МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

«БРАТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра управления в технических системах

УТВЕРЖДАЮ:
Проректор по учебной работе
_____ Е.И. Луковникова
17 мая 2019 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Б1.В.16

НАПРАВЛЕНИЕ ПОДГОТОВКИ

27.03.04 Управление в технических системах

ПРОФИЛЬ ПОДГОТОВКИ

Управление и информатика в технических системах

Программа академического бакалавриата

Квалификация (степень) выпускника: бакалавр

Программа составлена в соответствии с федеральным государственным образовательным стандартом высшего образования по направлению подготовки 27.03.04 Управление в технических системах от 20.10.2015 г № 1171 и учебным планом ФГБОУ ВО «БрГУ» от 01.04.2019 г № 196 для заочной формы обучения набора 2019 года

СОДЕРЖАНИЕ ПРОГРАММЫ **Стр.**

1. ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫХ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Вид деятельности выпускника

Дисциплина охватывает круг вопросов, относящихся к научно-исследовательским и проектно-конструкторским видам профессиональной деятельности выпускника в соответствии с компетенциями и видами деятельности, указанными в учебном плане.

Цель дисциплины

Изучение теории баз данных. Формирование практических навыков проектирования информационных систем на основе баз данных. Формирование практических навыков создания реляционных баз данных в современных СУБД. Формирование практических навыков по использованию языка запросов SQL. Формирование практических навыков работы с инструментальными средствами быстрой разработки приложений.

Задачи дисциплины

Подготовка студента к самостоятельному решению теоретических и прикладных задач связанных с проектированием информационных систем на основе баз данных.

.

Код компетенции	Содержание компетенций	Перечень планируемых результатов обучения по дисциплине
ОПК-6	Способность осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий	**Знать:** - Оценивать производительность вычислительных машин и систем, выбирать вычислительные средства для проектирования устройств и систем управления; **Уметь:** - Применять принципы и методы построения моделей, методы анализа, синтеза и оптимизации при создании и исследования средств и систем управления; **Владеть:** - Навыками работы с современными аппаратными и программными средствами исследования и проектирования систем управления.
ОПК-9	Способность использовать навыки работы с компьютером, владеть методами информационных технологий, соблюдать основные требования информационной безопасности	**Знать:** - Базовое устройство персонального компьютера. **Уметь:** - Использовать персональный компьютер для самостоятельной работы. **Владеть:** - Достаточным уровнем использования универсальных пакетов прикладных компьютерных программ.
ПК-5	Способность осуществлять сбор и анализ исходных данных для расчета и проектирования систем и средств автоматизации и управления	**Знать:** - Основные информационные процессы происходящие в персональном компьютере. **Уметь:** - Использовать сбор и анализ исходный данных для расчета и проектирования систем управления базами данных. **Владеть:** - Навыками работы с современными аппаратными и программными средствами сбора и анализа информации.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Дисциплина Б1.В.16 Системы управления базами данных относится к вариативной части.

Дисциплина системы управления базами данных базируется на знаниях, полученных при изучении дисциплин Б1.Б.12 Информационные технологии, Б1.В.7 Информатика, Б1.В.15 Структуры и алгоритмы обработки данных.

Основываясь на изучении перечисленных дисциплин, системы управления базами данных представляет основу для изучения дисциплин: Б1.Б.18 Моделирование систем управления и Б1.В.14 Автоматизированные информационно-управляющие системы.

Такое системное междисциплинарное изучение направлено на достижение требуемого ФГОС уровня подготовки по квалификации бакалавр.

3. РАСПРЕДЕЛЕНИЕ ОБЪЕМА ДИСЦИПЛИНЫ

3.1. Распределение объема дисциплины по формам обучения

Форма обучения	*Курс*	*Семестр*	*Трудоемкость дисциплины в часах*						*Контрольная работа*	*Вид промежуточной аттестации*
			Всего часов (с экз.)	*Аудиторных часов*	*Лекции*	*Лабораторные работы*	*Практические занятия*	*Самостоятельная работа*		
1	2	3	4	5	6	7	8	9	10	11
Очная	2,3	4,5	216	105	35	53	17	111		Зачет Экзамен
Заочная	3	-	216	17	6	6	5	199		Экзамен
Заочная (ускоренное обучение)	2	2	216	13	4	4	3	193		Экзамен
Очно-заочная	-	-	-	-	-	-	-	-	-	-

3.2. Распределение объема дисциплины по видам учебных занятий и трудоемкости:

Вид учебных занятий	*Трудоемкость (час.)*	*в т.ч. в интерактивной, активной, инновационной формах, (час.)*	*Распределение по семестрам, час*	
			4	*5*
I. Контактная работа обучающихся с преподавателем (всего)	105	18	54	51
Лекции (Лк)	35	8	18	17
Лабораторные работы (ЛР)	53	6	36	17
Практические работы (ПР)	17	4	-	17
Индивидуальные (групповые) консультации	+	-	+	+
II.Самостоятельная работа обучающихся	75	-	54	21

