5	KodR, RStart→KodP, RFinish
f6	KodP, Rstart→KodR, RFinish
Отношение P1	
f3	KodP→ Address, Rent, KodO, OName
f4	KodO→OName

Все, не входящие в первичный ключ, атрибуты отношений C1 и R1, функционально зависимы только от их первичных или потенциальных ключей. Следовательно, отношения C1 и R1 не имеют транзитивных зависимостей, а потому они уже находятся в третьей нормальной форме.

В отношении P1 имеется транзитивная зависимость:

KodP→ KodO, KodO → OName

Для преобразования P1 в 3НФ необходимо удалить транзитивную зависимость путем создания двух новых отношений P2 и O:

P2

KodP	Address	Rent	KodO
P04	Адрес 1	350	C40
P16	Адрес 2	450	C93
P36			C93

O

KodO	OName
C40	Кротова
C93	Зимина
C93	Зимина

Отношения P1 и O находятся в третьей нормальной форме, поскольку в них нет никаких транзитивных зависимостей от первичного ключа.

В результате выполнения нормализации исходное отношение R1, находящееся в 1НФ, было преобразовано в четыре отдельных отношения, каждое из которых находится в третьей нормальной форме:

C1

KodR	RName
R76	Саблев
R56	Рубин

R2

KodR	KodP	RStart	RFinish
R76	P04	1.06.94	30.08.96
R76	P16	1.09.96	1.09.98
R56	P04		
R56	P36		
R56	P16		

P2

KodP	Address	Rent	KodO
P04	Адрес 1	350	C40
P16	Адрес 2	450	C93
P36			C93

O

KodO	OName
C40	Кротова
C93	Зимина
C93	Зимина

5.8. Нормальная форма Бойса-Кодда (НФБК)

Нормальная форма Бойса-Кодда (НФБК) учитывает функциональные зависимости, в которых участвуют все потенциальные ключи отношения, а не только его первичный ключ. Для отношения с единственным потенциальным ключом его ЗНФ и НФБК являются эквивалентными.

Нормальная форма Бойса Кодда	Отношение находится в НФБК тогда и только тогда, когда каждый его детерминант является потенциальным ключом.

Для проверки принадлежности отношения к НФБК необходимо найти все функциональные зависимости, их детерминанты и убедиться в том, что они являются потенциальными ключами.

Нормальная форма Бойса-Кодда является более строгой версией формы 3НФ: каждое НФБК-отношение является 3НФ-отношением, но не всякое 3НФ-отношение является НФБК-отношением.

Перед тем как обратиться к очередному примеру, еще раз рассмотрим отношения C1, R1, P2 и O. Отношения C1, P2 и O являются НФБК-отношениями, так как каждое из них имеет только один детерминант, который в то же время является потенциальным ключом этого отношения. Отношение R1 содержит три детерминанта (KodR, KodP), (KodR, Rstart) и (KodP, Rstart), которые были выявлены нами в примере выше и имеют вид:

f1	KodR, KodP→Rstart, Rfinish
f5	KodR, RStart→KodP, RFinish
f6	KodP, Rstart→KodR, RFinish

Поскольку эти три детерминанта отношения являются также потенциальными ключами, то отношение находится в НФБК.

Нарушения требований НФБК происходят крайне редко – это может случиться только тогда, когда:

- имеются два (или более) составных потенциальных ключа;
- эти потенциальные ключи перекрываются, т.е. ими совместно используется, по крайней мере, один общий атрибут.

В следующем примере описывается ситуация, когда отношение нарушает требования НФБК, и представлен метод преобразования этого отношения в НФБК.

В этом примере мы рассмотрим отношение CS, которое содержит сведения о собеседованиях клиентов с сотрудниками компании «Аренда». Для проведения собеседования в тот день, на который назначена встреча с клиентом, в распоряжение сотрудника предоставляется особая комната. Все встречи в этот день он проводит только в ней, однако, в течение рабочего дня эта комната может использоваться несколькими разными сотрудниками. С клиентом проводится только одно собеседование в день, но он может участвовать в нескольких собеседованиях в разные дни.

Информация содержится в отношении CS:

CN	ID	IT	SN	RN
Код клиента	Дата	Время	Код сотрудника	Код комнаты
C76	13.05.98	10:30	S05	G101
C56	13.05.98	12:00	S05	G101
C74	13.05.98	12:00	S37	G102
C56	1.06.98	10:00	S05	G102

Отношение имеет три потенциальных ключа:

(CN, ID);

(SN, ID, IT);

(RN, ID, IT).

Таким образом, отношение CS обладает тремя составными потенциальными ключами, которые перекрываются, т.е. в них используется

один общий атрибут ID. Выберем в качестве первичного ключа комбинацию атрибутов (CN, ID).

В этом отношении имеются следующие функциональные зависимости:

fd1	CN, ID→IT, SN,RN
fd2	SN, ID, IT→CN, RN
fd3	RN, ID, IT→ SN, CN
fd4	SN, ID→ RN

Поскольку детерминанты функциональных зависимостей fd1, fd2 и fd3 являются потенциальными ключами этого отношения, то они не нарушают требований НФБК. Нам потребуется рассмотреть только функциональную зависимость fd4. Поскольку в этом отношении нет никаких частичных или транзитивных зависимостей от первичного ключа (CN, ID) и допускается наличие функциональной зависимости fd4, отношение CS находится в ЗНФ.

Однако это отношение не находится в НФБК, так как в нем присутствует детерминант (SN, ID), который не является потенциальным ключом этого отношения. Отношение CS может страдать от аномалий обновления. Например, 13 мая 1998 года при изменении номера комнаты, выделенной сотруднику SG5, потребуется обновить значения в двух строках.

Правило преобразования отношения в НФБК. Для преобразования отношения CS в форму НФБК необходимо устранить нарушающую ее ограничения функциональную зависимость посредством создания двух новых отношений C и S. В новое отношение S необходимо вынести функционально зависящие атрибуты с копией детерминанта.

C

CN	ID	IT	SN
C76	13.05.98	10:30	S05
C56	13.05.98	12:00	S05
C74	13.05.98	12:00	S37
C56	1.06.98	10:00	S05

S

SN	ID	RN
S05	13.05.98	G101
S05	13.05.98	G101
S37	13.05.98	G102
S05	1.06.98	G102

Процесс нормализации отношений может быть приведен в виде схемы (рис. 5.1).

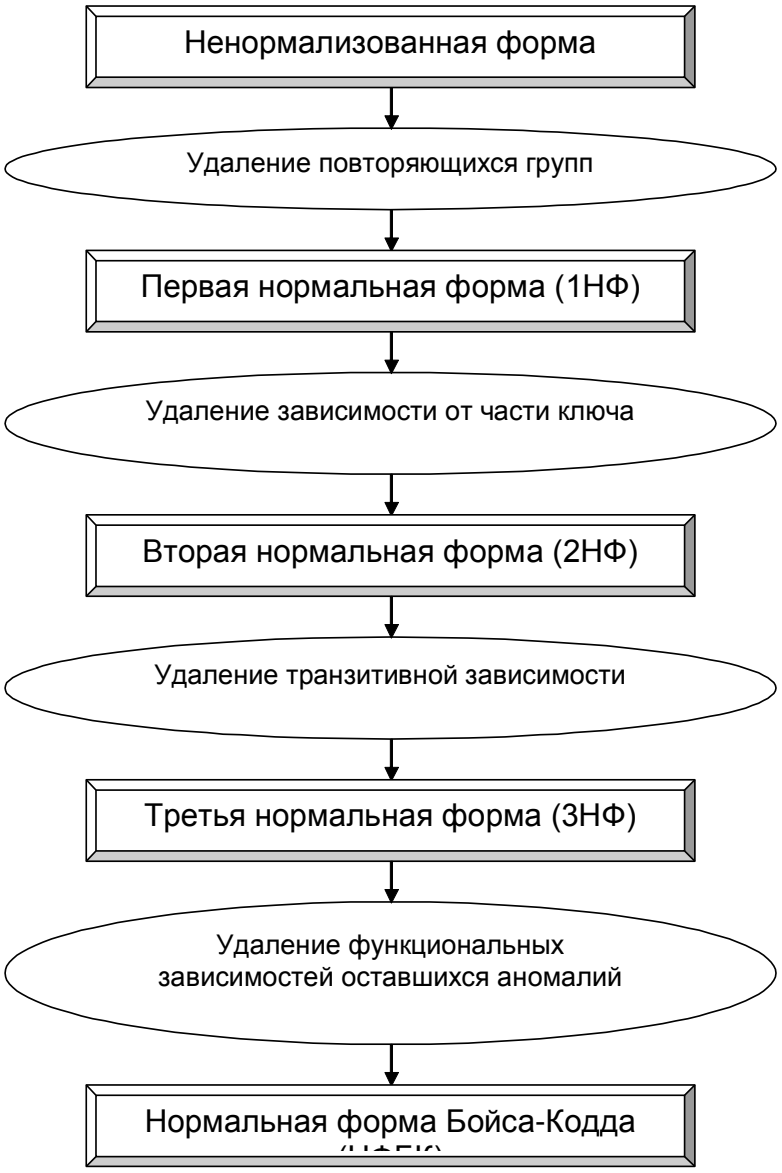


Рис. 5.1. Схема процесса нормализации

Задания.

Задание 5.1. Накладная

Разработать модель базы данных для автоматизации учета продажи товаров, нормализованную до формы Бойса-Кодда.

Учет товаров, отпускаемых со склада, осуществляется на основании накладной:

Накладная №	157
Дата	10/01/03
Покупатель	
Код	345
Наименование	«Сфера»
ИНН	072 001 000 6789
Другие реквизиты	…

№	Товар		Количество	Ед. изм.		Цена	Стоимость
	наименование	код		код	наим.		
1	Сахар	0012	100	12	кг	15.00	1 500
2	Макароны	0015	200	23	пачка	13.00	2 600
	Всего		600				4 100

Условия задачи:

1. При разработке базы данных считать, что номер накладной – уникален.

2. Накладная выписывается на конкретного клиента.

3. В накладной строки нумеруются последовательно 1, 2, …. Каждая накладная содержит, по крайней мере, одну строку товаров.

4. Цена отпускаемого товара в данный момент времени, и единица измерения определяется кодом товара.

5. Цена товара в некоторый момент может измениться, но в накладных, оформленных ранее, она должна остаться прежней.

Решение

Сведем данные накладных в одно отношение - плоскую таблицу. Присвоим отношению и атрибутам наименования.

Атрибут «Стоимость» в это отношение включать не будем, так как его значение может быть вычислено. В отношение не будем включать строку «Всего», так как в ней тоже находятся вычисляемые значения.

Для сокращения таблицы в нее не будем включать атрибуты «Другие реквизиты». Они относятся, как и атрибут «ИНН» к клиенту и будут находиться в той же таблице, где и реквизит «ИНН».

Так как цена товара может меняться, нам необходимо хранить в таблице значения цены продажи товара и текущей цены товара.

В результате получим отношение PR

№ накл.	Дата	Код покуп.	Наименование	ИНН	№ строки	Код товара	Товар	Кол-во	Код ЕИ	ЕИ	Цена продажи	Цена текущая
N	D	KodP	NP	INN	NS	KodT	T	Kol	KodE	EI	P	PT
157	10/01/03	345	«Сфера»	…	1 2 3	0012 0015 …	Сахар Макароны	100 200	12 23	кг пачка	15.00 13.00	16.00 13.00
158	11/01/03	444	…		1	0012	Сахар		12	кг	16.00	16.00

Первичный ключ отношения – атрибут N – номер накладной.

Отношение не нормализовано, так как в нем имеется повторяющаяся группа атрибутов (NS, KodT, T, Kol, KodE, EI, P, PT).

1НФ.

Приведем его к форме 1НФ путем преобразования каждой повторяющейся группы в кортеж и дублированием недостающих данных. В результате получим новое отношение PR1:

N	D	KodP	NP	INN	NS	KodT	T	Kol	KodE	EI	P	PT
157	10/01/03	345	«Сфера»	…	1	0012	Сахар	100	12	кг	15.00	16.00
157	10/01/03	345	«Сфера»	…	2	0015	Макароны	200	23	пачка	13.00	13.00
158	…				1							

Отношение PR1 находится в первой нормальной форме 1НФ, так как значения всех атрибутов атомарны и нет повторяющихся групп.

Потенциальный ключ – совокупность атрибутов: (N, NS), так как никакие другие атрибуты не могут однозначно идентифицировать кортежи.

Может ли совокупность атрибутов (N, KodT) быть потенциальным ключом? Да, если в одной накладной никогда не встречается один и тот же товар более одного раза.

Выберем в качестве первичного ключа совокупность атрибутов (N, NS). Отношение может быть записано в виде (атрибуты первичного ключа подчеркнуты):

PR1(<u>N</u>, D, KodP, NP, INN, <u>NS</u>, KodT ,T, Kol, KodE, EI, P, PT).

2НФ

В отношении PR1 имеются функциональные зависимости:

f1	N, NS → D, KodP, NP, INN, KodT, T, Kol, KodE, EI, P, PT	от ПК
f2	N → D, KodP, NP, INN,	частичная
f3	KodP → NP, INN	транзитивная
f4	KodT → T, KodE, EI, PT	транзитивная
f5	KodE → EI,	транзитивная

В отношении PR1 имеется частичная ФЗ f2.

Выполним декомпозицию отношения: создадим новое отношение и вынесем в него функционально зависимые от атрибута N атрибуты (D, KodP, NP, INN) вместе с **копией** части первичного ключа (N).

Получим два отношения:

NAK(N, D, KodP, NP, INN).

PR2(N, NS, KodT ,T, Kol, KodE, EI, P, PT).

Эти отношения находятся во второй нормальной форме 2НФ.

3НФ

В отношении NAK имеется транзитивная зависимость:

N → KodP; KodP → NP, INN

Вынесем транзитивно зависимые атрибуты (NP, INN) вместе с копией детерминанта (KodP) в новое отношение.

Получим из отношения NAK:

NAK1(N, D, KodP).

POK(KodP, NP, INN).

В отношении PR2:

PR2(N, NS, KodT ,T, Kol, KodE, EI, P, PT)

имеется транзитивная зависимость:

N, NS, → KodT , KodT →T, KodE, EI, PT

Вынесем в новое отношение транзитивно зависимые атрибуты T, KodE, EI, PT вместе с копией детерминанта KodT. Получим новое отношение:

TOV(KodT ,T, KodE, EI, PT)

Отношение PR2 преобразуется в отношение:

PR3(N, NS, KodT, Kol, P)

В PR3 имеются ФЗ только от первичного ключа.

В TOV имеется транзитивная зависимость:

KodT $\rightarrow$ KodE; KodE $\rightarrow$ EI

Вынесем в новое отношение

ED(KodE, EI)

останется

TOV2(KodT ,T, KodE, PT)

В результате получим набор отношений:

Отношение	Название
NAK1(N, D, KodP)	Накладные
POK(KodP, NP, INN)	Покупатели
PR3(N, NS, KodT, Kol, P)	Продажи
TOV2(KodT ,T, KodE, PT)	Товары
ED(KodE, EI)	Ед. измерения

В этих отношениях имеются только функциональные зависимости от первичных ключей.

НФ Бойса-Кодда

Во всех полученных отношениях имеется только один потенциальный ключ, следовательно, они находятся в нормальной форме Бойса-Кодда.

Задание 5.2. Учет работы автотранспорта

Разработать модель базы данных для автоматизации учета работы автотранспорта, осуществляющего доставку товаров в магазины, нормализованную до формы Бойса-Кодда.

Учет работы автотранспорта, осуществляется на основании путевого листа.

Путевой лист №		157		Цена бензина	17,00
Дата		10/01/05			
Водитель:				Автомобиль:	
Код		345		Код	333
Наименование		Фамилия И.О.		Гос. Номер	Т345АИ
Категория		С		Расход бензина на 1 км (л)	1

№	Объект		Расстояние (км)	Расход бензина (л)	Стоимость горючего
	наименование	код			
1	Магазин № 2	0012	14		
2	Магазин № 5	0015	16		
3	Магазин № 7	0022	12		
	Всего				
Диспетчер				Иванов	

При разработке базы данных считать, что номер путевого листа – уникален, для его формирования может быть использован счетчик.

Путевой лист выписывается на одного водителя.

В путевом листе строки нумеруются последовательно 1, 2, …. Каждый путевой лист содержит, по крайней мере, одну строку.

Расстояние до объектов не меняется и определяется кодом объекта.

Водитель может работать на разных автомобилях.

Расход горючего на 1 км пути определяется кодом автомобиля.

Цена 1 л бензина – 17 руб. Она может в некоторый момент измениться, но в путевых листах, оформленных ранее она не должна меняется.

Задание 5.3. Учет работы сотрудников

Разработать модель базы данных для автоматизации учета работы сотрудников организации и начисления заработной платы (ЗП), нормализованную до формы Бойса-Кодда.

Учет работы сотрудников и начисление ЗП, осуществляется на основании ведомости:

Ведомость №	157		Дата	3/02/05
За период	Год		Месяц	Норматив (час.)
	2005		01	180
Бригада				
Код	345			
Наименование	Конструкторский			

№	Сотрудник		Должность		Отработано часов	Начислено ЗП
	Фамилия И.О.	Таб. номер	код	название		
1	Иванов И.И.	0012	1	мастер	180	
2	Петров П.П.	0015	2	слесарь	160	
3	Сидорова С.С.	0022	2	слесарь	165	
	Всего					

При разработке модели базы данных считать, что номер ведомости – уникален, для его формирования может быть использован счетчик.

Ведомость оформляется один раз в месяц на каждую бригаду. В ведомости строки нумеруются последовательно 1, 2, …. Каждая ведомость содержит, по крайней мере, одну строку.

Состав бригады может меняться, он формируется из полного списка сотрудников.

Оклад определяется должностью (кодом).

Заработная плата вычисляется на основании оклада пропорционально отработанным часам.

Норматив рабочего времени устанавливается для каждого периода времени (год, месяц).

Задание 5.4. Учет нагрузки преподавателей

Разработать модель базы данных для автоматизации учета учебной нагрузки преподавателей вуза, нормализованную до формы Бойса-Кодда.

Учет нагрузки преподавателей, осуществляется на основании карточки:

Код преподавателя		157		Код кафедры	21
Фамилия И.О.		Иванов И.И.		Кафедра	ИС
Код должности		3			
Должность		доцент			
За период (уч. год)		2006/2007			

№	Дисциплина		Часов			Всего
	Код	Наименование	лекций	практ.	прочее	
1	111	Информатика	36	36	40	112
2	321	Базы данных				
3	121	Основы ВМ				
	Всего					

При разработке базы данных считать, что код карточки – уникален, для его формирования может быть использован счетчик.

Карточка оформляется для преподавателя каждый учебный год.

В карточке строки нумеруются последовательно 1, 2, …. Каждая карточка содержит, по крайней мере, одну строку.

Количество часов определяется кодом дисциплины и может различаться в разные годы.

Задание 5.5.

Разработать модель базы данных «Сессия» для автоматизации учета сдачи экзаменов студентами, нормализованную до формы Бойса-Кодда. В процессе нормализации для всех отношений определить потенциальные и первичные ключи, все функциональные зависимости. Дополнительные атрибуты не использовать.

Информация о сдаче экзаменов представляется в виде (выписка из зачетной книжки):

Студенты		Факультет			Дисциплина			
№ зач. книжки	Фамилия И.О.	Код	Наименование	Группа	Код	Наименование	Оценка	Дата
NZ	**F**	**KF**	**NF**	**G**	**KD**	**DIS**	**P**	**D**
23456	Иванов И.И.	300	ФМиКН	391	21	Математика	4	21.01.99
					22	Информатика	4	12.01.99
					23	Программирование	4	17.01.99
					24	Экономика	5	7.03.99
12345	Петров П.П.	300	ФМиКН	392	21	Математика	4	21.01.99
					22	Информатика	4	22.01.99
					23	Программирование	3	24.01.99
34566	Иванов И.И.	400	Экономический	491	41	Информатика	3	11.01.99
					42	Математика	3	23.01.99

Для данных справедливы следующие правила:

1. Студент имеет личные данные (Фамилию ИО) и зачетную книжку, которой присваивается уникальный номер.
2. Студент обучается в группе, которая «находится» на факультете.

3. Студент сдает экзамены по дисциплинам, результат (оценка) и дата сдачи отмечается записью в зачетной книжке.
4. Перечень дисциплин, по которым сдаются экзамены, определяется для группы.
5. Студент не может иметь несколько оценок по дисциплине: (при пересдаче оценка заменяется);
6. Студент может не сдавать некоторые экзамены (например, по болезни).
7. Студент может сдавать не более одного экзамена в день.

Вариант задания. В базе данных необходимо хранить все результаты экзаменов, в том числе пересдачи.

Задание 5.6.

Разработать модель базы данных «Расписание занятий», нормализованную до формы Бойса-Кодда. В процессе нормализации для всех отношений определить потенциальные и первичные ключи, все функциональные зависимости. Дополнительные атрибуты не использовать.

После приведения отношения к 1НФ выбрать в качестве первичного ключа потенциальный ключ, который содержит атрибут KP.

Для данных справедливы следующие правила:

1. Во время проведения занятия в аудитории проводит занятие один преподаватель, занимается одна группа (группы не объединяются);
2. Преподаватель может проводить занятия по нескольким дисциплинам;
3. В группе по одной дисциплине проводит занятие один преподаватель;
4. По каждой дисциплине для группы проводится каждую неделю ровно одно занятие: 1 или 2 часа;

5. Наименование дисциплины и количество часов по дисциплине определяется кодом дисциплины.
6. Расписание занятий приведено в следующей таблице

Преподаватель		Группа	Дисциплина		День недели	Номер пары	Кол-во часов	Аудитория
Код	Фамилия		Код	Название				
KP	**F**	**G**	**KD**	**D**	**DN**	**N**	**T**	**A**
21	Иванов	315	М21	Математика	1	3	2	315
		315	М23	Логика	2	4	1	320
22	Петров	315	И33	Информатика	2	3	2	320
		340	И33	Информатика	3	3	2	320
23	Сидорова	316	М21	Математика	4	1	2	106
…								

Задание 5.7.

Разработать модель базы данных «Нагрузка преподавателей» для автоматизации учета нагрузки преподавателей вуза, нормализованную до формы Бойса-Кодда. В процессе нормализации для всех отношений определить потенциальные и первичные ключи, все функциональные зависимости. Дополнительные атрибуты не использовать.

После приведения отношения к 1НФ выбрать в качестве первичного ключа потенциальный ключ, который содержит атрибут KP.

Данные о нагрузках преподавателей сведены в следующую таблицу:

Факультет		Кафедра		Преподаватель		Дисциплина		Кол-во часов
Код	Название	Код	Название	Код	Название	Код	Название	
KF	**NF**	**KK**	**NK**	**KP**	**NP**	**KD**	**ND**	**CH**
01	МиКН	31	ИС	P021	Иванов	D011	Информатика	140
						D012	БД	60
				P022	Петров	D011	Информатика	140
		32	ПО	P023	Сидоров	D013	ОВМ	140
				P024	Степанов	D033	Программирование	136
02	Исторический	41	Истории	P122	Иванов			
		42	ИС	P221	Иванов	D014		

Для данных справедливы следующие правила:

7. Преподаватель работает только на одной кафедре;
8. Преподаватель может проводить занятия по нескольким дисциплинам;
9. По одной дисциплине занятия могут проводить разные преподаватели;
10. Дисциплина закреплена за кафедрой;
11. Количество часов устанавливается для дисциплины;
12. Кафедра является подразделением факультета.

Задание 5.8

Разработать модель базы данных «Автотранспорт» для автоматизации учета работы автотранспорта предприятия, нормализованную до формы Бойса-Кодда. В процессе нормализации для всех отношений определить потенциальные и первичные ключи, все функциональные зависимости. Дополнительные атрибуты не использовать.

Информация о работе автотранспорта представляется в виде:

Дата	Автомобиль			Пробег за смену (км)	Расход горючего на 100 км (л)	Цена горючего	Расход горючего за смену (л)	Водитель		
	гос. №	модель	тип					код	Фамилия ИО	категория
DT	**N**	**M**	**T**	**P**	**R**	**C**	**RS**	**KV**	**F**	**K**
8.01.2008	М345ТР	Ваз2111	легковой	255	9	17	22,95	123	Иванов	D
	С346МА	ЗИЛ 130	грузовой	126	21	17	26,46	345	Петров	C
9.01.2008	Т444ТР	Ваз2111	легковой	555	9	18	49,95	345	Петров	C
	С346МА	ЗИЛ 130	грузовой	444	21	18	93,24	123	Иванов	D
	М222ММ	Ваз2110	легковой	100	9	18	9,00	333	Сидоров	C

Для данных справедливы следующие правила:

13. Гос. № автомобиля – уникальный;
14. В один день водитель может работать только на одном автомобиле;
15. В один день на автомобиле может работать только один водитель
16. Тип автомобиля определяется моделью;
17. Расход горючего для автомобилей одной модели одинаковый;

18.Цена горючего в течение дня не меняется;

19.Категория водителя не меняется.

Задание 5.9.

Разработать модель базы данных «Книги» для автоматизации учета издания книг в издательствах, нормализованную до формы Бойса-Кодда. В процессе нормализации для всех отношений определить потенциальные и первичные ключи, все функциональные зависимости. Дополнительные атрибуты не использовать.

Код книги	Название	Авторы		Издательство		Год	Тираж	Объем (стр.)
		Код	Фамилия	Код	Издательство			
201	Сказки	21 22	Иванов Петров	1	Издательство 1	1997	500	200
302	Поэма	33	Сидоров	2	Издательство 2	1996	1000	300
201	Сказки	21 22	Иванов Петров	2	Издательство 2	1995	700	200
401	Повесть	21	Иванов	1	Издательство 1	1997	1000	50

Для данных справедливы следующие правила:

1. Книга пишется авторами. Один автор может написать несколько книг.
2. Книга издается издательством.

3. Книга может быть издана в некотором году только одним издательством.
4. Книга в издательстве печатается только один раз.
5. При переиздании объем книги не изменяется, тираж может быть различным.

Задание 5.10.

Разработать модель базы данных «Путевки» для автоматизации учета путевок туристической фирмы, нормализованную до формы Бойса-Кодда. В процессе нормализации для всех отношений определить потенциальные и первичные ключи, все функциональные зависимости. Дополнительные атрибуты не использовать.

На каждого туриста заведена карточка, в которой ведется учет в следующем виде:

Клиент	код	**KL**	A23
	фамилия	**FIO**	Иванов
	телефон	**Tel**	33-33-33

№ п/п	Турпоездка			Срок поездки		Туроператор	
	код	страна	тип	начало	окончание	код	фирма
NP	**KP**	**S**	**T**	**DN**	**DE**	**KO**	**F**
1	P24	Египет	отдых	1.12.1999	11.09.2000	V23	Visa
2	P33	Турция	учеба	15.09.2000	17.12.2001	V33	Temp

Для данных справедливы следующие правила:

1. Клиент не может оформить несколько турпоездок на одно и то же время;
2. Поездки туриста в карточке нумеруются последовательно 1, 2, и т.д.
3. Код турпоездки определяет страну, тип и туроператора (но не даты).

4. На турпоездку с одним кодом может быть оформлено несколько клиентов;
5. Туроператор обеспечивает несколько турпоездок.

Задание 5.11.

Разработать модель базы данных «ТО» (техническое обслуживание автомобилей). Привести ее к НФБК.

Для всех отношений определить потенциальные ключи и первичный ключ, выписать все функциональные зависимости. Дополнительные атрибуты не использовать.

Учет выполнения работ по ТО осуществляется на основании документа, который называется «Заказ-наряд» (ЗН):

ЗАКАЗ-НАРЯД №			**NZ**	12			
	Дата		**DT**	21/01/2008			
Клиент	Код		**KodKl**	23			
	Фамилия ИО		**FioKl**	Петров			
Транспортное средство	Гос. №		**NTS**	Х234АС			
	Модель	Код	**KodMod**	111			
		Модель	**ModTS**	ВАЗ2111			
Выполненные работы							
№ п/п	Работа		Кол-во	Цена	Сумма	Мастер	
	код	название				код	Фамилия ИО
NS	**KodR**	**R**	**Kol**	**P**	**S**	**KodM**	**FioM**
1	233	замена ТК	2	300	600	21	Иванов
2	123	ремонт замка					

Для данных справедливы следующие правила:

6. На каждое транспортное средство (ТС) оформляется отдельный ЗН;
7. Номер ЗН уникальный;
8. Все работы по ЗН выполняются в один день;
9. Выполненные работы в каждом ЗН нумеруются 1,2,…, пустых ЗН не бывает;
10. В один день ТС обслуживается только по одному ЗН;
11. Клиент может иметь несколько транспортных средств, у транспортного средства владелец не меняется;
12. Расценки на работы определяются кодом работы, они могут меняться (во времени), в выполненных заказах они должны оставаться прежними;
13. Один вид работ могут выполнять различные мастера;
14. Мастер может выполнять несколько видов работ.

Тема 6. ЯЗЫК SQL.
ПОДЪЯЗЫК МАНИПУЛИРОВАНИЯ ДАННЫМИ DML

Язык **SQL** (Structured Query Language) в настоящее время является стандартом в реляционных СУБД.

6.1. Введение в язык SQL

Язык SQL позволяет:

- создавать базы данных и таблицы с полным описанием их структуры;
- выполнять основные операции манипулирования данными, такие как вставка, модификация и удаления данных из таблиц;
- выполнять простые и сложные запросы, осуществляющие преобразование необработанных данных в необходимую информацию.

Язык SQL имеет два основных компонента:

- язык DDL (Data Definition Language), предназначенный для определения структур базы данных;
- язык DML (Data Manipulation Language), предназначенный для выборки и обновления данных.

Язык SQL включает только команды определения и манипулирования данными — в нем отсутствуют какие-либо команды управления ходом вычислений.

Язык SQL может использоваться двумя способами. Первый предусматривает *интерактивную* работу, заключающуюся во вводе пользователем с терминала отдельных SQL- операторов. Второй состоит во *внедрении* SQL-операторов в программы, разработанные на процедурных языках.

В настоящее время для языка SQL существует международный стандарт (ISO, 1992 г.), формально превращающий этот язык в стандартный язык определения и манипулирования реляционными базами данных — каковым он фактически и является.

Стандарт ISO SQL не поддерживает таких формальных терминов, как «отношение», «атрибут» и «кортеж», вместо них применяются термины «таблица», «столбец» и «строка». Кроме того, следует отметить, что стандарт SQL не придерживается жестко определений реляционной модели данных. Например, в языке SQL допускается, что созданная в результате выполнения операции таблица может содержать дублирующиеся строки, устанавливается упорядоченность столбцов, а пользователю разрешается упорядочивать строки в таблице.

6.2. Запись SQL-операторов

В этом разделе приводится краткое описание структуры операторов и нотации, которую используются для определения формата конструкций языка SQL. SQL-оператор состоит из зарезервированных слов, а также из слов, определяемых пользователем. Зарезервированные слова являются постоянной частью языка SQL и имеют фиксированное значение. Их следует записывать в *точности* так, как это установлено, и нельзя разбивать на части для переноса из одной строки в другую.

Слова, определяемые пользователем, задаются самим пользователем (в соответствии с определенными синтаксическими правилами) и представляют собой имена различных объектов базы данных — таблиц, столбцов, представлений, индексов и т.д.

Фрагмент оператора, состоящий из зарезервированного слова и (возможно) слов пользователя, называют фразой, предложением или контейнером.