(СР)				
Подготовка к лабораторным работам	35	-	30	5
Подготовка к практическим работам	10			10
Подготовка к зачету	24	-	24	-
Подготовка к экзамену в течение семестра	6	-	-	6
Выполнение контрольной работы	0	-	-	0
III. Промежуточная аттестация зачет **,**экзамен	+ 36	-	+	36
Общая трудоемкость дисциплины ……. час.	216	-	108	108
зач. ед**.**	5	-	2	3

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

4.1. Распределение разделов дисциплины по видам учебных занятий - для очной формы обучения:

№ раз-дела и темы	*Наименование раздела и тема дисциплины*	*Трудоем-кость, (час.)*	*Виды учебных занятий, включая самостоятельную работу обучающихся и трудоемкость; (час.)*			
			учебные занятия			*самостоя тельная работа обучаю-щихся*
			лекции	*лабораторные работы*	*практиче ские работы*	
1	2	3	4	5	6	7
1.	**Введение в базы данных**	**18**	**6**	**-**	**-**	**12**
1.1.	Основные понятия и определения	5	2	-	-	3
1.2.	Современное состояние технологий баз данных.	5	2	-	-	3
1.3.	Базы данных.	4	1	-	-	3
1.4.	Системы управления базами данных.	4	1	-	-	3
2.	**Архитектура СУБД**	**18**	**6**	**-**	**-**	**12**
2.1.	Трехуровневая архитектура базы данных	5	2	-	-	3
2.2.	Функции СУБД.	5	2	-	-	3
2.3.	Языки СУБД.	4	1	-	-	3
2.4.	Архитектура многопользовательских СУБД.	4	1	-	-	3
3.	**Модели данных**	**34**	**6**	**10**	**6**	**12**
3.1.	Классификация моделей данных.	11	2	3	2	4
3.2.	Сетевая модель.	11	2	3	2	4
3.3.	Иерархическая модель данных.	12	2	4	2	4
4.	**Реляционная модель данных**	**34**	**6**	**15**	**-**	**13**
4.1.	История вопроса.	9	2	4	-	3

4.2.	Структура часть реляционной модели.	9	2	4	-	3
4.3.	Обновление отношений.	9	1	4	-	4
4.4.	Целость базы данных.	7	1	3	-	3
5.	**Проектирование базы данных.**	**39**	**6**	**14**	**6**	**13**
5.1.	Избыточность данных и аномалии обновления в БД.	12	2	4	2	4
5.2.	Нормализация отношений.	13	2	5	2	4
5.3.	Проектирование реляционной базы данных.	14	2	5	2	5
6.	**Язык SQL**	**37**	**5**	**14**	**5**	**13**
6.1.	Оператор выбора SELECT. Формирование запросов к базе.	12	2	4	2	4
6.2.	Операторы манипулирования данными.	13	2	5	2	4
6.3.	Операторы определения данных.	12	1	5	1	5
	ИТОГО	**180**	**35**	**53**	**17**	**75**

- для заочной формы обучения:

№ раз-дела и темы	***Наименование раздела и тема дисциплины***	***Трудоем-кость, (час.)***	***Виды учебных занятий, включая самостоятельную работу обучающихся и трудоемкость; (час.)***			
			учебные занятия			***самостоя тельная работа обучаю-щихся***
			лекции	***лабораторные работы***	***практиче ские работы***	
1	**2**	**3**	**4**	**5**	**6**	**7**
1.	**Введение в базы данных**	**32**	**1**	**-**	**-**	**31**
1.1.	Основные понятия и определения	7,5	0,5	-	-	7
1.2.	Современное состояние технологий баз данных.	8	-	-	-	8
1.3.	Базы данных.	8,5	0,5	-	-	8
1.4.	Системы управления базами данных.	8		-	-	8
2.	**Архитектура СУБД**	**32**	**1**	**-**	**-**	**31**
2.1.	Трехуровневая архитектура базы данных	7,5	0,5	-	-	7
2.2.	Функции СУБД.	8,5	0,5	-	-	8
2.3.	Языки СУБД.	8	-	-	-	8
2.4.	Архитектура многопользовательских СУБД.	8	-	-	-	8
3.	**Модели данных**	**32**	**1**	**-**	**-**	**31**
3.1.	Классификация моделей данных.	10,33	0,33	-	-	10
3.2.	Сетевая модель.	10,33	0,33	-	-	10
3.3.	Иерархическая модель данных.	11,34	0,34	-	-	11

4.	**Реляционная модель данных**	**40**	**1**	**6**	**-**	**33**
4.1.	История вопроса.	9,5	0,5	2	-	7
4.2.	Структура часть реляционной модели.	10,5	0,5	2	-	8
4.3.	Обновление отношений.	9	-	1	-	8
4.4.	Целость базы данных.	9	-	1	-	8
5.	**Проектирование базы данных.**	**39**	**1**	**-**	**5**	**33**
5.1.	Избыточность данных и аномалии обновления в БД.	12,5	0,5	-	1	11
5.2.	Нормализация отношений.	13,5	0,5	-	2	11
5.3.	Проектирование реляционной базы данных.	13	-	-	2	11
6.	**Язык SQL**	**32**	**1**	**-**	**-**	**31**
6.1.	Оператор выбора SELECT. Формирование запросов к базе.	11	1	-	-	10
6.2.	Операторы манипулирования данными.	10	-	-	-	10
6.3.	Операторы определения данных.	11	-	-	-	11
	ИТОГО	**207**	**6**	**6**	**5**	**190**