Слова в операторе размещаются в соответствии с установленными синтаксическими правилами. Хотя в стандарте это не указано, многие диалекты языка SQL требуют задания в конце оператора некоторого символа, обозначающего окончание его текста (как правило, с этой целью используется символ точка с запятой (;)).

Большинство компонентов SQL-операторов не чувствительно к регистру.

Для определения формата SQL-операторов будет применяться следующая нотация:

- прописные буквы будут использоваться для записи зарезервированных слов и должны указываться в операторах точно так же, как это будет показано;
- строчные буквы будут использоваться для записи слов, определяемых пользователем;
- вертикальная черта (|) указывает на необходимость выбора одного из нескольких приведенных значений;
- фигурные скобки определяют обязательный элемент — например, {a};
- квадратные скобки определяют необязательный элемент — например, [a].
- многоточие (...) используется для указания необязательной возможности повторения конструкции.

Литералы представляют собой константы, которые используются в SQL-операторах. Все *нецифровые* значения данных всегда должны заключаться в апострофы.

6.3. Операторы манипулирование данными

В этом разделе обсуждаются следующие операторы языка SQL DML:

- SELECT — выборка данных из базы;
- INSERT — вставка данных в таблицу;
- UPDATE — обновление (изменение) данных в таблице;
- DELETE — удаление данных из таблицы.

Для построения примеров SQL-операторов будем использовать пример учебной базы данных «АГЕНСТВО», приведенной в приложении:

B(KodB, City, Street, Pind, Tel)

S(KodS, Lname, Fname, Pol, DR, Dol, ZP, KodB)

P(KodP, City, Street, Type, Rooms, Rent, KodO, KodS, KodB)

R(KodR, Name, Address, Type, MaxRent, KodB)

O(KodO, Name, Address, Tel)

V(KodR, KodP, Date, Comment).

6.4. Оператор SELECT. Простые запросы

Назначение оператора SELECT состоит в выборке и отображении данных одной или более таблиц базы данных. Этот оператор реализует все операции реляционной алгебры.

Общий формат оператора SELECT имеет следующий вид:

SELECT [DISTINCT | ALL] {* | [*column_expression [AS new_name*]] [,...]}
FROM *table_name [alias tname] [,...]*
[WHERE *condition*]
[GROUP BY *column_list*] [HAVING *condition*]
[ORDER BY *column_list*]

Здесь параметр *column_expression* представляет собой имя столбца или выражение. Параметр *table_name* является именем существующих в базе данных таблицы или представления, к которым необходимо получить доступ. Необязательный параметр *alias* — это сокращение, устанавливаемое для имени таблицы.

Фразы оператора SELECT определяют:

SELECT	столбцы, которые должны присутствовать в выходных данных
FROM	имена используемой таблицы или нескольких таблиц
WHERE	фильтрацию строк объекта в соответствии с заданными условиями
GROUP BY	группировку строк, имеющих одно и то же значение в указанном столбце
HAVING	фильтрацию группы строк объекта в соответствии с указанным условием
ORDER BY	упорядоченность результатов выполнения оператора

Порядок фраз в операторе SELECT *не может* быть изменен. Только две фразы — SELECT и FROM — являются обязательными, все остальные могут

быть опущены. Операция SELECT является закрытой**:** результат представляет собой другую таблицу. Существует множество вариантов записи данного оператора, что иллюстрируется приведенными ниже примерами.

Выбор всех строк

Пример 6.1. Получить детальные сведения о каждом из сотрудников.

Поскольку не указаны никакие ограничения, помещать в оператор фразу WHERE не требуется. Кроме того, необходимо выбрать все существующие в таблице столбцы. Поэтому данный запрос записывается следующим образом:

```
SELECT Kods, Lname, Fname, Pol, DR, Dol, ZP, Kodb
    FROM S;
```

Поскольку выборка всех имеющихся в таблице столбцов выполнятся достаточно часто, в языке SQL определен упрощенный вариант записи значения «все столбцы» — вместо имен столбцов указывается символ *. Приведенный ниже оператор полностью эквивалентен первому и представляет собой упрощенный вариант записи того же самого запроса:

```
SELECT * FROM S
```

Результаты выполнения этих запросов – таблица, являющаяся копией таблицы S.

Пример 6.2. Получить сведения об индивидуальном номере, фамилии, имени и заработной плате сотрудников.

```
SELECT Kods, Lname, Fname, Dol, ZP FROM S;
```

Результат:

KodS	Lname	Fname	Dol	Zp
S21	Иванов	Иван	Руководитель	500
S37	Петрова	Татьяна	Агент	250
S14	Сидоров	Стеман	Менеджер	400
S09	Зотова	Ирина	Агент	240
S05	Степанова	Елена	Секретарь	200
S41	Петров	Игорь	Менеджер	450

Пример 6.3. Получить сведения (код) об осмотренных клиентами объектах.

SELECT *Kodp* FROM *V;*

Результатом выполнения этого оператора является таблица:

Результат:

KodP
P14
P04
P04
P14
P36

Если из результирующей таблицы необходимо исключить дублирующие строки, необходимо указать ключевое слово DISTINCT:

SELECT DISTINCT *Kodp* FROM *V;*

Результат:

KodP
P14
P04
P36

Для переопределения имени столбца или для его определения в случае использования в фразе SELECT выражения используется ключевое слово AS.

Пример 6.4. Получить сведения о годовой заработной плате сотрудников (включить индивидуальный номер, фамилию, имя, суммарная заработная плата).

SELECT *Kods, Lname, Fname, ZP*12* AS *SZP* FROM *S*;

Результат:

KodS	Lname	Fname	SZP
S21	Иванов	Иван	6000
S37	Петрова	Татьяна	3000
S14	Сидоров	Стеман	4800
S09	Зотова	Ирина	
S05	Степанова	Елена	
S41	Петров	Игорь	

Выбор строк (фраза WHERE)

В приведенных выше примерах в результате выполнения операторов SELECT выбирались все строки указанной таблицы. Однако очень часто требуется тем или иным образом ограничить набор строк, помещаемых в результирующую таблицу. Это достигается с помощью указания в запросе фразы WHERE. Оно состоит из ключевого слова WHERE, за которым следует перечень условий отбора записей

Существует пять основных типов условий поиска (или предикатов).

Сравнение.

Сравниваются результаты вычисления одного выражения с результатами вычисления другого выражения.

Пример 9.5. Сравнение. Получить сведения о сотрудниках, заработная плата которых превышает 300.

```
SELECT Kods, Lname, Fname, Dol, ZP
    FROM S
    WHERE ZP > 300;
```

В этом запросе используется таблица S и предикат ZP > 300.

Результат:

KodS	Lname	Fname	Dol	ZP
S21	Иванов	Иван	Руководитель	500
S14	Сидоров	Стеман	Менеджер	400
S41	Петров	Игорь	Менеджер	450

В языке SQL можно использовать следующие операторы сравнения:

=	равенство
<	меньше
>	больше
<=	меньше или равно
>=	больше или равно
<>	не равно (стандарт ISO)
!=	не равно (используется в некоторых диалектах)

Более сложные предикаты могут быть построены с помощью логических операторов AND, OR или NOT, а также с помощью скобок, используемых для определения порядка вычисления выражения для устранения любой возможной неоднозначности. Вычисление выражений в условиях выполняется по следующим правилам:

- выражение вычисляется слева направо;
- первыми вычисляются подвыражения в скобках;
- операторы NOT выполняются до выполнения операторов AND и OR;
- операторы AND выполняются до выполнения операторов OR.

Пример 6.6. Сложные условия поиска. Получить список всех отделений в городах Тюмень и Нижневартовск.

```
SELECT KodB, City, Street, Pind FROM B
    WHERE City = 'Тюмень' OR City = 'Нижневартовск' ;
```

Результат:

KodB	City	Street	Pind
B5	Тюмень	Береговая, 22	625002
B7	Нижневартовск	Новая, 15	625234
B3	Тюмень	Северная, 34	625026

Диапазон (BETWEEN / NOT BETWEEN).

Проверяется, попадает ли результат вычисления выражения в заданный диапазон значений.

Пример 6.7. Использование диапазонов. Получить список сотрудников с заработной платой от 200 до 300.

```
SELECT Kods, Lname, Fname, Dol, ZP
    FROM S
    WHERE ZP BETWEEN 200 AND 300;
```

Результат:

KodS	Lname	Fname	Dol	ZP
S37	Петрова	Татьяна	Агент	250
S09	Зотова	Ирина	Агент	240
S05	Степанова	Елена	Секретарь	200

Наличие ключевого слова BETWEEN требует задания границ диапазона значений. В данном случае результаты проверки будут положительными для всех работников компании с годовой заработной платой от **200** до **300.** Имеется и негативная версия проверки диапазона значений (NOT BETWEEN).

Приведенный выше запрос можно переписать следующим образом:

```
SELECT Kods, Lname, Fname, Dol, ZP  FROM S
        WHERE ZP <= 200 AND ZP >= 300;
```

Принадлежность и множеству (IN / NOT IN).

Проверяется, *принадлежит* ли результат вычисления выражения к заданному множеству значений.

Пример 6.8. Условия поиска с проверкой вхождения в множество. Получит список всех руководителей и менеджеров.

```
SELECT Kods, Lname, Fname, Dol  FROM S
        WHERE Dol IN ( 'Руководитель' , 'Менеджер' );
```

Проверка вхождения результата вычисления выражения в заданное множество организуется с помощью ключевого слова IN. При этом проверяется, соответствует ли результат вычисления выражения одному из значений в предоставленном списке. Существует и отрицательная версия этой проверки (NOT IN).

Соответствие шаблону (LIKE/NOT LIKE).

Проверяется, отвечает ли некоторое строковое значение заданному шаблону. В стандарте языка SQL существует два специальных символа шаблона, используемых при проверке символьных значений.

%	любая последовательность из нуля или более символов
_	символ подчеркивания представляет любой одиночный символ

Примеры использования знаков шаблона для столбца Addr (адрес):

Add LIKE *'T%'*	шаблон означает, что первый символ значения

	должен быть символом Т, а все остальные символы не проверяются;
Addr LIKE *'Н_ _ _'*	шаблон означает, что значение должно иметь длину, равную четырем символам, причем первым символом обязательно должен быть символ 'Н',
Addr LIKE *'%а'*	шаблон определяет любую последовательность символов длинной не менее одного символа, причем последним символом обязательно должен быть символ а;
Addr LIKE '%Тюмень%'	шаблон означает, что нас интересует любая последовательность символов, включающая подстроку Тюмень;
Addr NOT LIKE 'Н%'	шаблон указывает, что требуются любые строки, которые не начинаются с символа Н.

Пример 6.9. Условия поиска с указанием шаблона. Получить список всех арендаторов, проживающих в Тюмени.

```
SELECT KodR, Name, Address
    FROM R
    WHERE Address LIKE '%Тюмень%';
```

Значение NULL (IS NULL / IS NOT NULL).

Проверяется, содержит ли данный столбец определитель Null (неопределенное значение).

Пример 6.10. Использование значения NULL. Получить список осмотров помещений, по которым не сделано никаких заключений.

```
SELECT KodR, KodP, Comment FROM V
    WHERE Comment IS NULL
```

Результат

KodR	KodP	Comment
R56	P04	
R56	P36	

Использование условия Comment='' не эквивалентно Comment IS NULL!

6.5. Сортировка результатов (фраза ORDER BY)

В общем случае строки в результирующей таблице SQL-запроса не упорядочены. Однако их можно требуемым образом отсортировать, для чего в оператор SELECT помещается фраза ORDER BY. Фраза ORDER BY включает список разделенных запятыми идентификаторов столбцов, по которым требуется упорядочить результирующую таблицу запроса. Идентификатор столбца может представлять собой либо его имя, либо номер, который идентифицирует элемент списка. Номера столбцов могут использоваться в тех случаях, когда столбцы, по которым следует упорядочить результат, являются вычисляемыми, а фраза AS с указанием имени этого столбца в операторе SELECT отсутствует. Фраза ORDER BY позволяет упорядочить выбранные записи в порядке возрастания (ASC) или убывания (DESC) значений любого столбца или комбинации столбцов.

Фраза ORDER BY всегда должна быть последней в операторе SELECT.

Пример 6.11. Сортировка по значениям одного столбца. Составить отчет о зарплате всех работников компании, расположив строки в порядке убывания суммы зарплаты.

```
SELECT KodS, Fname, Lname, ZP
    FROM S
    ORDER BY ZP DESC;
```

Во фразе ORDER BY может быть указано и больше одного элемента. Первый элемент называется **главным ключом** сортировки, и он определяет общую упорядоченность строк результирующей таблицы. Если в результирующей таблице значения главного ключа сортировки не уникальны, может оказаться желательным упорядочить строки с одним и тем же значением главного ключа по какому-либо дополнительному ключу сортировки. Если во

фразе ORDER BY присутствуют второй и последующие элементы, то такие элементы называют **младшими ключами** сортировки.

Пример 6.12. Сортировка по нескольким столбцам. Подготовить список сдаваемых в аренду объектов, упорядоченный по их типу.

```
SELECT KodP, City, Type, Rooms, Rent
    FROM P
    ORDER BY Type
```

Результат:

KodP	City	Type	Rooms	Rent
P14	Тюмень	дом	6	650
P04	Тюмень	дом	4	350
P94	Нижневартовск	кварт.	4	400
P36	Ишим	кварт.	2	375
P21	Надым	кварт.	3	600
P16	Сургут	кварт.	2	450

В полученных результатах присутствуют сведения о четырех квартирах. Если не указан младший ключ сортировки, система расположит эти строки в произвольном порядке. Для того чтобы упорядочить их, например, в порядке убывания числа комнат, следует дополнительно указать младший ключ сортировки:

```
SELECT KodP, City, Type, Rooms, Rent
    FROM P ORDER BY Type, Rooms DESC
```

Результат:

KodP	City	Type	Rooms	Rent
P14	Тюмень	дом	6	650
P04	Тюмень	дом	4	350
P94	Нижневартовск	кварт.	4	400
P21	Надым	кварт.	3	600
P36	Ишим	кварт.	2	375
P16	Сургут	кварт.	2	450

6.6. Использование обобщающих функций языка SQL

Стандарт ISO содержит определение следующих пяти **обобщающих функций.**

COUNT	Возвращает количество значений в указанном столбце
SUM	Возвращает сумму значений в указанном столбце
AVG	Возвращает усредненное значение в указанном столбце
MIN	Возвращает минимальное значение в указанном столбце
MAX	Возвращает максимальное значение в указанном столбце

Все эти функции оперируют со значениями в столбце таблицы и возвращают единственное значение. Функции COUNT, MIN и MAX применимы как к числовым, так и к нечисловым полям, тогда как функции SUM **и** AVG могут использоваться только в случае числовых полей. За исключением COUNT(*), при вычислении результатов любых функции сначала исключаются все пустые значения, после чего требуемая операция применяется только к оставшимся значениям столбца.

Вариант COUNT(*) является особым случаем использования функции COUNT — его назначение состоит в подсчете всех строк в результирующей таблице, независимо того, содержатся там пустые, дублирующиеся или любые другие значения.

Если до применения обобщающей функции необходимо исключить дублирующиеся значения, следует перед именем столбца в определении функции поместить ключевое слово DISTINCT. Ключевое слово DISTINCT не имеет смысла для функций MIN и MAX. Однако его использование может оказывать эффект на результаты выполнения функций SUM и AVG. Ключевое слово DISTINCT в каждом запросе может быть указано не более одного раза.

Очень важно отметить, что обобщающие функции могут использоваться только в списке предложения SELECT и в составе предложения HAVING. Во всех других случаях использование этих функций недопустимо. Если список в предложении SELECT содержит обобщающие функции, а в тексте запроса

отсутствует фраза GROUP BY, обеспечивающая объединение данных в группы, то ни один из элементов списка предложения SELECT не может включать каких-либо ссылок на колонки, за исключением случая, когда эта колонка используется как аргумент обобщающей функции. Например, следующий запрос является некорректным:

```
SELECT KodS, COUNT(ZP) FROM S;
```

Ошибка состоит в том, что в данном запросе отсутствует фраза GROUP BY, а обращение к колонке KodS в предложении SELECT выполняется без применения обобщающей функции.

Пример 6.13. Использование функции COUNT (*). Определить, сколько сдаваемых в аренду объектов имеют ставку арендной платы менее 350.

```
SELECT COUNT(*) AS C  FROM P WHERE ZP > 350;
```

Ограничение на подсчет только тех сдаваемых в аренду объектов, арендная плата, которых составляет менее 350, реализуется посредством использования предложения WHERE. Количество сдаваемых в аренду объектов, отвечающих указанному условию, может быть определено с помощью обобщающей функции COUNT. Результаты выполнения запроса:

C
5

Пример 6.14. Использование функции COUNT (DISTINCT). Определить, сколько различных сдаваемых в аренду объектов были осмотрены.

```
SELECT COUNT(DISTINCT KodP) AS C  FROM V ;
```

Результаты выполнения запроса:

C
3

Общее количество осмотренных объектов, удовлетворяющих указанному условию, может быть определено с помощью обобщающей функции COUNT. Однако, поскольку один и тот же объект может быть осмотрен различными клиентами несколько раз, необходимо в определении функции указать

ключевое слово DISTINCT — это позволит исключить из расчета дублирующиеся значения.

Пример 6.15. Использование функций COUNT и SUM

Определить общее количество менеджеров компании и вычислить сумму их зарплаты.

```
SELECT COUNT(KodS) AS C, SUM(ZP) AS SZP
    FROM S
    WHERE Dol = 'Manager';
```

Ограничение на отбор сведений только о менеджерах компании достигается указанием в запросе соответствующего предложения WHERE. Общее количество менеджеров и сумма их годовой заработной платы определяются путем применения обобщающих функций COUNT и SUM соответственно.

Результаты выполнения запроса:

C	SZP
2	850

Пример 6. 16. Использование функций MIN, MAX и AVG

Вычислить значение минимальной, максимальной и средней заработной платы.

```
SELECT MIN(ZP) AS MinZP, MAX(ZP) AS MaxZP, AVG(ZP) AS AZP
    FROM S;
```

6.7. Группировка результатов (фраза GROUP BY)

Приведенные выше примеры сводных данных подобны итоговым строкам, обычно размещаемым в конце отчетов. В итогах все детальные данные отчета сжимаются в одну обобщающую строку. *Однако* очень часто в отчетах требуется формировать и промежуточные итоги. Для этой цели в операторе SELECT может указываться фраза GROUP BY. Запрос, в котором присутствует фраза GROUP BY, называется **группирующим запросом,** поскольку в нем группируются данные, полученные в результате выполнения операции

SELECT, после чего для каждой отдельной группы создается единственная суммарная строка. Столбцы, перечисленные во фразе GROUP BY, называются **группируемыми столбцами.**

Все имена столбцов, приведенные в списке предложения SELECT, должны присутствовать и во фразе GROUP BY — за исключением случаев, когда имя столбца используется в обобщающей функции. Если совместно с фразой GROUP BY используется предложение WHERE, то оно обрабатывается первым, а группированию подвергаются только те строки, которые удовлетворяют условию поиска.

Пример 6.16. Использование фразы GROUP BY.

Определить количество персонала, работающего в каждом из отделений компании, а также их суммарную заработную плату.

```
SELECT KodB, COUNT(KodS) AS C, SUM(ZP) AS SZP FROM S
    GROUP BY KodB
    ORDER BY KodB;
```

Концептуально, при обработке этого запроса выполняются следующие действия.

1. Строки таблицы S распределяются в группы в соответствии со значениями в столбце номера отделения компании. В пределах каждой из групп, оказываются данные обо всем персонале одного из отделений компании. В нашем примере будут созданы три группы.

KodB	KodS	ZP
B3	S37	250
B3	S14	400
B3	S05	200
B5	S41	450
B5	S21	500
B7	S09	240

2. Для каждой из групп вычисляется общее количество строк, равное количеству работников в отделении, а также сумма значений в столбце ZP.

Затем генерируется единственная сводная строка для всейгруппы исходных строк.

3. Наконец, полученные строки результирующей таблицы сортируются в порядке возрастания номера отделения.

Результирующая таблица:

KodB	C	SZP
B3	3	850
B5	2	950
B7	1	240

Ограничения на выполнение группирования (фраза HAVING)

Фраза HAVING предназначена для использования совместно с фразой GROUP BY для задания ограничений, указываемых с целью отбора тех групп**,** которые будут помещены в результирующую таблицу запроса. Хотя фраза HAVING и предложение WHERE имеют сходный синтаксис, их назначение различно. Предложение WHERE предназначено для фильтрации отдельных строк, используемых для группирования или помещаемых в результирующую таблицу запроса, тогда как фраза HAVING используется для фильтрации групп**,** помещаемых в результирующую таблицу запроса. Стандарт ISO требует, чтобы имена столбцов, используемые во фразе HAVING, обязательно присутствовали в списке фразы GROUP BY или применялись в обобщающих функциях. На практике условия поиска во фразе HAVING всегда включают, по меньшей мере, одну обобщающую функцию, в противном случае эти условия поиска должны быть помещены в предложение WHERE и применяться для отбора отдельных строк.

Пример 6.17. Использование фразы HAVING

Для каждого отделения компании с численностью персонала более одного человека определить количество работающих и сумму их заработной платы.

```
SELECT KodB, COUNT(KodS) AS C, SUM (ZP) AS SZP
    FROM S
```

```
GROUP BY KodB
HAVING COUNT(KodS) > 1
ORDER BY KodB;
```

Этот пример аналогичен предыдущему, но здесь используются дополнительные ограничения, указывающие на то, что нас интересуют сведения только о тех отделениях компании, в которых работает больше одного человека. Подобное требование налагается на группы, поэтому в запросе следует использовать фразу HAVING. Результат выполнения запроса представлен ниже.

KodB	C	SZP
B3	3	850
B5	2	950

6.8. Подзапросы

В этом разделе рассматривается использование операторов SELECT, внедренных в тело другого оператора SELECT. Внешний (первый) оператор SELECT использует результат выполнения внутреннего (второго) оператора для определения содержания окончательного результата всей операции. Внутренние запросы могут быть помещены в предложения WHERE и HAVING внешнего оператора SELECT — в этом случае они получают название подзапросов, или вложенных запросов. Текст подзапроса должен быть заключен в скобки. Кроме того, внутренние операторы SELECT могут использоваться в операторах INSERT, UPDATE и DELETE. Существует три типа подзапросов.

Скалярный подзапрос возвращает значение, выбираемое из пересечения одного столбца с одной строкой, — т.е. единственное значение. В принципе, скалярный подзапрос может использоваться везде, где требуется указать единственное значение.

Строковый подзапрос возвращает значения нескольких столбцов таблицы, но в виде единственной строки. Строковый подзапрос может использоваться везде, где применяется конструктор строковых значений — обычно это предикаты.

Табличный подзапрос возвращает значения одного или больше столбцов таблицы, размещенные в более чем одной строке. Табличный подзапрос может использоваться везде, где допускается указывать таблицу — например, как операнд предиката IN

Пример 6.18. Использование подзапроса с проверкой на равенство.

Получить список персонала, работающего в отделении компании, расположенном по адресу «Северная, 34».

```
SELECT KodS, Lname, Fname, Dol
        FROM S
       WERE KodB =
       (SELECT KodB FBOM B WHERE Street = 'Северная, 34');
```

Внутренний оператор предназначен для определения номера отделения компании, расположенного по указанному адресу. Внутренний оператор SELECT возвращает таблицу, состоящую из единственного значения 'B3'.

Результаты выполнения этого запроса:

KodS	Lname	Fname	Dol
S37	Петрова	Татьяна	Агент
S14	Сидоров	Стеман	Менеджер
S05	Степанова	Елена	Секретарь

Подзапрос представляет собой инструмент создания временной таблицы, содержимое которой извлекается и обрабатывается внешним оператором. Подзапрос может указываться непосредственно после операторов сравнения (т.е. операторов =, <, >, <=, >=, <>) в предложениях WHERE и HAVING.

Пример 6.19. Использование подзапросов с обобщающими функциями.

Получить список всех сотрудников, имеющих зарплату выше средней, указав, на насколько их зарплата превышает среднюю зарплату.

```
SELECT KodS, Fname, Lname, Dol,
        ZP -(SELECT AVG(ZP) FROM S) AS Dzp
        FROM S
        WERE ZP > (SELECT AVG(ZP) FROM S);
```

Необходимо отметить, что нельзя прямо использовать предложение WHERE ZP > AVG(ZP) поскольку применять обобщающие функции в предложениях WHERE запрещено.

Результат выполнения запроса:

KodS	Lname	Fname	Dol	Dzp
S21	Иванов	Иван	Руководитель	160
S14	Сидоров	Стеман	Менеджер	60
S41	Петров	Игорь	Менеджер	110

К подзапросам применяются следующие правила и ограничения.

1. В подзапросах не должна использоваться фраза ORDER BY, хотя она может присутствовать во внешнем запросе.

2. Список в предложении SELECT подзапроса должен состоять из имен отдельных столбцов или составленных из них выражений — за исключением случая, когда и подзапросе используется ключевое слово EXISTS (см. ниже).

3. По умолчанию имена столбцов в подзапросе относятся к таблице, имя которой указано в его предложении FROM. Однако допускается ссылаться и на столбцы таблицы, указанной во фразе FROM внешнего запроса, для чего используются квалифицированные имена столбцов (см. ниже).

4. Если подзапрос является одним из двух операндов, участвующих в операции сравнения, то запрос должен указываться в правой части этой операции.

Пример 6.20. Вложенные подзапросы и использование предиката IN

Получить перечень сдаваемых в аренду объектов, за которые отвечают сотрудники отделения компании, расположенного в г. Тюмень.

```
SELECT KodP, City, Street, Type, Rooms, Rent,
        FROM P
        WHERE KodO IN
        (SELECT KodS FROM S WHERE KodB IN
        (SELECT KodB FBOM B WHERE City = 'Тюмень')),
```

KodP	City	Street	Type	Rooms	Rent
P94	Нижневартовск	…	кварт.	4	400
P04	Тюмень		дом	4	350
P36	Ишим		кварт.	2	375
P21	Надым		кварт.	3	600
P16	Сургут		кварт.	2	450

Первый, самый внутренний, запрос предназначен для определения номеров отделений компании, расположенных в г. Тюмени.

Второй, промежуточный, осуществляет выборку сведений о персонале, работающем в этом отделе. В том и другом случаях выбирается более одной строки данных, и поэтому во внешних запросах нельзя использовать оператор сравнения =. Вместо него необходимо использовать ключевое слово IN.

6.9. Ключевые слова ANY и ALL

Ключевые слова ANY и ALL могут использоваться с подзапросами, возвращающими столбец чисел. Если подзапросу будет предшествовать ключевое слово ALL, условие сравнения считается выполненным только в том случае, если оно выполняется для всех значений в результирующем столбце подзапроса. Если записи подзапроса предшествует ключевое слово ANY, то условие сравнения будет считаться выполненным, если оно выполняется хотя бы для одного из значений в результирующем столбце подзапроса. Если в результате выполнения подзапроса будет получено пустое значение, то для

ключевого слова ALL условие сравнения будет считаться выполненным, а для ключевого слова ANY — невыполненным. Согласно стандарту ISO дополнительно можно использовать ключевое слово SOME, являющееся синонимом ключевого слова ANY.

Пример 9.21. Использование ключевого слова ANY

Получить список всех сотрудников, чья зарплата превышает зарплату хотя бы одного сотрудника отделения компании под номером 'B5'.

```
SLECT KodS, Lname, Fname, Dol, ZP FROM S
    WHERE ZP > ANY (SELECT ZP FROM S WHERE KodB = 'B5');
```

Внутренний подзапрос создает набор числовых значений (350, 500), а внешний запрос выбирает сведения о тех работниках, чья зарплата больше хотя бы одного из значений в этом наборе.

Результат выполнения запроса:

KodS	Lname	Fname	Dol	ZP
S21	Иванов	Иван	Руководитель	500
S14	Сидоров	Стеман	Менеджер	400
S41	Петров	Игорь	Менеджер	350

Пример 6.22. Использование ключевого слова ALL.

Получить список всех сотрудников, чья заработная плата больше заработной платы любого работника отделения компании под номером 'B3'.

```
SELECT KodS, Lname, Fname, Dol, ZP FROM S
    WHERE ZP > ANY (SELECT ZP FROM S WHERE KodB = 'B3');
```

Результат выполнения запроса:

KodS	Lname	Fname	Dol	ZP
S21	Иванов	Иван	Руководитель	500

6.10. Многотабличные запросы

Если результирующая таблица запроса должна содержать данные из разных таблиц, то необходимо использовать механизм соединения таблиц. Для выполнения соединения достаточно в предложении FROM указать имена двух

и более таблиц, разделив их запятыми, после чего включить в запрос предложение WHERE с определением столбцов, используемых для соединения указанных таблиц.

Пример 6.23. Простое соединение.

Получить список имен всех клиентов, которые уже осмотрели хотя бы один, сдаваемый в аренду объект.

```
SELECT R.KodR, Name, KodP, Comment
    FROM Renter R, Viewing V
    WHERE R.KodR = V.KodR
```

В этом отчете требуется представить сведения как из таблицы R, так и из таблицы V, поэтому при построении запроса мы воспользуемся механизмом соединения таблиц. В предложении SELECT перечисляются все столбцы, которые должны быть помещены в результирующую таблицу запроса. Обратите внимание что для столбца с номером клиента (KodR) необходимо указать квалификатор — этот столбец может быть выбран из любой соединяемой таблицы, поэтому необходимо точно указать, значения какой таблицы нас интересуют. Уточнение имени осуществляется посредством указания перед именем столбца имени той таблицы (или ее псевдонима), из которой он выбирается.

Для построения результирующих строк используются те строки исходных таблиц, которые имеют одинаковое значение в столбце KodR. Это условие определяется посредством задания условия поиска R.KodR = V.KodR. Подобные столбцы исходных таблиц называют сочетаемыми столбцами.

Результат выполнения запроса:

KodR	Name	KodP	Comment
R56	Рубин Степан	P14	Мала
R56	Рубин Степан	P04	
R56	Рубин Степан	P36	
R62	Зимина Елена	P14	Дорого
R76	Саблев Иван	P04	Далеко

Обычно многотабличные запросы выполняются для двух таблиц, соединенных связью типа 1:M.

Стандарт SQL2 дополнительно предоставляет следующие способы определения данного соединения:

```
... FROM Renter R JOIN Viewing V ON R.KodR = V.KodR
... FROM Renter JOIN Viewing USING KodR
... FROM Renter NATURAL JOIN Viewing
```

Пример 6.24. Соединение трех таблиц.

Для каждого отделения компании получить список работников, отвечающих за какие-либо сдаваемые в аренду объекты, с указанием города, в котором расположено данное отделение компании, и номеров объектов.

```
SELECT B.KodB, B.City, S.KodS, Fname, Lname, KodP
    FROM B, S, P
    WHERE B.KodB = S.KodB AND S KodS=P.KodS
    ORDER BY B.KodB, S.KodS, KodP.
```

В результирующую таблицу необходимо поместить столбцы из трех исходных таблиц — B, S и P. Поэтому в запросе следует выполнить соединение этих таблиц.

Результат выполнения запроса:

KodB	City	KodS	Fname	Lname	KodP
B3	Тюмень	S14	Сидоров	Стеман	P04
B3	Тюмень	S14	Сидоров	Стеман	P16
B3	Тюмень	S37	Петрова	Татьяна	P21
B3	Тюмень	S37	Петрова	Татьяна	P36
B5	Тюмень	S41	Петров	Игорь	P94
B7	Нижневартовск	S09	Зотова	Ирина	P14

Пример 6.26. Группирование по нескольким столбцом.