- для заочной формы обучения (ускоренное обучение)**:**

№ раз-дела и темы	***Наименование раздела и тема дисциплины***	***Трудоем-кость, (час.)***	***Виды учебных занятий, включая самостоятельную работу обучающихся и трудоемкость; (час.)***			
			учебные занятия			***самостоя тельная работа обучаю-щихся***
			лекции	***лабораторные работы***	***практиче ские работы***	
1	**2**	**3**	**4**	**5**	**6**	**7**
1.	**Введение в базы данных**	**21**	**1**	**-**	**-**	**20**
1.1.	Основные понятия и определения	5,5	0,5	-	-	5
1.2.	Современное состояние технологий баз данных.	5	-	-	-	5
1.3.	Базы данных.	5,5	0,5	-	-	5
1.4.	Системы управления базами данных.	5		-	-	5
2.	**Архитектура СУБД**	**20**	**-**	**-**	**-**	**20**
2.1.	Трехуровневая архитектура базы данных	5	-	-	-	5
2.2.	Функции СУБД.	5	-	-	-	5
2.3.	Языки СУБД.	5	-	-	-	5
2.4.	Архитектура многопользовательских СУБД.	5	-	-	-	5
3.	**Модели данных**	**21**	**1**	**-**	**-**	**20**
3.1.	Классификация моделей данных.	6,33	0,33	-	-	6
3.2.	Сетевая модель.	7,33	0,33	-	-	7

3.3.	Иерархическая модель данных.	7,34	0,34	-	-	7
4.	**Реляционная модель данных**	**27**	**1**	**4**	**-**	**22**
4.1.	История вопроса.	7	0,5	1	-	5,5
4.2.	Структура часть реляционной модели.	7	0,5	1	-	5,5
4.3.	Обновление отношений.	6,5	-	1	-	5,5
4.4.	Целость базы данных.	6,5	-	1	-	5,5
5.	**Проектирование базы данных.**	**26**	**1**	**-**	**3**	**22**
5.1.	Избыточность данных и аномалии обновления в БД.	8,5	0,5	-	1	7
5.2.	Нормализация отношений.	8,5	0,5	-	1	7
5.3.	Проектирование реляционной базы данных.	9	-	-	1	8
6.	**Язык SQL**	**20**	**-**	**-**	**-**	**20**
6.1.	Оператор выбора SELECT. Формирование запросов к базе.	8	-	-	-	8
6.2.	Операторы манипулирования данными.	8	-	-	-	8
6.3.	Операторы определения данных.	9	-	-	-	9
	ИТОГО	**135**	**4**	**4**	**3**	**124**

4.2. Содержание дисциплины, структурированное по разделам и темам

1. ВВЕДЕНИЕ В БАЗЫ ДАННЫХ

1.1. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Стержневые идеи современных информационных технологий базируются на концепции *баз данных.*

Согласно этой концепции, основой информационных технологий являются *данные,* которые должны быть организованы в базы данных в целях адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей.

Одним из важнейших понятий в теории баз данных является понятие *информации.* Под *информацией* понимаются любые сведения о каком-либо событии, процессе, объекте.

Данные — это информация, представленная в определенном виде, позволяющем автоматизировать ее сбор, хранение и дальнейшую обработку человеком или информационным средством. Для компьютерных технологий данные — это информация в дискретном, фиксированном виде, удобная для хранения, обработки на ЭВМ, а также для передачи по каналам связи.

База данных (БД) — именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области, или иначе БД — это совокупность взаимосвязанных данных при такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений в определенной предметной области. БД состоит из множества связанных файлов.

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

Автоматизированная информационная система (АИС) — это система, реализующая автоматизированный сбор, обработку, манипулирование данными, функционирующая на основе ЭВМ и других технических средств и включающая соответствующее программное обеспечение (ПО) и персонал. В дальнейшем в этом качестве будет использоваться термин *информационная система* (ИС), который подразумевает понятие автоматизированная.

Каждая ИС в зависимости от ее назначения имеет дело с той или иной частью реального мира, которую принято называть *предметной областью* (ПрО) *системы.* Выявление ПрО — это необходимый начальный этап разработки любой ИС. Именно на этом этапе определяются информационные потребности всей совокупности пользователей будущей системы, которые, в свою очередь, предопределяют содержание ее базы данных.

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

Под *задачами обработки данных* обычно понимается специальный класс решаемых на ЭВМ задач, связанных с видом, хранением, сортировкой, отбором по заданному условию и группировкой записей однородной структуры.

Отдельные программы или комплекс программ, реализующие автоматизацию решения прикладных задач обработки данных, называются *приложениями.* Приложения, созданные средствами СУБД, относят к *приложениям СУБД.* Приложения, созданные вне среды СУБД с помощью систем программирования, использующих средства доступа к БД, к примеру, Delphi или Visual Studio, называют *внешними приложениями.*

1.2. СОВРЕМЕННОЕ СОСТОЯНИЕ ТЕХНОЛОГИЙ БАЗ ДАННЫХ

Кратко сформулируем основные современные принципы организации баз данных.