Определить количество сдаваемых в аренду объектов, за которые отвечает каждый из работников компании.

```
SELECT S.KodB, S.KodS, COUNT(*) AS C
        FROM S, P
        WHERE S.KodS = P.KodS
        GROUP BY S.KodB, S.KodS
        ORDER BY S.KodB, S.KodS;
```

Чтобы составить требуемый отчет, прежде всего, необходимо выяснить, какие из работников компании отвечают за сдаваемые в аренду объекты. Эту задачу можно решить посредством соединения таблиц S и P по ключу KodS.

Затем необходимо сформировать группы, состоящие из номера отделения и табельных номеров его работников, для чего следует определить предложение GROUP BY. Наконец, результирующая таблица должна быть упорядочена с помощью задания предложения ORDER BY.

Результаты выполнения запроса:

KodB	KodS	C
B3	S14	2
B3	S37	2
B5	S41	1
B7	S09	1

Выполнение соединений

Соединение является подмножеством более общей комбинации данных двух таблиц, называемой их декартовым произведением. Декартово произведение двух таблиц представляет собой другую таблицу, состоящую из всех возможных пар строк, входящих в состав обеих таблиц. Набор столбцов результирующей таблицы представляет собой все столбцы первой таблицы, за которыми следуют все столбцы второй таблицы. Если ввести запрос к двум таблицам без задания предложения WHERE, результат выполнения запроса в среде SQL будет представлять собой декартово произведение содержимого этих таблиц. Кроме того, стандарт предусматривает специальный формат

оператора SELECT, позволяющий вычислить декартово произведение двух таблиц:

```
SELECT [DISTINCT | ALL] {column_list}
        FROM table_name1 CROSS JOIN table_name2
```

Процедура генерации результирующей таблицы содержащего соединение оператора SELECT состоит в следующем.

1. Формируется декартово произведение таблиц, указанных в предложении FROM.

2. Если в запросе присутствует предложение WHERE, применение условий поиска к каждой строке таблицы производится с сохранением в таблице только тех строк, которые удовлетворяют заданным условиям. В терминах реляционной алгебры эта операция называется **ограничением** декартового произведения.

3. Для каждой оставшейся строки определяется значение каждого элемента, указанного в списке предложения SELECT, в результате чего формируется отдельная строка результирующей таблицы.

4. Если в исходном запросе присутствует фраза SELECT DISTINCT, из результирующей таблицы удаляются все строки-дубли. В реляционной алгебре действия, выполняемые на 3 и 4 этапах, эквивалентны операции **проекции** по столбцам, заданным в списке предложения SELECT.

Открытые соединения

При выполнении операции соединения данные из двух таблиц комбинируются с образованием пар связанных строк, в которых значения сопоставляемых столбцов одинаковые. Если строка одной из таблиц не находит себе соответствия в другой, то она не попадает в результирующий набор данных. Именно это правило применялось во всех рассмотренных выше примерах соединения таблиц. Стандартом ISO предусмотрен и другой набор операторов соединений, называемых **открытыми соединениями**. При

открытом соединении в результирующую таблицу помещаются также строки, не удовлетворяющие условию соединения. Чтобы понять особенности выполнения операций открытых соединений, воспользуемся упрощенными таблицами B и P, содержимое которых представлено в табл. 13.31 и 13.32.

B

KodB	Bcity
B3	Надым
B4	Ишим
B2	Тюмень

P

KodP	Pcity
P14	Сургут
P94	Тюмень
P04	Надым

Обычное (закрытое) соединение этих таблиц выполняется с помощью следующего SQL-оператора:

```
SELECT B.*, P.* FROM B, P WHERE B.Bcity = P.Pcity;
```

Результаты выполнения этого запроса:

KodB	Bcity	KodP	Pcity
B3	Надым	P04	Надым
B2	Тюмень	P94	Тюмень

Как можно видеть, в результирующей таблице запроса имеются только две строки, содержащие одинаковые названия городов, выбранные из обеих таблиц, в результирующих таблицах нет строк для отделения компании в Ишиме и для объекта, сдаваемого в аренду в городе Сургут. Если в результирующую таблицу потребуется включить и эти не имеющие соответствия строки, то следует использовать открытое соединение. Существует три типа открытых соединений: левое, правое и полное открытое. Рассмотрим особенности каждого из них на приведенных ниже примерах.

Пример 6.27. Левое открытое соединение.

Получить список отделений компании и сдаваемые в аренду объекты, которые расположены в одном и том же городе, а также прочие отделения компании.

Используем левое открытое соединение этих двух таблиц, которое выглядит следующим образом:

```
SELECT B.*, P.*
        FROM B LEFT JOIN P ON B,Bcity = P.Pcity;
```

Результаты выполнения этого запроса:

KodB	Bcity	KodP	Pcity
B3	Надым	P04	Надым
B4	Ишим	Null	Null
B2	Тюмень	P94	Тюмень

В этом примере за счет применения левого открытого соединения в результирующую таблицу попали не только две строки, в которых имеется соответствие между названиями городов, но также та строка первой из соединяемых таблиц (левой), которая не нашла себе соответствия во второй таблице. В этой строке все поля второй таблицы заполнены значениями NULL.

Пример 6.28. Правое открытое соединение.

Получить список отделений компании и сдаваемые в аренду объекты, которые расположены в одном и том же городе, а также все остальные объекты.

Используем правое открытое соединение этих двух таблиц, которое выглядит следующим образом:

```
SELECT B.*, P.*
        FROM B RIGHT JOIN P ON B,Bcity = P.Pcity;
```

Результаты выполнения этого запроса:

KodB	Bcity	KodP	Pcity
Null	Null	P14	Сургут
B3	Надым	P04	Надым
B2	Тюмень	P94	Тюмень

В этом примере при выполнении правого открытого соединения в результирующую таблицу были включены не только те две строки, которые

имеют одинаковые значения в сопоставляемых столбцах, но также и та строка из второй (правой), таблицы, которой не нашлось соответствия в первой (левой) таблице. В этой строке все поля из первой таблицы получили значения NULL.

Пример 6.29. Полное открытое соединение.

Получить список отделений компании и сдаваемые в аренду объекты, расположенные, в одном и том же городе, а также все остальные отделения и объекты.

.Используем полное открытое соединение этих таблиц, которое выглядит следующим образом:

```
SELECT B.*, P.*
        FROM B FULL JOIN P ON B,Bcity = P.Pcity;
```

Результаты выполнения этого запроса:

KodB	Bcity	KodP	Pcity
Null	Null	P14	Сургут
B3	Надым	P04	Надым
B4	Ишим	Null	Null
B2	Тюмень	P94	Тюмень

В случае полного открытого соединения в результирующую таблицу помещаются не только те две строки, которые имеют одинаковые значения в сопоставляемых столбцах, но и все остальные строки исходных таблиц, не нашедшие себе соответствие. В этих строках все столбцы той таблицы, в которой не было найдено соответствия, заполняются значениями NULL.

6.11. Ключевые слова EXISTS и NOT EXISTS

Ключевые слова EXISTS и NOT EXISTS предназначены для использования только совместно с подзапросами. Результат их обработки представляет собой логическое значение TRUE или FALSE. Для ключевого слова EXISTS результат равен TRUE в том и только в том случае, если в возвращаемой подзапросом результирующей таблице присутствует хотя бы

одна строка. Если результирующая таблица подзапроса пуста, результатом обработки ключевого слова EXISTS будет значение FALSE. Для ключевого слова NOT EXISTS используются правила обработки, обратные по отношению к ключевому слову EXISTS.

Поскольку по ключевым словам EXISTS и NOT EXISTS проверяется лишь наличие строк в результирующей таблице подзапроса, то эта таблица может содержать произвольное количество столбцов. Как правило, с целью упрощения во всех следующих за обсуждаемыми ключевыми словами подзапросах применяется такая форма записи:

Пример 6.30. Запрос с использованием ключевого слова EXISTS

Получить список всех сотрудников компании, которые работают в ее отделениях в г. Тюмени.

```
SELECT KodS, Lname, Fname, Dol
    FROM S
    WHERE EXISTS
    (SELECT * FROM B
    WHERE S.KodB = B.KodB AND City = 'Тюмень');
```

Результаты выполнения запроса:

KodS	Lname	Fname	Dol
S21	Иванов	Иван	Руководитель
S41	Петров	Игорь	Менеджер

Обратите внимание, что первая часть условия поиска, S.KodB = B.KodB, необходима для получения гарантий того, что для каждого работника будет анализироваться корректная строка данных об отделении компании. Если опустить это условие, то в результирующую таблицу запроса будут помещены сведения обо всех работниках компании, поскольку подзапрос всегда будет возвращать не менее одной строки и проверка существования в каждом случае будет давать значение TRUE.

Кроме того, этот запрос можно записать, используя методы соединения:

```
SELECT KodS, Lname, Fname, Dol  FROM S, B
        WHERE  S.KodB = B.KodB AND City = 'Тюмень';
```

6.12. Комбинирование результирующих таблиц (операции UNION, INTERSECT и EXCEPT)

В языке SQL можно использовать обычные операции над множествами – объединение (UNION), пересечение (INTERSECTION) и разность (EXCEPT), позволяющие комбинировать результаты выполнения двух и более запросов в единую результирующую таблицу.

Объединением (UNION) двух таблиц и называется таблица, содержащая все строки, которые имеются в первой таблице, во второй таблице или в обеих этих таблицах сразу.

Пересечением (INTERSECT) двух таблиц называется таблица, содержащая все строки, присутствующие в обеих исходных таблицах одновременно.

Разностью (EXCEPT) двух таблиц A и B называется таблица, содержащая все строки, которые присутствуют в таблице A, но отсутствуют в таблице B.

На таблицы, которые могут комбинироваться с помощью операций над множествами, накладываются определенные ограничения. Самое важное из них состоит в том, что таблицы должны быть **совместимы по соединению —** т.е. они должны иметь одну и ту же структуру. Это означает, что таблицы должны иметь одинаковое количество столбцов, причем в соответствующих столбцах должны размещаться данные одного и того же типа и длины.

Три операции над множествами, предусмотренные стандартом ISO, носят название UNION, INTERSECT и EXCEPT. В каждом случае формат предложения с операцией над множествами должен быть следующим:

```
operator [ALL] [CORRESPONDING [BY (column1 [,... ]))]
```

При указании фразы CORRESPONDING BY операция над множествами выполняется для указанных столбцов. Если задано только ключевое слово CORRESPONDING, а фраза BY отсутствует, операция над множествами выполняется для столбцов, которые являются общими для обеих таблиц. Если указано ключевое слово ALL, результирующая таблица может содержать дублирующиеся строки. Одни диалекты языка SQL не поддерживают операций INTERSECT и EXCEPT, а в других вместо ключевого слова EXCEPT используется ключевое слово MINUS.

Пример 6.31. Использование операции UNION.

Получить список всех городов, в которых либо находится отделение компании, либо располагаются сдаваемые в аренду объекты.

```
(SELECT City FROM B)  UNION  (SELECT City FROM P)
```

Этот запрос выполняется посредством подготовки результирующей таблицы первого запроса и результирующей таблицы второго запроса с последующим слиянием обеих таблиц в единую результирующую таблицу, которая включает все строки из обеих таблиц, но с удалением дублирующихся строк.

Результат выполнения запроса:

City
Тюмень
Нижневартовск
Ишим
Надым
Сургут

Пример 6.32. Использвоание операции INTERSECT. Создайте список всех городов, в которых располагаются и отделения компании, и сдаваемые в аренду объекты.

```
(SELECT city FROM B)
INTERSECT
(SELECT city FROM P);
```

или

(SELECT * FROM B)

INTERSECT CORRESPONDING BY city

(SELECT * : FROM P)

Этот запрос выполняется посредством подготовки результирующей таблицы первого запроса и результирующей таблицы второго запроса с последующим созданием единой результирующей таблицы, включающей только те строки, которые являются общими для обеих промежуточных таблиц.

Этот запрос можно записать и без использования операции INTERSECT:

(SELECT B.city FROM B, P WHERE B.city = P.city;

Пример 6.33. Использование операции EXCEPT. Создайте список всех городов, в которых имеется отделение компании, но нет ' сдаваемых в аренду объектов.

(SELECT city FROM branch)

EXCEPT

(SELECT city (SELECT * FROM P)

Этот запрос выполняется посредством подготовки результирующей таблицы первого запроса и результирующей таблицы второго запроса с последующим созданием единой результирующей таблицы, включающей только те строки, которые имеются в первой промежуточной таблице, но отсутствуют во второй.

Этот запрос можно записать и без использования операции EXCEPT:

SELECT DISTINCT city FROM B WHERE city NOT IN

(SELECT city FROM P)

6.13. Изменение содержимого базы данных

Язык SQL является полнофункциональным языком манипулирования данными, который может использоваться не только для выборки данных из базы, но и для модификации ее содержимого. К ним относятся три оператора языка SQL, предназначенных для модификации содержимого базы данных:

- INSERT — для добавления данных в таблицу;
- UPDATE — для модификации уже помещенных в таблицу данных;
- DELETE — позволяет удалять из таблицы строки данных.

Добавление новых данных в таблицу (оператор INSERT)

Существует две формы оператора INSERT. Первая предназначена для вставки единственной строки в указанную таблицу. Эта форма оператора INSERT имеет следующий формат:

```
INSERT INTO table_name [(column_list)] VALUES
        (data_value_list)
```

Здесь параметр *table_name* - имя таблицы - может представлять либо имя таблицы базы данных, либо имя обновляемого представления.

Параметр column_list - список столбцов - представляет собой список имен столбцов. Параметр является необязательным. Если он опущен, то предполагается использование списка из имен всех столбцов таблицы, указанных в том порядке, в котором они были описаны в операторе CREATE TABLE. Если в операторе INSERT указывается конкретный список имен столбцов, то любые опущенные в нем столбцы должны быть объявлены при создании таблицы как допускающие значение NULL — за исключением случаев, когда при описании столбца использовался параметр DEFAULT.

Параметр *data_value_list* - список значений данных - должен следующим образом соответствовать параметру *column_list:*

• количество элементов в обоих списках должно быть одинаковым;

• должно существовать прямое соответствие между позициями элементов в обоих списках (первый элемент списка *data_value_list* полагается относящимся к первому элементу списка *column_list* и т. д.);

• типы данных элементов списка *data value list* должны быть совместимы с типом данных соответствующих столбцов таблицы.

Пример 6.34. Использование конструкции INSERT

Добавить в таблицу S новую запись, содержащую данные во всех столбцах.

```
INSERT INTO S  VALUES
        ('S16', 'Кузьмин', 'Петр', 'M', DATE '1957-05-25', 'Агент',
        430, 'B3');
```

Пример 6.35. Вставка новой записи с использованием значений, принимаемых по умолчанию

Добавить в таблицу S новую запись, содержащую данные во всех обязательных столбцах KodS , LName, FName, Dol, ZP и KodB.

```
INSERT INTO S (KodS , LName, FName, Dol, ZP и KodB)
        VALUES ('S44', 'Семенова', 'Ирина', 'Агент', 480, 'B5');
```

Поскольку данные вставляются только в определенные столбцы таблицы, необходимо указать имена столбцов, в которые эти данные будут помещаться. Порядок следования имен столбцов несущественен, однако более естественно указать их в том порядке, в каком они следуют в таблице. Кроме того, оператор INSERT можно было бы записать и таким образом (явное указание, что в столбцы Pol и DR должны быть помещены значения NULL):

```
INSERT INTO S
        VALUES ('S44', 'Семенова', 'Ирина', NULL, NULL, 'Агент',
        480, 'B5');
```

Вторая форма оператора INSERT позволяет скопировать множество строк одной таблицы в другую таблицу. Этот оператор имеет следующий формат:

```
INSERT INTO table_name [(column_list)] SELECT …
```

Предложение SELECT может представлять собой любой допустимый оператор SELECT. Строки, вставляемые в указанную таблицу, соответствуют строкам результирующей таблицы, созданной при выполнении вложенного

запроса. Все ограничения, указанные выше для первой формы оператора INSERT, применимы и в этом случае.