 Значительная часть современных СУБД способна работать на компьютерах различной архитектуры под управлением разных операционных систем.

 Подавляющее большинство современных СУБД обеспечивают поддержку полной реляционной модели данных, обеспечивая целостность категорий и целостность на уровне ссылок.

 Современные СУБД для определения данных и манипуляции ими опираются на принятые стандарты в области языков, а при обмене данными между различными СУБД базируются на существующих технологиях по обмену информацией.
 Многие существующие СУБД относятся к так называемым сетевым СУБД, которые предназначены для поддержки многопользовательского режима работы с базой данных и поддержки возможности децентрализованного хранения данных.
 Такие СУБД имеют развитые средства администрирования баз данных и средства защиты хранимой в них информации.
 Подобные СУБД имеют средства подключения клиентских приложений.
 Современные СУБД характеризуются опытами применения концепции фундаментальной идеи объектно-ориентированного подхода, способствующей повышению уровня абстракции баз данных, являющейся перспективным этапом на пути развития технологий баз данных.
Информационные системы, созданные средствами технологии баз данных, иногда принято называть *банками данных (БнД).*
БнД включает в себя:
 технические средства;
 одну или несколько БД;
 СУБД;
 словарь или каталог данных;
 администратора;
 вычислительную систему;
 обслуживающий персонал.
Схематично это выглядит так, как показано на рис. 1.1.

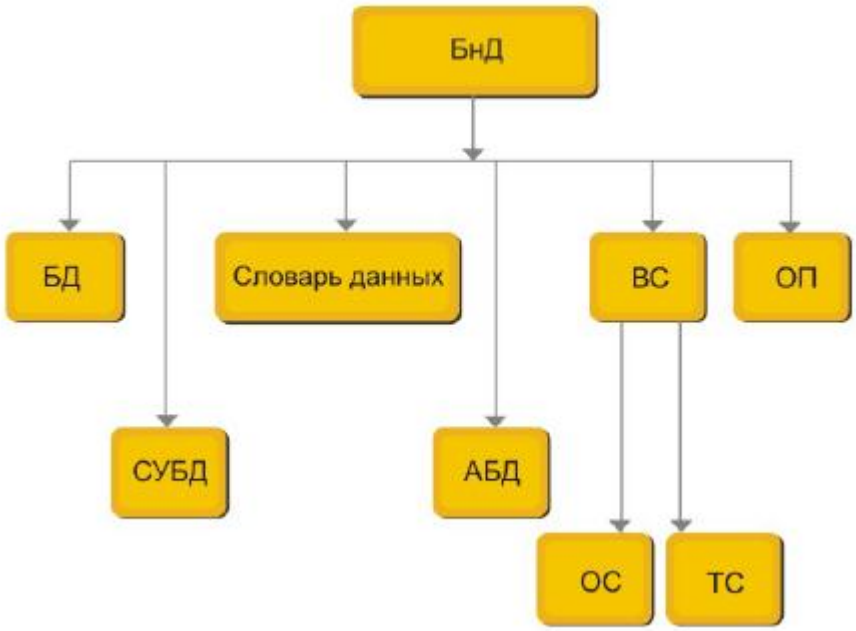

Рис. 1.1. Банк данных
Дадим краткие определения новым составляющим этой схемы.
Словарь или каталог данных служит для централизованного накопления и описания ресурса данных. Он содержит описание ПрО, сведения о структуре БД, о связях между элементами БД. Словарь данных можно рассматривать как часть самой базы данных.
Администратор БД (АБД) — человек или группа лиц, которые принимают решения. Основные функции АБД:
 участие в разработке БД;
 контроль правильности функционирования БД.
Вычислительная система (ВС) — включает программные (ПС) и аппаратные средства (ТС).
Обслуживающий персонал (ОП) — это лица, прямыми обязанностями которых является создание и поддержание корректного функционирования банка данных. Они ответственны за

работу БнД и прикладного программного обеспечения. К обслуживающему персоналу относятся: разработчики и администраторы базы данных, аналитики, программисты.

1.3. БАЗЫ ДАННЫХ

БД, как правило, создается как общий ресурс всего предприятия, где данные являются *интегрированными и общими.* Под понятием *интегрированные* данные подразумевается возможность представить базу данных как объединение нескольких отдельных файлов данных. Под понятием *общие* данные подразумевается возможность использования отдельных областей данных в БД несколькими различными пользователями для разных целей.

В базе данных информация должна быть организована так, чтобы обеспечить минимальную долю ее избыточности. Частичная избыточность информации необходима, но она должна быть минимизирована, так как чрезмерная избыточность данных влечет за собой ряд негативных последствий. Главные из них:

 увеличение объема информации, а значит, потребность в дополнительных ресурсах для хранения и обработки дополнительных объемов данных;

 появление ошибок при вводе дублирующей информации, нарушающих целостность базы данных и создающих противоречивые данные.