Пример 6.36. Использование конструкции INSERT … SEIECT

Предположим, что существует таблица SP, у которой имеются столбцы для хранения кода сотрудника, фамилии, имени работников компании и количества сдаваемых в аренду объектов, за которые они отвечают:

SP(KodS, Lname, Fname, Pcount)

Необходимо заполнить таблицу SP данными, используя информацию из таблиц S и P

```
INSERT INTO SP
        (SELECT S.KodS, Lname, Fname, Pcount, COUNT(*)
                FROM S, P WHERE S.KodS = P.KodS
               GROUP BY S.KodS, Lname, Fname)
        UNION
        (SELECT KodS, Lname, Fname, 0
                FROM S WHERE KodS NOT IN
                (SELECT DISTINCT KodS FROM P));
```

Если опустить вторую часть операции UNION, то будет создан список только тех работников компании, которые в настоящее время отвечают хотя бы за один объект. Иначе говоря, из результатов будут исключены все работники, которые в данный момент не отвечают ни за один сдаваемый в аренду объект. По этой причине для составления полного списка работников компании необходимо использовать оператор UNION, в котором второй оператор SELECT предназначен для выборки сведений именно о таких работниках, причем в столбец Pcount соответствующих строк помещается значение 0.

KodS	Lname	Fname	Pcount
S21	Иванов	Иван	0
S37	Петрова	Татьяна	2
S14	Сидоров	Степан	2
S09	Зотова	Ирина	1

S05	Степанова	Елена	0
S41	Петров	Игорь	1

Модификация данных в базе (оператор UPDATE)

Оператор UPDATE позволяет изменять содержимое уже существующих строк указанной таблицы. Этот оператор имеет следующий формат:

```
UPDATE имя_таблицы
    SET имя_столбца1 = значение1
    [, имя_столбца2 = значение2 … ]
      [WHERE предикат]
```

Здесь параметр имя_таблицы представляет либо имя таблицы базы данных, либо имя обновляемого представления. В предложении SET указываются имена одного или более столбцов, данные в которых необходимо изменить.

Предложение WHERE является необязательным. Если оно опущено, значения указанных столбцов будут изменены во всех строках таблицы. Если предложение WHERE присутствует, то обновлены будут только те строки, которые удовлетворяют условию поиска, заданному в параметре предикат. Параметры значение1 представляют новые значения соответствующих столбцов и должны быть совместимы с ними по типу данных.

Пример 6.37. Обновление всех строк таблицы

Повысить всему персоналу заработную плату на 3%.

```
UPDATE S SET ZP = ZP*1.03;
```

Поскольку изменения касаются всех строк таблицы, предложение WHERE в этом примере указывать не требуется.

Пример 6.38. Обновление некоторых строк таблицы.

Повысить заработную плату всем менеджерам компании на **5%.**

```
UPDATE S SET ZP = ZP*1.05  WHERE Dol = 'Менеджер';
```

Пример 6.39. Обновление нескольких столбцов

Перевести Петрова Игоря (KodS= 'S41') на должность руководителя и повысьте ему зарплату до 550.

```
UPDATE S SET Dol = 'Руководитель', ZP = 550
        WHERE  KodS= 'S41';
```

Удаление данных из базы (оператор DELETE)

Оператор DELETE позволяет удалять строки данных из указанной таблицы. Этот оператор имеет следующий формат:

```
DELETE FROM имя_таблицы [WHERE предикат]
```

Как и в случае операторов INSERT и UPDATE, параметр имя_таблицы может представлять собой либо имя таблицы базы данных, либо имя обновляемого представления. Предложение WHERE является необязательным — если оно опущено, из таблицы будут удалены все существующие в ней строки (сама таблица удалена не будет). Если необходимо удалить не только содержимое таблицы, но и ее определение, следует использовать оператор DROP TABLE. Если предложение WHERE присутствует, из таблицы будут удалены только те строки, которые удовлетворяют условию отбора.

Пример 6.40. Удаление определенных строк таблицы.

Удалить все записи об осмотрах сдаваемого в аренду объекта с учетным номером P4.

```
DELETE FROM V WHERE KodP = 'P4';
```

Пример 6.41. Удаление всех строк таблицы.

Удалить все строки из таблицы V.

```
DELETE FROM M;
```

Поскольку в данном операторе предложение WHERE не указано, будут удалены все строки таблицы. В результате в базе данных сохранится лишь описание таблицы V, что в дальнейшем позволит ввести в нее новую информацию.

Задания 6

Задание 6.1.

В БД «Продажа товаров» имеется 3 таблицы: Т – товары, К – клиенты, Р – продажи. Имена полей заданы.

Т		
Код товара	Товар	Цена
KT	T	P

K		
Код клиента	Фамилия	Город
KK	F	G

P			
Код продажи	Код клиента	Код товара	Количество
KP	KK	KT	K

Построить запросы:

1. Список товаров, цена которых больше 100 и меньше 200;
2. Список клиентов, отсортированных по алфавиту, фамилия которых начинается на букву «М»;
3. Список клиентов, проживающих в городах Тюмень, Сургут и Тобольск;
4. Список городов, в которых имеются клиенты с указанием их количества;
5. Список кодов клиентов, которые делали покупки (без повторов);
6. Список фамилий клиентов, которые делали покупки;
7. Список товаров, общее число продаж которых более 100;
8. Список фамилий клиентов с указанием их общей суммы покупок. Если клиент не делал покупок, он все равно должен присутствовать в этом списке.
9. Список товаров, с указанием количества купленного. Если товар не покупался, выводить 0.
10. Список фамилий клиентов, которые не делали покупки;
11. Получить таблицу:

Код товара	Товар	Общее количество продаж
KT	T	K

Задание 6.2.

В БД «Сессия» имеется 3 таблицы: St – студенты, Dis – дисциплины, Ex – экзамены. Имена полей заданы.

ST		
Код студента	Фамилия	Группа
KS	F	G

DIS	
Код дисциплины	Название
KD	D

EX				
Код экзамена	Код студента	Код дисциплины	Оценка	Дата
KE	KS	KD	Z	DT

Построить запросы для получения следующей информации:

1. Список студентов групп 321, 341 и 351, упорядоченный сначала по убыванию по группам, затем по возрастанию по фамилиям (не проверять на =);
2. Список кодов экзаменов, сданных в марте 2003 г.
3. Список дисциплин, которые начинаются на букву «М»;
4. Список кодов дисциплин, по которым сдавались экзамены (без повторов);
5. Список студентов, которые сдавали экзамены с указанием количества сданных;
6. Список студентов, которые не сдавали ни одного экзамена;
7. Список студентов, сдавших 4 экзамена и обучающихся на 4 и 5;
8. Список дисциплин, с указанием количества студентов (поле KOL), которые ее сдали и среднего балла по дисциплине (поле SR). Если дисциплина не сдавалась ни одним студентом, она должна присутствовать в списке, KOL=0, SR=0;
9. Добавить в таблицу DIS информацию о дисциплине «Информатика» с кодом 331
10. Удалить из таблицы ST студентов 335 группы

Задание 6.3.

В БД «Библиотека» имеется 3 таблицы: KN – книги, KL – клиенты, VD – выдача книг.

У книги всегда только один автор.

Имена полей заданы, первичные ключи подчеркнуты.

KN		
Код книги	Название	Автор
KKN	N	A

KL	
Код клиента	Фамилия
KK	F

VD				
Код	Код книги	Код клиента	Дата	Срок
K	KKN	KK	DT	SR

Построить запросы для получения следующей информации:

1. Список книг авторов «Иванов», «Петров» и «Андреев», упорядоченный сначала по убыванию по авторам, затем по возрастанию по названиям (не проверять на =);
2. Список кодов книг, выданных в I полугодии 2002 г.
3. Список клиентов, фамилии которых заканчиваются на «ов»;
4. Список кодов книг, которые выдавались (без повторов);
5. Список клиентов, которым выдавались книги с указанием количества выдач;
6. Список клиентов с указанием количества различных книг, которые он брал (повторные не учитывать)
7. Список книг, которые не выдавались;
8. Список клиентов, бравших книги более 5 раз;
9. Список клиентов с полем KOL, содержащим количество выдач книг данному клиенту. Если клиенту книги не выдавались, значение этого поля должно быть не определено (использовать внешнее соединение).

10. Список клиентов, бравших одну и ту же книгу более 1 раза. В списке отобразить название книги и сколько раз она бралась.
11. Список книг, с указанием сколько раз она выдавалась (поле KOL), и среднего срока выдачи (поле SR). Если книга не выдавалась, она должна присутствовать в списке, KOL=0, SR=0;
12. Список книг, которые брались более 10 раз на срок не менее 30 дней.
13. Добавить в БД информацию о книге«Колобок», с кодом 234, автор не задан.
14. Удалить из БД информацию о клиентах, которые ни разу не брали книги.
15. Изменить для книги «Колобок» с кодом 234 автора на «народ».

Задание 6.4.

В БД «Аренда» имеется 3 таблицы: A – арендаторы, P – объекты аренды, S – сведения об аренде. Имена полей заданы, первичные ключи подчеркнуты.

P		
Код объекта	Тип	Цена за месяц
KP	T	C

A	
Код арендатора	Фамилия
KA	F

S				
Код	Код объекта	Код арендатора	Дата начала	Продолжи-тельность (мес.)
K	KP	KA	DT	SR

Построить запросы для получения следующей информации:

1. Список объектов типа «квартира» или «дом», упорядоченный сначала по убыванию по алфавиту, затем по возрастанию цены;
2. Список арендаторов, которым сдавались объекты с указанием количества аренд;
3. Список объектов, которые сдавались в аренду (без повторов);

4. Список объектов, которые не сдавались;
5. Список объектов, которые сдавались более 3 раз;
6. Список объектов, которые сдавались в аренду больше 2 раз на срок более 1 года со столбцом KOL – количество таких аренд.
7. Список арендаторов с указанием количества **различных** арендуемых объектов;
8. Список объектов со столбцами KO и KS, содержащими количество сдач каждого объекта и выплаченную общую сумму. Если объект не сдавался, значения в этих столбцах должно быть не определено (использовать внешнее соединение).
9. Список арендаторов, с указанием сколько раз он арендовал объекты (поле KOL), и среднего срока аренды (поле SR). Если арендатор не арендовал никаких помещений, он должен присутствовать в списке с столбцами KOL=0, SR=0;
10. Список объектов, которые арендовались одним и тем же арендатором более одного раза. В таблице должны быть столбцы: KP, T, KA, F, Kol – сколько раз объект арендовался арендатором.
11. Список объектов с указанием типа, сданных в аренду в I квартале 2002 г. Упорядочить по дате начала аренды.

Задание 6.5.

В БД «Авиабилеты» имеется 3 таблицы: **R** – рейсы, **P** – пассажиры, **F** – полеты. Имена полей заданы, первичные ключи подчеркнуты.

R		
Код рейса	Аэропорт (назначения)	Цена
KR	NR	PR

P	
Код пассажира	Фамилия
KP	FIO

F			
Код	Код рейса	Код клиента	Дата
KF	KR	KP	DV

В запросах, если требуется получить список аэропортов, необходимо указать полную информацию: код, название, цена. Аналогично с пассажирами.

Построить запросы для получения следующей информации:

1. Среднюю цену авиабилета и количество рейсов до каждого аэропорта;
2. Список аэропортов, до которых совершались полеты (без повторов);
3. Список пассажиров, которые не совершали полеты в 2006 году;
4. Список пассажиров, с информацией о количестве совершенных полетов;
5. Список пассажиров с указанием аэропортов, до которых они совершали полеты и количество таких полетов;
6. Список пассажиров, совершивших более 5 полетов до аэропорта Внуково;
7. Список пассажиров с полем S, содержащим количество совершенных полетов. Если пассажир полеты не совершал, значение этого поля должно быть не определено (использовать внешнее соединение).
8. Список аэропортов, с указанием сколько пассажиров (поле KOL) совершили до него полеты (повторные полеты не учитывать). Если до аэропорта полеты не совершались, он должен присутствовать в списке, KOL=0;
9. Список пассажиров, совершивших полет в один аэропорт более 5 раз. В списке отобразить название аэропорта и количество полетов.

Тема 7. ЯЗЫК SQL.
ПОДЪЯЗЫК ОПРЕДЕЛЕНИЯ ДАННЫХ DDL

Подъязык DDL (Data Definition Language), позволяет создавать и уничтожать различные объекты базы данных — например, схемы, домены, таблицы, представления или индексы. Рассмотрим операторы создания и удаления схемы, таблицы и индексы.

Основными операторами языка SQL, предназначенными для определения данных, являются следующие:

Создание	Модификация	Удаление
CREATE SCHEMA		DROP SCHEMA
CREATE DOMAIN	ALTER DOMAIN	DROP DOMAIN
CREATE TABLE	ALTER TABLE	DROP TABLE
CREATE VIEW		DROP VIEW

Эти операторы используются для создания, изменения и уничтожения структур из которых состоят концептуальные схемы.

Кроме того, многие реализации языка SQL дополнительно включают следующие операторы CREATE INDEX и DROP INDEX

7.1. Идентификаторы языка SQL

Идентификаторы языка SQL предназначены для обозначения объектов в базе данных и являются именами таблиц, представлений и столбцов. Символы, которые могут использоваться в создаваемых пользователем идентификаторах языка SQL, должны быть определены как набор символов**.** Стандарт ISO задает набор символов, который должен использоваться по умолчанию, — он включает строчные и прописные буквы латинского алфавита (A-Z, a-z), цифры (0-9) и символ подчеркивания. На формат идентификаторов накладываются следующие ограничения:

• идентификатор может иметь длину до 128 символов (большинство диалектов предусматривает более жесткие ограничения);

• идентификатор должен начинаться с буквы;

• идентификатор не может содержать пробелов.

7.2. Типы данных языка SQL, определенные стандартом ISO

В языке SQL существует шесть скалярных типов данных.

Символьные данные (тип character)

Для определения данных символьного типа используется следующий формат:

CHARACTER [VARYING] [length]

могут использоваться сокращения:

CHARACTER - CHAR,

CHARACTER VARYING - VARCHAR.

При определении столбца с символьным типом данных параметр length используется для указания максимального количества символов, которые могут быть помещены в данный столбец (по умолчанию принимается значение 1). Если строка определена с фиксированной длинной, то при вводе в нее меньшего количества символов значение дополняется до указанной длины пробелами, добавляемыми справа. Если строка определена с переменной длинной, то при вводе в нее меньшего количества символов в базе данных будут сохранены только введенные символы, что позволяет достичь определенной экономии внешней памяти.

Например, столбец Pind таблицы B имеет фиксированную длину в 6 символов и может быть объявлен следующим образом:

Pind CHAR(6)

Битовые данные (тип bit)

Битовый тип данных используется для определения битовых строк, т.е. последовательности двоичных цифр (битов), каждая из которых может иметь

значение либо 0, либо 1. Для определения данных битового типа используется формат:

BIT [VARYING] [length]

Например, для сохранения битовой строки с фиксированной длиной и значением '0011' может быть объявлен столбец BS:

BS BIT(4)

Точные числа (тип exact numeric)

Тип точных числовых данных используется для определения чисел, которые имеют точное представление в компьютере. Числа состоят из цифр, необязательной десятичной точки и необязательного символа знака. Данные точного числового типа определяются **значностью** *(precision)* и длиной **дробной части** (scale). Значность задает общее количество значащих десятичных цифр числа, в которое входят длина целой и дробной частей, но без учета самой десятичной точки. Дробная часть указывает количество дробных десятичных разрядов числа. Существует несколько способов определения данных точного числового типа:

NUMERIC [*precision* [, scale]]

DECIMAL [*precision* [, scale]]

INTEGER

SMALLINT

INTEGER может быть сокращено до INT, а DECIMAL до DEC.