БД содержит не только данные, всесторонне характеризующие деятельность самой организации, фирмы, процесса или другой предметной области, но и описания этих данных. Информацию о данных принято называть *"метаданными",* т. е. "данными о данных". В совокупности описания всех данных образуют *словарь данных.*

В БД должны храниться данные, логически связанные между собой. Для того чтобы данные можно было связать между собой, и связать так, чтобы эти связи соответствовали реально существующим в данной предметной области, последнюю подвергают детальному анализу, выделяя сущности или объекты. Сущность или объект — это то, о чем необходимо хранить информацию. Сущности имеют некоторые характеристики, называемые атрибутами, которые тоже необходимо сохранять в БД. Атрибуты по своей внутренней структуре могут быть простыми, а могут быть сложными. Простые атрибуты могут быть представлены простыми типами данных. Различного рода графические изображения, являющиеся атрибутами сущностей, — это пример сложного атрибута. Определив сущности и их атрибуты, необходимо перейти к выявлению связей, которые могут существовать между некоторыми сущностями. Связь — это то, что объединяет две или более сущностей. Связи между сущностями также являются частью данных, и они также должны храниться в базе данных.

Если все это: сущности, атрибуты сущностей и связи между сущностями определено, то схема базы данных может выглядеть примерно так, как представлено на рис. 1.2. На нем показан пример схемы базы данных, которую можно назвать ФАКУЛЬТЕТ.

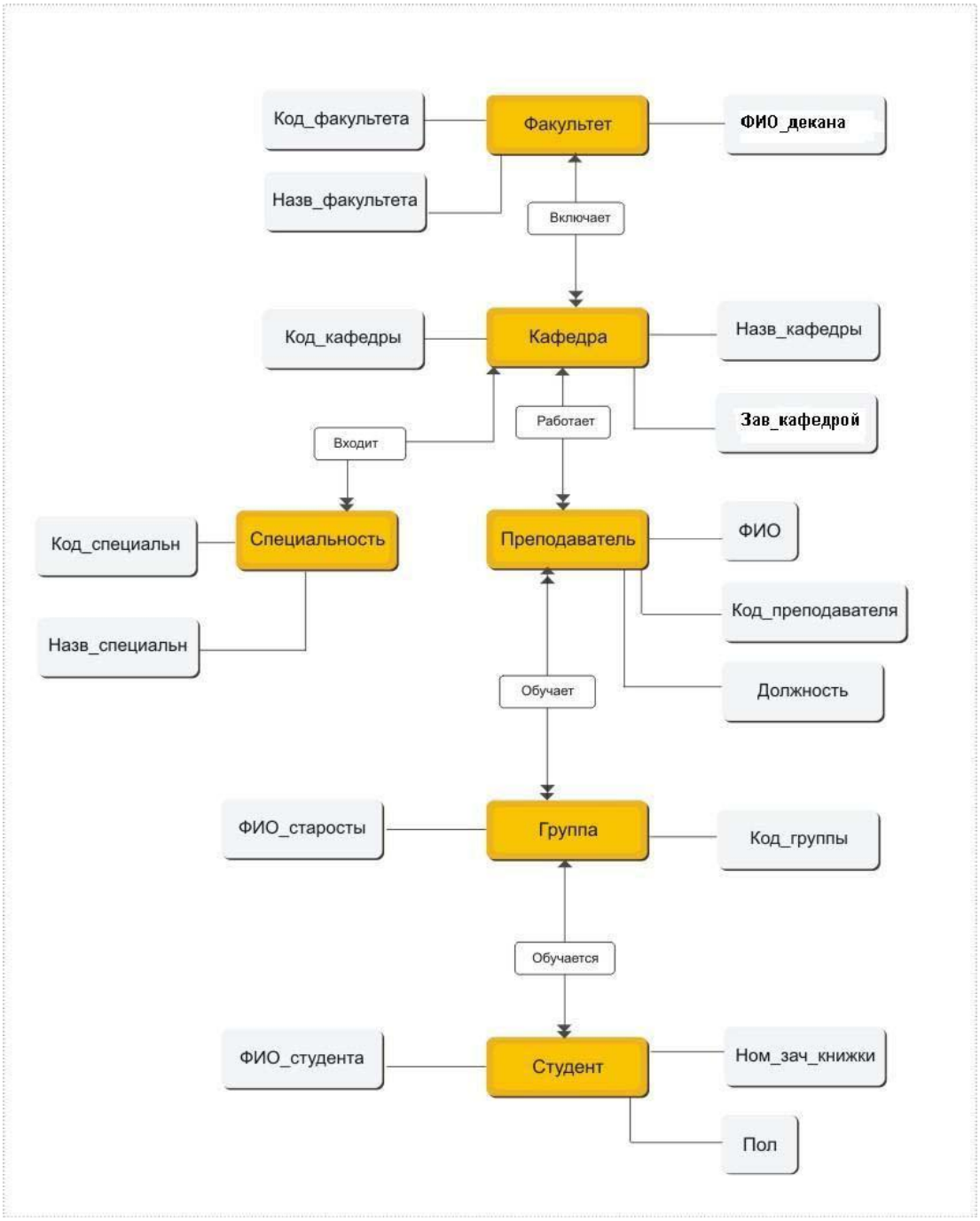

Рис. 1.2. Пример ER-диаграммы базы данных ФАКУЛЬТЕТ

Схема, которая называется ER-диаграммой (Entity-Relationship), состоит из следующих компонентов:

 шести сущностей, которые изображены прямоугольниками, каждый из которых имеет свои атрибуты, помещенные в овалы, а в нижеприведенном списке они перечислены в скобках рядом с именем сущностей:

ФАКУЛЬТЕТ: (Код_факультета, Назв_факультета, ФИО_декана);
КАФЕДРА: (Код_кафедры, Назв_кафедры, Зав_кафедрой);
СПЕЦИАЛЬНОСТЬ: (Код_специальности, Назв_специальности);
ПРЕПОДАВАТЕЛЬ: (Код_преподавателя, ФИО, Должность);
ГРУППА: (Код_группы, ФИО_старосты);
СТУДЕНТ: (Ном_зач_книжки, ФИО, Пол);

 пяти связей, которые обозначены стрелками и связывают те сущности, на которые они направлены:

связь **ВКЛЮЧАЕТ** показывает, что на факультете несколько кафедр;

связь **ВХОДИТ** изображает, что одна и та же кафедра готовит специалистов по нескольким специальностям;

связь **РАБОТАЕТ** определяет то, что на кафедре работает ряд преподавателей;
связь **ОБУЧАЕТ** с двойными стрелками в обоих направлениях поясняет тот факт, что один и тот же преподаватель преподает в разных группах, а одна и та же группа занимается с разными преподавателями;
связь **ОБУЧАЕТСЯ** определяет, что каждая группа включает в себя ряд студентов.
Из представленной диаграммы понятно, что данные обладают определенной структурой. Для выявления этой структуры база данных должна пройти процесс проектирования.
Проектируемая БД должна обладать определенными свойствами. Назовем основные свойства БД.
Целостность. В каждый момент времени существования БД сведения, содержащиеся в ней, должны быть непротиворечивы. Целостность БД достигается вследствие введения ограничений целостности, в частности, к ним относятся ограничения, связанные с нормализацией БД.
Восстанавливаемость. Данное свойство предполагает возможность восстановления БД после сбоя системы или отдельных видов порчи системы.
Безопасность. Безопасность БД предполагает защиту данных от преднамеренного и непреднамеренного доступа, модификации или разрушения. Применяется запрещение несанкционированного доступа, защита от копирования и криптографическая защита.
Эффективность. Свойство эффективности обычно понимается как:
 минимальное время реакции на запрос пользователя;
 минимальные потребности в памяти;
 сочетание этих параметров.

1.4. СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Итак, как уже не один раз упоминалось, СУБД — это программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать базу данных, а также осуществлять к ней контролируемый доступ.
По степени универсальности различаются два класса СУБД — системы общего назначения и специализированные системы.
СУБД общего назначения не ориентированы на какую-либо конкретную предметную область или на информационные потребности конкретной группы пользователей. Каждая система такого рода реализуется как программный продукт, способный функционировать на некоторой модели ЭВМ в определенной операционной обстановке. СУБД общего назначения обладает средствами настройки на работу с конкретной БД в условиях конкретного применения.
В некоторых ситуациях СУБД общего назначения не позволяют добиться требуемых проектных и эксплуатационных характеристик (производительность, занимаемый объем памяти и прочее). Тем не менее создание *специализированных СУБД* весьма трудоемкий процесс и для того, чтобы его реализовать, нужны очень веские основания. В процессе реализации своих функций СУБД постоянно взаимодействует с базой данных и с другими прикладными программными продуктами пользователя, предназначенными для работы с данной БД и называемыми приложениями.
Для того чтобы СУБД успешно справлялась со своими задачами, она должна обладать определенными возможностями.
Можно дать следующую обобщенную характеристику возможностям современных СУБД.
1. СУБД включает язык определения данных, с помощью которого можно определить базу данных, ее структуру, типы данных, а также средства задания ограничений для хранимой информации. В многопользовательском варианте СУБД этот язык позволяет формировать представления как некоторое подмножество базы данных, с поддержкой которых пользователь может создавать свой взгляд на хранимые данные, обеспечивать дополнительный уровень безопасности данных и многое другое.
2. СУБД позволяет вставлять, удалять, обновлять и извлекать информацию из базы данных посредством языка управления данными.

3. Большинство СУБД могут работать на компьютерах с разной архитектурой и под разными операционными системами, причем на работу пользователя при доступе к данным практически тип платформы влияния не оказывает.
4. Многопользовательские СУБД имеют достаточно развитые средства администрирования БД.
4. СУБД предоставляет контролируемый доступ к базе данных с помощью:
 системы обеспечения безопасности, предотвращающей несанкционированный доступ к информации базы данных;
 системы поддержки целостности базы данных, обеспечивающей непротиворечивое состояние хранимых данных;
 системы управления параллельной работой приложений, контролирующей процессы их совместного доступа к базе данных;
 системы восстановления, позволяющей восстановить базу данных до предыдущего непротиворечивого состояния, нарушенного в результате аппаратного или программного обеспечения.

2. АРХИТЕКТУРА СУБД

2.1. ТРЕХУРОВНЕВАЯ АРХИТЕКТУРА БАЗЫ ДАННЫХ

Одним из важнейших аспектов развития СУБД является идея отделения логической структуры БД и манипуляций данными, необходимыми пользователям, от физического представления, требуемого компьютерным оборудованием.