Типы NUMERIC и DECIMAL предназначены для хранения чисел в десятичном формате. По умолчанию длина дробной части равна нулю, а принимаемая по умолчанию значность зависит от реализации. Для столбца Rooms таблицы P, в котором сохраняются сведения о количестве комнат сдаваемого в аренду объекта, можно выбрать тип SMALLINT и объявить его следующим образом:

Rooms SMALLINT

Столбец ZP таблицы S может быть объявлен следующим образом:

```
ZP DECIMAL (6,2)
```

В этом случае максимальное значение составит 9 999.99.

Округленные числа (тип *approximate numeric)*

Тип округленных чисел используется для описания действительных чисел. Существует несколько способов определения данных с типом округленных чисел:

```
FLOAT [precision]
REAL
DOUBLE PRECISION
```

Параметр *precision* задает количество знаков мантиссы. Количество знаков мантиссы определений типа REAL и DOUBLE PRECISION зависит от конкретной реализации.

Дата и время (тип datetime)

Тип данных "дата/время" используется для определения моментов времени с некоторой установленной точностью. Стандарт ISO разделяет тип данных "дата/время" на подтипы YEAR (Год), MONTH (Месяц), DAY (День), HOUR (Час), MINUTE (Минута), SECOND (Секунда)). Поддерживается три типа полей даты/времени:

```
DATE
TIME [time_precision]
TIMESTAMP [time_precision]
```

Тип данных DATE используется для хранения календарных дат, включающих поля YEAR, MONTH и DAY. Тип данных TIME используется для хранения отметок времени, включающих поля HOUR, MINUTE и SECOND. Тип данных TIMESTAMP используется для совместного хранения даты и момента времени. Параметр time_precision задает количество дробных

десятичных знаков, определяющих точность сохранения значения в поле SECOND. Если этот параметр опускается, по умолчанию его значение для столбцов типа TIME принимается равным нулю (т.е. сохраняются целые секунды), тогда как для полей типа TIMESTAMP он принимается равным 6 (т.е. отметки времени сохраняются с точностью до миллисекунд).

Скалярные функции

Язык SQL включает некоторое количество встроенных скалярных операторов и функций, которые могут использоваться для построения скалярных выражений:

Оператор	Назначение
BIT_LENGTH	Возвращает длину заданной строки в битах. Например, результат вычисления выражения BIT_LENGTH(X'FFFF') равен 16
CHAR_LENGTH	Возвращает длину заданной строки в символах (или в октетах, если строка является битовой строкой). Например, результат вычисления выражения CHAR_LENGTH('Beech') равен 5
CAST(… AS …)	Преобразует значение выражения, построенного из данных одного типа, в значение другого типа данных.
||	Оператор конкатенации. Соединенные этим оператором две символьные или битовые строки преобразуются в одну строку. Например, выражение Lname || Fname
LOWER	Функция преобразует в заданной строке все прописные буквы в строчные. Например, в результате вычисления выражения ___будет получено значение
UPPER	Функция преобразует в заданной строке все строчные буквы в прописные. На- пример, в результате вычисления выражения __ будет получено значение __
TRIM	Функция удаляет ведущие (LEADING), конечные (TRAILING) или те и другие (BOTH) указанные символы из заданной строки. Например, вычисление выражения TRIM (BOTH '*' FROM '**Hello World *') даст результат 'Hello World'
POSITION	Функция возвращает позицию одной строки в пределах другой строки. Например, в результате вычисления выражения POSITION ('ee' IN 'Beech') будет получено значение 2

Оператор	Назначение
SUBSTRING	Функция выполняет выделение подстроки из заданной строки, Например, в результате вычисления выражения SUBSTRING(Beech' FROM 1 TO 3) будет получено значение 'Bee'
CASE	Оператор возвращает одно из значений заданного набора исходя из результатов проверки выполнения указанных условий. Например: CASE type WHEN ‘дом’ THEN 1 WHEN ‘квартира’ THEN 2 ELSE 0 END
CURRENT_DATE	Функция возвращает текущую дату
CURRENT_TIME	Функция возвращает текущее время. Например, выражение CURRENT TIME(6) возвратит текущее время с точностью до микросекунд
EXTRACT	Функция возвращает значение указанного поля из значения типа даты или времени. Например, в результате вычисления выражения EXTRACT(YEAR FROM staff dob) будет извлечено значение года в колонке Dob записи таблицы Staff

7.3. Средства поддержки целостности данных

Поддержка целостности данных включает средства задания ограничений, которые вводятся с целью защиты базы от нарушения согласованности сохраняемых в ней данных. Существует пять типов ограничений поддержки целостности:

- обязательные данные;
- ограничения для доменов;
- целостность сущностей;
- ссылочная целостность;
- требования конкретного предприятия.

7.3.1. Обязательные данные

Для некоторых столбцов требуется наличие в каждой строке таблицы конкретного, отличного от неопределенного значения. Для задания

ограничений подобного типа предусматривается использование спецификатора NOT NULL, указываемого в операторах CREATE TABLE и ALTER TABLE. Если для столбца задан спецификатор NOT NULL, система отвергает любые попытки вставить в такой столбец пустое значение. А если при определении характеристик столбца задан спецификатор NULL, то система допускает размещение в этом столбце значений NULL. По умолчанию применяется спецификатор NULL.

Например, для указания того, что столбец position (Должность) в таблице Staff (Персонал) не может содержать пустых значений, следует определить его, как показано ниже.

```
position VARCHAR(IO) NOT NULL
```

7.3.2. Ограничения для доменов

Каждый столбец имеет собственный домен, т.е. некоторый набор допустимых значений. Например, для определения пола работника достаточно всего двух значений, поэтому домен для столбца sex (Пол) таблицы Staff можно определить как набор из двух строк длиной в один символ со значением либо 'M', либо ' F ' . Существует два механизма определения доменов в операторах CREATE TABLE и ALTER TABLE. Первый состоит в использовании конструкции CHECK, позволяющей задать требуемые ограничения для столбца или таблицы в целом. Конструкция CHECK имеет следующий формат:

```
CHECK (SearchCondition)
```

При определении ограничений для отдельного столбца в конструкции CHECK можно ссылаться только на определяемый столбец. Например, для указания того, что столбец sex может содержать лишь два допустимых значения (' M ' и 'F'), следует объявить его таким образом:

```
sex CHAR NOT NULL CHECK (sex IN ('M' , 'F') )
```

Однако стандарт ISO позволяет определять и более сложные домены, для чего предназначен второй механизм — использование оператора CREATE DOMAIN, имеющего следующий формат:

```
CREATE DOMAIN DomainName [AS] datatype
        [DEFAULT DefaultOption]
        [CHECK (SearchCondition)]
```

Каждому создаваемому домену присваивается имя, задаваемое параметром DomainName, тип данных, определяемый параметром DataType, необязательное значение по умолчанию, устанавливаемое параметром DefaultOption, и необязательный набор допустимых значений, определяемый в конструкции CHECK.

Таким образом, в условиях предыдущего примера мы могли бы определить домен для столбца sex с помощью следующего оператора:

```
CREATE DOMAIN SexType AS CHAR
        DEFAULT 'M'
        CHECK (VALUE IN ( ' M ' , ' F ' ) ) ;
```

Теперь столбец sex в таблице Staff можно будет описать, используя домен SexType вместо определителя типа данных CHAR:

```
sex SexType NOT NULL
```

Значение параметра SearchCondition может предусматривать обращение к справочной таблице. Например, можно создать домен KodB (номер отделения), который позволит вводить в соответствующие столбцы различных таблиц только те значения, которые уже существуют в столбце KodB таблицы B. Для этой цели необходимо использовать следующий оператор:

```
CREATE DOMAIN BranchNumber AS VARCHAR(4)
        CHECK (VALUE IN (SELECT KodB FROM B));
```

Удаление доменов из базы данных выполняется с помощью оператора DROP.

Спецификатор способа удаления домена (RESTRICT или CASCADE) определяет, какие действия выполняются в базе данных, если домен в настоящее время используется. Если задан спецификатор RESTRICT, а домен применяется в существующей таблице, представлении или определении проверки, то операция удаления оканчивается неудачей.

Если задан спецификатор CASCADE, то в любой столбец таблицы, который основан на определении домена, автоматически вносятся изменения таким образом, чтобы в нем применялся базовый тип данных домена. Любые ограничения или применяемые по умолчанию конструкции операторов для этого домена заменяются в случае необходимости ограничениями столбца или применяемой по умолчанию конструкцией оператора для соответствующего столбца.

7.3.3. Целостность сущностей

Первичный ключ таблицы должен иметь уникальное непустое значение в каждой ее строке. Например, каждая строка таблицы P должна содержать уникальное значение номера объекта недвижимости, помещенное в столбец KodP, именно оно будет уникальным образом определять объект недвижимости, представленный этой строкой таблицы. Стандарт позволяет задавать подобные требования поддержки целостности данных с помощью конструкции PRIMARY KEY в операторах CREATE TABLE и ALTER TABLE.

Например, для определения первичного ключа таблицы P можно использовать следующую конструкцию:

```
PRIMARY KEY(KodP)
```

В случае составного первичного ключа, например, первичного ключа таблицы V, состоящего из двух столбцов, конструкция определения первичного ключа PRIMARY KEY будет иметь вид

```
PRIMARY KEY(KodR, KodP)
```

Конструкция PRIMARY KEY может указываться в определении таблицы только один раз. Однако существует возможность гарантировать уникальность значений и для любых альтернативных ключей таблицы, для чего предназначено ключевое слово UNIQUE. Кроме того, при определении столбцов альтернативных ключей рекомендуется использовать и спецификаторы NOT NULL. В каждой таблице может быть определено произвольное количество конструкций UNIQUE. База данных отвергает любые попытки выполнения операций INSERT или UPDATE, которые влекут за собой создание повторяющегося значения в любом потенциальном ключе.

7.3.4. Ссылочная целостность

Внешние ключи представляют собой столбцы или наборы столбцов, предназначенные для связывания каждой из строк дочерней таблицы, содержащей этот внешний ключ, со строкой родительской таблицы, содержащей соответствующее значение потенциального ключа. Понятие ссылочной целостности означает, что если поле внешнего ключа содержит некоторое значение, то оно обязательно должно ссылаться на существующую допустимую строку в родительской таблице. Например, значение в столбце номера отделения KodB таблицы P всегда должно связывать данные об объекте недвижимости с конкретной строкой таблицы B, соответствующей тому отделению компании, за которым закреплен этот объект недвижимости. Если столбец с номером отделения не пуст, он обязательно должен являться допустимым значением столбца KodB таблицы B.

Стандарт предусматривает механизм определения внешних ключей с помощью конструкции FOREIGN KEY.

Например, для определения внешнего ключа KodB в таблице P можно использовать следующую конструкцию:

```
FOREIGN KEY(KodB) REFERENCES B
```

Теперь система отклонит выполнение любых операторов INSERT или UPDATE, с помощью которых будет предпринята попытка создать в дочерней таблице значение внешнего ключа, не соответствующее одному из уже существующих значений потенциального ключа родительской таблицы.

7.4. Создание баз данных

В соответствии со стандартом ISO, таблицы и другие объекты базы данных существуют в некоторой среде (environment). Помимо всего прочего, каждая среда состоит из одного или более каталогов (catalog), а каждый каталог — из набора схем (schema). Схема представляет собой поименованную коллекцию объектов базы данных, которые некоторым образом связаны друг с другом (все объекты в базе данных должны быть описаны в той или иной схеме). Объектами схемы могут быть таблицы, представления и домены. Все объекты схемы имеют одного и того же владельца

Оператор определения схемы имеет следующий формат (упрощенно):

```
CREATE SCHEMA [name | AUTHORIZATION creator-identifier]
```

Таким образом, если создателем схемы под именем Test будет пользователь Ivanov, то данный оператор будет выглядеть следующим образом:

```
CREATE SCHEMA Test AUTHORIZATION Ivanov;
```

Схема удаляется с помощью оператора DROP SCHEMA, который имеет следующий формат:

```
DROP SCHEMA name
```

Для создания БД используется оператор SQL, имеющий формат:

```
CREATE {DATABASE} "<имя_файла>"
    [ USER "имя_пользователя" [PASSWORD "пароль"]]
    [PAGE_SIZE [=] целое]
    [LENTH [=] целое]
    [DEFAULT CHARACTER SET набор символов]
```

"<имя_файла>"	Имя файла, в котором будет храниться содержимое БД
"имя_пользователя" "пароль"	Имя пользователя и пароль, которые проверяются при соединении пользователя с сервером
PAGE_SIZE [=] целое	Размер страницы БД в байтах (**1024**, 2048, 4096, 8192)
LENTH [=] целое	Длина файла в страницах (>50, по умолчанию 75)
DEFAULT CHARACTER SET	Определяет набор символов, применяемый в БД. Например, WIN1251

7.4.1. Создание таблиц (оператор CREATE TABLE)

После создания общей структуры базы данных можно приступить к созданию таблиц, представляющих отношения, входящие в состав проекта базы данных. Для этой цели используется оператор CREATE TABLE, имеющий следующий упрощенный формат:

```
CREATE TABLE ИмяТаблицы (
        <определение_столбца1> [<ограничение1>]
         [<определение_столбца2]| [<ограничение2>]
        [PRIMARY KEY (ListOfColumns)]
        [FOREIGN KEY (ListOfColumns) );
<определение_столбца> = имя_столбца
        {тип_данных | COMPUTED [BY] (<выражение>) | домен}
        [DEFAULT { литерал  |NULL | USER}]  [NOT NULL]
        [CHSCK (<ограничение>)]
```

Здесь:

имя_столбца	допустимое имя столбца
тип_данных	допустимый тип данных
COMPUTED [BY] (<выражение>)	служит для определения столбца вычисляемых значений
DEFAULT	определяет значение, которое по умолчанию присваивается столбцу новой записи
[<ограничение>	ограничения, накладываемые на значения в столбце

В результате выполнения этого оператора будет создана таблица, состоящая из одного или более столбцов с именами, задаваемыми параметрами имя столбца, содержащими данные с типом, указанным параметрами тип. Ключевое слово NULL используется для указания того, что в данном столбце могут содержаться значения NULL. Если указано ключевое слово NOT NULL, то будут отклонены любые попытки поместить значение NULL в данный столбец.

Столбцы первичных ключей всегда должны определяться с указанием ключевого слова NOT NULL. Столбцы внешних ключей также часто (но не всегда) являются кандидатами на использование ключевого слова NOT NULL.

Пример 7.1. Создание таблицы

Для иллюстрации процесса разработки таблиц рассмотрим операторы создания двух таблиц приложения: B и S для учебной БД "Агентство":

```
CREATE TABLE B(
        KodB    VARCHAR(5)    NOT NULL,
        City    VARCHAR(15)   NOT NULL,
        Street  VARCHAR(25)   NOT NULL,
        Tel     CHAR(10)      NOT NULL
        PRIMARY KEY (KodB))
```

```
CREATE TABLE S(
        KodS    VARCHAR(5)    NOT NULL
        Lname   VARCHAR(15)   NOT NULL,
        Fname   VARCHAR(15)   NOT NULL,
        Address VARCHAR(50),
        Pol     CHAR,
        DR      DATETIME
        Dol     VARCHAR (10) NOT NULL,
        ZP      DECIMAL(7,2)  NOT NULL,
        KodB    CHAR(9)       NoT NULL
        PRIMARY KEY (KodS)]
        FOREIGN KEY (KodB) REFERENCES S);
```

7.4.2. Удаление таблиц (оператор DROP TABLE)

Таблицы удаляются из базы данных с помощью оператора DROP TABLE, имеющего следующий формат:

```
DROP TABLE имя_таблицы [RESTRICT | CASCADE]
```

Например, для удаления таблицы P можно использовать следующий оператор:

```
DROP TABLE P;
```

Однако следует отметить, что эта команда удалит не только указанную таблицу, но и все входящие в нее строки данных.