Одна и та же БД в зависимости от точки зрения может иметь различные уровни описания. По числу уровней описания данных, поддерживаемых СУБД, различают одно-, двух- и трехуровневые системы. В настоящее время чаще всего поддерживается трехуровневая архитектура описания БД (рис. 2.1), с тремя уровнями абстракции, на которых можно рассматривать базу данных.

Такая архитектура включает:

- внешний уровень, на котором пользователи воспринимают данные, где отдельные группы пользователей имеют свое представление (ПП) на базу данных;
- внутренний уровень, на котором СУБД и операционная система воспринимают данные;
- концептуальный уровень представления данных, предназначенный для отображения внешнего уровня на внутренний уровень, а также для обеспечения необходимой их независимости друг от друга; он связан с обобщенным представлением пользователей.

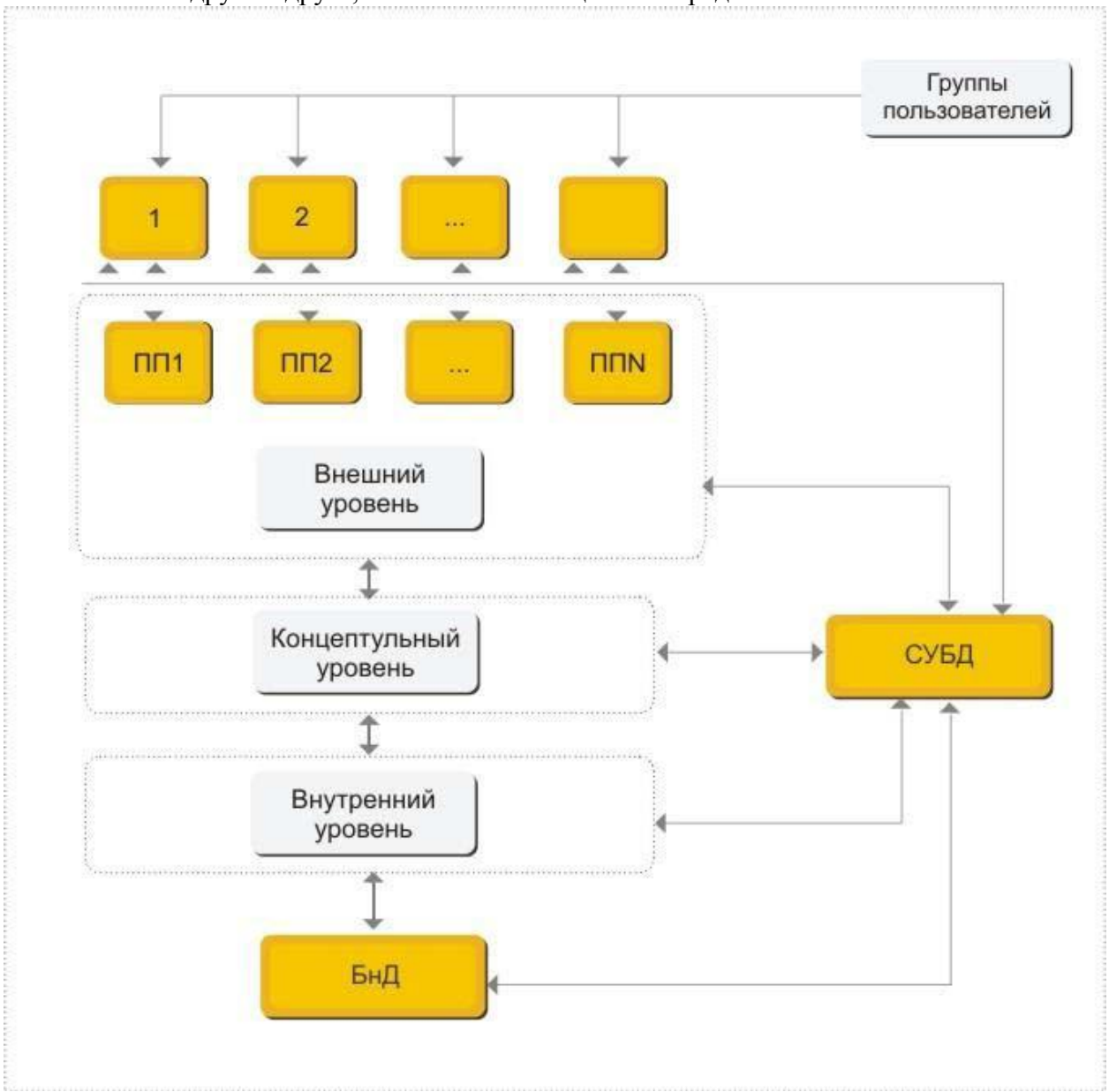

Рис. 2.1. Трехуровневая архитектура СУБД

Описание структуры данных на любом уровне называется *схемой.* Существует три различных типа схем базы данных, которые определяются в соответствии с уровнями

абстракции трехуровневой архитектуры. На самом высоком уровне имеется несколько внешних схем или подсхем, которые соответствуют разным представлениям данных. На концептуальном уровне описание базы данных называют *концептуальной схемой,* а на самом низком уровне абстракции — *внутренней схемой.*

Основным назначением трехуровневой архитектуры является обеспечение независимости от данных. Суть этой независимости заключается в том, что изменения на нижних уровнях никак не влияют на верхние уровни. Различают два типа независимости от данных: логическую и физическую.