Оператор DROP TABLE дополнительно позволяет указывать, следует ли операцию удаления выполнять каскадно. Если в операторе указано ключевое слово RESTRICT, то при наличии в базе данных хотя бы одного объекта, существование которого зависит от удаляемой таблицы, выполнение данного оператора будет отменено.

Если в операторе указано ключевое слово CASCADE, автоматически будут удалены и все прочие объекты базы данных, существование которых

зависит от удаляемой таблицы, а также другие объекты, существование которых зависит от удаляемых объектов.

7.4.3. Создание индекса (оператор CREATE INDEX)

Индекс представляет собой структуру, позволяющую выполнять ускоренный доступ к строкам таблицы на основе значений одного или более ее столбцов. Наличие индекса может существенно повысить скорость выполнения некоторых запросов. Однако, поскольку индексы должны обновляться системой при каждом внесении изменений в их базовую таблицу, они создают дополнительную нагрузку на систему. Большинство диалектов поддерживает следующий оператор:

```
CREATE [UNIQUE] INDEX <имя_индекса> ON <имя_таблицы>
        (столбцы [ASC | DESC] [,... ])
```

Указанные в операторе столбцы составляют ключ индекса и должны быть перечислены в возрастающем или убывающем порядке.

Если в операторе указано ключевое слово UNIQUE, уникальность значений ключа индекса будет автоматически поддерживаться системой. Рекомендуется создавать уникальные индексы непосредственно при создании таблицы. В результате система возьмет на себя контроль за уникальностью значений данных в соответствующих столбцах.

Для таблиц S и P должны быть созданы, по крайней мере, следующие индексы:

```
CREATE UNIQUE INDEX KodS_ind ON S (KodS);
CREATE UNIQUE INDEX KodP_ind ON P (KodP);
```

Для каждого из ключевых столбцов может быть указан порядок следования значений — по возрастанию (ASC) или по убыванию (DESC), причем значение ASC используется по умолчанию. Например, для таблицы P можно создать следующий индекс:

```
CREATE INDEX Rent_ind ON P (Type, Rooms DESC);
```

7.4.4. Удаление индекса (оператор DROP INDEX)

Если для таблицы базы данных был создан индекс, который впоследствии оказался ненужным, то его можно удалить с помощью оператора DROP INDEX. Этот оператор имеет следующий формат:

```
DROP INDEX имя_индекса
```

Задания 7

Задание 7.1.

Создать базу данных «Продажа товаров», в которой будет 3 таблицы: T – товары, K – клиенты, P – продажи.

T		
Код товара	Товар	Цена
KT	T	P

K		
Код клиента	Фамилия	Город
KK	F	G

P			
Код продажи	Код клиента	Код товара	Количество
KP	KK	KT	K

Задание 7.2.

Создать базу данных «Сессия» с 3 таблицами St – студенты, Dis – дисциплины, Ex – экзамены.

ST		
Код студента	Фамилия	Группа
KS	F	G

DIS	
Код дисциплины	Название
KD	D

EX				
Код экзамена	Код студента	Код дисциплины	Оценка	Дата
KE	KS	KD	Z	DT

Задание 7.3.

Создать базу данных «Библиотека» с 3 таблицами KN – книги, KL – клиенты, VD – выдача книг.

KN		
Код книги	Название	Автор
KKN	N	A

KL	
Код клиента	Фамилия
KK	F

VD				
Код	Код книги	Код клиента	Дата	Срок
K	KKN	KK	DT	SR

Задание 7.4.

Создать базу данных «Аренда помещений» с 3 таблицами: A – арендаторы, P – объекты аренды, S – сведения об аренде.

P		
Код объекта	Тип	Цена за месяц
KP	T	C

A	
Код арендатора	Фамилия
KA	F

S				
Код	Код объекта	Код арендатора	Дата начала	Продолжи-тельность (мес.)
K	KP	KA	DT	SR

Задание 7.5.

Создать базу данных «Продажа авиабилетов» с 3 таблицами: **R** – рейсы, **P** – пассажиры, **F** – полеты.

Имена полей заданы, первичные ключи подчеркнуты.

R		
Код рейса	Аэропорт (назначения)	Цена
KR	NR	PR

P	
Код пассажира	Фамилия
KP	FIO

F			
Код	Код рейса	Код клиента	Дата
KF	KR	KP	DV

Тема 8. КУРСОВАЯ РАБОТА

Для индивидуального задания разработать базу данных (СУБД MS Access, InterBase, MS SQL Server) и приложение базы данных (Delphi). База данных должна состоять не менее чем из 5 таблиц.

Уделить особое внимание:

- описанию предметной области: оно должно быть достаточным для моделирования БД методами нормализации и «сущность-связь»
- моделированию данных с помощью процесса нормализации (от ненормализованных данных до НФБК). Описать шаги нормализации, на каждом шаге выявить все потенциальные и первичный ключи, проанализировать все функциональные зависимости;
- моделированию данных с помощью ER-диаграмм.

Приложение должно обеспечивать следующие функции:

- иметь удобный интерфейс и предусматривать возможность ввода, редактирования и удаления всех данных в БД с помощью форм. При вводе информации максимально использовать возможность выбора данных из связанных таблиц (поле со списком);
- возможность вывода в формах одновременно данных из нескольких связанных таблиц (формы с подчиненной) и обобщенной информации по хранящимся в БД данным (не менее 3);
- реализация ограничений целостности и ограничений, накладываемых постановкой задачи;
- осуществлять удобные режимы просмотра информации с помощью функций поиска, сортировки и фильтрации.

Например, для БД «Учет продаж» вывод форм:

- форма накладные клиента: предоставляет возможность просмотра для каждого клиента его накладных с указанием общей суммы каждой накладной (общего количества купленных товаров?) и подведением общего итога;

- список клиентов с указанием общего количества купленных товаров и общей суммы;
- список проданных товаров с информацией о количестве и средней цене;
- список продавцов с информацией об общем объеме продаж;
- список клиентов, совершивших более 10 покупок с информацией об их общей сумме.

Оформить отчет о выполненной работе (файл MS Word). Отчет (поля слева 2,5 см, остальные – 1,5 см, шрифт Times…12, межстрочный интервал - 1) должен быть не менее 10 станиц. Курсовая работа должна иметь титульный лист, содержание и включать разделы:

1. Описание задачи. В разделе должно быть представлено описание предметной области, бизнес-правил, корпоративных ограничений, форм первичных документов и функций приложения;
2. Концептуальная модель БД. В разделе приводится построение модели БД посредством нормализация отношений от ненормализованных данных до нормальной формы Бойса-Кодда;
3. Модель «сущность-связь». В разделе описывается моделирование БД посредством построения ER-диаграмм. Должно быть приведено итерационное построение модели, включающее не менее 3 итераций (этапов);
4. Описание БД (таблиц, столбцов, связей, доменов);
5. Описание приложения БД (форм и отчетов). Инструкция пользователю.
6. Описание операторов создания БД (скрипты).

Описание задачи включает:

Описание предметной области. Например, для темы «Учет продаж товаров»:

Программа «Учет продаж» предназначена для учета продаж товаров. Учет продаж товаров осуществляется на основании накладной:

Накладная №		157					
Дата		10/01/03					
Покупатель							
Код		345					
Наименование		ОАО «Сфера»					
ИНН		072 001 000 6789					
Адрес		г. Тюмень, ул. Ленина, 120					
№	Товар		Количество	Ед. изм.		Цена	Стоимость
	наименование	код		код	нам.		
1	Сахар	0012	100	12	кг	15.00	1 500
2	Макароны	0015	200	23	пачка	13.00	2 600
3	Тушенка	0022	300	24	банка	25.00	7 500
	Всего		600				**11 600**
Кладовщик							**Петрова**

Накладная выписывается на конкретного клиента. Номер накладной – уникален. В каждой накладной строки нумеруются последовательно 1, 2, …. Каждая накладная содержит, по крайней мере, одну строку. Цена отпускаемых товаров и единицы измерения определяется кодом товара.

Цена товара может меняться, при этом в накладных, оформленных ранее, цена на товар изменяться не должна. И т.д.

Функции приложения. Описание процессов, которые должны быть автоматизирована приложением. Например:

Программа «Учет» выполняет следующие функции:

- *предоставляет возможность вводить, удалять и изменять информацию о товарах, клиентах, кладовщиках;*

- *при формировании новой накладной приложение предоставляет возможность вводить информацию о … путем выбора значений из связанных таблиц;*

- *позволяет заносить в базу данных информацию о продажах товаров;*

- *предоставляет возможность получения обобщенной информации о продажах товаров, о покупках клиентов, о товарах, отпущенных кладовщиком;*

- *и т.д.*

ТЕМЫ КУРСОВЫХ РАБОТ

1. Автоматизация работы отдела кадров организации.
2. Документооборот и исполнение поручений.
3. «Абитуриент» - автоматизация учета поступления абитуриентов.
4. Учет успеваемости студентов.
5. Учет нагрузки преподавателей.
6. Автоматизация учета выдачи книг в библиотеке.
7. Автоматизация учета экспонатов в картинной галерее.
8. Автоматизация учета записей в фонотеке.
9. Автоматизация учета работы студентов в компьютерных кабинетах.
10. Автоматизация учета расхода материалов, используемых при выпуске продукции.
11. Автоматизация учета продаж и гарантийного обслуживания автомобилей.
12. Построить базу данных, обслуживающую ведение заказов авторемонтной мастерской. Информация должна содержать сведения о клиенте(ФИО, адрес), тип работы, оплату и информацию об исполнителе (ФИО, квалификация).
13. Продажа компьютеров (комплектация).
14. Транспортное агентство (оказание транспортных услуг)
15. Учет работы транспортных средств.
16. Рекламное агентство (учет заказов).
17. Туристическое агентство.
18. Учет вычислительной техники и оргтехники организации.
19. Гарантийное обслуживание и ремонт видеооборудования.
20. Автоматизация документооборота и учета деятельности юридической фирмы.
21. Автоматизация учета работы пользователей в сети Интернет.
22. Аренда автотранспорта.

23. Автоматизация учета безработных в департаменте занятости населения.
24. Организация учета в ГИБДД (нарушения и оплата штрафов).
25. БД расписаний занятий в университете (кафедры - группы – дисциплины – аудитории – преподаватели - расписание занятий)
26. Агентство сдачи в аренду объектов недвижимости.
27. Учет оплаты коммунальных услуг.
28. БД расписаний маршрутов движения транспорта (станции - маршруты - станции маршрутов - время прибытия/убытия).
29. БД студенческого общежития (корпус – комнаты – проживающие – оплаты за проживание)
30. Учет больных в поликлинике. Построить базу данных, описывающую обращение больных в поликлинику. Информация должна содержать сведения о больных (ФИО, адрес, дату рождения), враче (ФИО, специальность), дате осмотра и заключение врача.
31. БД тем курсовых проектов (кафедра – группа – студент – дисциплина – семестр – тема курсового проекта - оценка)
32. БД рецептов (раздел – подраздел – тип рецепта - рецепта)
33. Учет банковских операций.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ

ГОУ ВПО ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ

КУРСОВАЯ РАБОТА

ДИСЦИПЛИНЕ «БАЗЫ ДАННЫХ»

АВТОМАТИЗАЦИЯ УЧЕТА УСПЕВАЕМОСТИ СТУДЕНТОВ

Выполнил:
студент 394 гр. Степанов С. С.

Руководитель:
канд. физ.-мат. наук,
доцент Моор П. К.

Тюмень - 2004

СПИСОК ЛИТЕРАТУРЫ

1. Базы данных: модели, разработка, реализация. /Т.С. Карпова. – СПб.: Питер, 2001. – 304 с.
2. Базы данных: интеллектуальная обработка информации / Корнеев В.В., Гареев А.Ф., Васюткин С.В. и др. – М.: Издательство Молчанова, 2001. – 496 с.
3. Конноли Т., Бегг К., Страчан А. Базы данных: проектирование, реализация, сопровождение. Пер. с англ. М.: издательский дом «Вильямс», 2000, 1120 с.
4. Фаронов В.В., Шумаков П.В. Delphi. Руководство разработчика баз данных. М.: «Нолидж», 2001. 640 с.
5. Тихомиров Ю.В. Microsoft SQL Server 7.0. СПб.: БХВ-Санкт-Петербург, 2000, 720 с.
6. Гончаров А. Access 97 в примерах. СПб.:Питер, 1997.
7. Бобровски С. Oracle7 и вычисления клиент/сервер. М.: Лори,1995.
8. Мейер Д. Теория реляционных баз данных. М.: Мир,1987
9. Дейт К. Дж. Введение в системы баз данных. 6-е издание. К., М.; СПб.: Издательский дом «Вильямс», 2000. 848 с.

ПРИЛОЖЕНИЕ.

ПРИМЕР УЧЕБНОЙ БАЗЫ ДАННЫХ «АГЕНТСТВО»

Агентство занимается сдачей в аренду объектов недвижимости по поручению их владельцев: учет сдаваемых объектов, их осмотр арендаторами, оформление договоров аренды и инспекция сдаваемых в аренду объектов недвижимости. Агентство имеет ряд отделений в городах. Необходимо разработать базу данных, обеспечивающих информационную поддержку деятельности агентства.

Сведения об отделениях приведены в таблице:

B - Отделения (Branch)

KodB	City	Street	Tel
код отделения	город	улица, дом	телефон
B5	Тюмень	Береговая, 22	46-25-25
B7	Нижневартовск	Новая, 15	44-35-35
B3	Тюмень	Северная, 34	36-23-52
B4	Ишим	Окружная, 44	3-45-65
B2	Надым	Садовая 74	7-45-33

В каждом отделении агентства работает ряд сотрудников:

S - Сотрудники (Staff)

KodS	Lname	Fname	Pol	DR	Dol	ZP	KodB
код	фамилия	имя	пол	дата р.	должность	Оклад	код отд.
S21	Иванов	Иван	М	01.02.70	Руководитель	500	B5
S37	Петрова	Татьяна	Ж	12.11.72	Агент	250	B3
S14	Сидоров	Степан	М	06.05.68	Менеджер	400	B3
S09	Зотова	Ирина	Ж	11.03.75	Агент	240	B7
S05	Степанова	Елена	Ж	16.07.71	Секретарь	200	B3
S41	Петров	Игорь	М	02.02.68	Менеджер	350	B5

Информация о сдаваемой недвижимости приведена в таблице:

P - Характеристика Property_Rent

KodP	City	Street	Type	Rooms	Rent	KodO	KodS
код	город	улица, дом	тип	число помещ.	стоимость	код влад.	код сотр.
P14	Тюмень	Садовая, 23	дом	6	650	C46	S09
P94	Нижневартовск	…	кварт.	4	400	C87	S41
P04	Тюмень		дом	4	350	C40	S14
P36	Ишим		кварт.	2	375	C93	S37
P21	Надым		кварт.	3	600	C87	S37
P16	Сургут		кварт.	2	450	C93	S14

Информация об арендаторах приведена в таблице:

R - Арендатор (Renter)

KodR	Name	Address	Type	MaxRent	KodB
код	фамилия, имя	адрес	тип	стоимость	код отдел
R76	Саблев Иван	Тюмень Полевая, 23	кварт.	425	B7
R56	Рубин Степан		кварт.	350	B5
R74	Кротова Ирина		дом	750	B3
R62	Зимина Елена		кварт.	600	B3

Информация о владельцах приведена в таблице:

O - Владелец (Owner)

KodO	Name	Address	Tel
код владельца	фамилия, имя	адрес	телефон
C46	Рублев Петр	Тюмень Полевая, 23	46-19-34
C87	Силин Роман		46-17-03
C40	Бажова Светлана		22-33-44
C93	Кубасов Игорь		3-45-45

Информация об осмотрах недвижимости приведена в таблице:

V - Осмотр (Viewing)

KodR	KodP	Date	Comment
код арендатора	код недвижимости	дата	Заключение
R56	P14	24.05.01	Мала
R76	P04	20.04.01	Подходит
R56	P04	26.05.01	
R62	P14	14.05.01	Дорого
R56	P36	28.04.01	