Логическая независимость от данных означает полную защищенность внешних схем от изменений, вносимых в концептуальную схему. Такие изменения концептуальной схемы, как добавление или удаление новых сущностей, атрибутов или связей, должны осуществляться без необходимости внесения изменений в уже существующие внешние схемы для других групп пользователей.

Физическая независимость от данных означает защищенность концептуальной схемы от изменений, вносимых во внутреннюю схему. Такие изменения внутренней схемы, как использование различных файловых систем или структур хранения, разных устройств хранения, модификация индексов или хеширование, должны осуществляться без необходимости внесения изменений в концептуальную или внешнюю схемы.

Далее рассмотрим каждый из трех названных уровней.

Внешний уровень — это пользовательский уровень. Пользователем может быть программист, или конечный пользователь, или администратор базы данных. Представление базы данных с точки зрения пользователей называется *внешним представлением.* Каждая группа пользователей выделяет в моделируемой предметной области, общей для всей организации, те сущности, атрибуты и связи, которые ей интересны. Эти частичные или переопределенные описания БД для отдельных групп пользователей или ориентированные на отдельные аспекты предметной области называют *подсхемой.*

Концептуальный уровень является промежуточным уровнем в трехуровневой архитектуре и обеспечивает представление всей информации базы данных в абстрактной форме. Описание базы данных на этом уровне называется концептуальной схемой, которая является результатом концептуального проектирования.

Концептуальное проектирование базы данных включает анализ информационных потребностей пользователей и определение нужных им элементов данных. Таким образом, *концептуальная схема* — это единое логическое описание всех элементов данных и отношений между ними, логическая структура всей базы данных. Для каждой базы данных имеется только одна концептуальная схема.

Концептуальная схема должна содержать:

- сущности и их атрибуты;
- связи между сущностями;
- ограничения, накладываемые на данные;
- семантическую информацию о данных;
- обеспечение безопасности и поддержки целостности данных.

Внутренний уровень является третьим уровнем архитектуры БД. Внутреннее представление не связано с физическим уровнем, так как физический уровень хранения информации обладает значительной индивидуальностью для каждой системы.

На нижнем уровне находится внутренняя схема, которая является полным описанием внутренней модели данных. Для каждой базы данных существует только одна внутренняя схема.

Внутренняя схема описывает физическую реализацию базы данных и предназначена для достижения оптимальной производительности и обеспечения экономного использования дискового пространства. На внутреннем уровне хранится следующая информация:

- распределение дискового пространства для хранения данных и индексов;
- описание подробностей сохранения записей (с указанием реальных размеров сохраняемых элементов данных);
- сведения о размещении записей;
- сведения о сжатии данных и выбранных методах их шифрования.

СУБД отвечает за установление соответствия между всеми тремя типами схем разных уровней, а также за проверку их непротиворечивости. Ниже внутреннего уровня находится *физический уровень,* который контролируется операционной системой, но под руководством СУБД. Физический уровень учитывает, каким образом данные будут представлены в машине. Он обеспечивает физический взгляд на базу данных: дисководы, физические адреса, индексы, указатели и т. д. За этот уровень отвечают проектировщики физической базы данных, которые работают только с известными операционной системе элементами. Область их интересов: указатели, реализация последовательного распределения, способы хранения полей внутренних записей на диске. Однако функции СУБД и операционной системы на физическом уровне не вполне четко разделены и могут варьироваться от системы к системе.

2.2. ФУНКЦИИ СУБД

Управление данными во внешней памяти

Данная функция предоставляет пользователям возможности выполнения самых основных операций, которые осуществляются с данными, — это сохранение, извлечение и обновление информации. Она включает в себя обеспечение необходимых структур внешней памяти как для хранения данных, непосредственно входящих в БД, так и для служебных целей, например для ускорения доступа к данным.

Управление транзакциями

Транзакция — это последовательность операций над БД, рассматриваемых СУБД как единое целое. Транзакция представляет собой набор действий, выполняемых с целью доступа или изменения содержимого базы данных. Примерами простых транзакций может служить добавление, обновление или удаление в базе данных сведений о некоем объекте. Сложная же транзакция образуется в том случае, когда в базу данных требуется внести сразу несколько изменений. Инициализация транзакции может быть вызвана отдельным пользователем или прикладной программой.

Восстановление базы данных

Одним из основных требований к СУБД является надежность хранения данных во внешней памяти. Под надежностью хранения понимается то, что СУБД должна быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя. Обычно рассматриваются два возможных вида аппаратных сбоев:

- мягкие сбои, которые можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания);
- жесткие сбои, характеризуемые потерей информации на носителях внешней памяти.

Поддержание надежности хранения данных в БД требует избыточности хранения данных, причем та часть данных, которая используется для восстановления, должна храниться особо надежно. Наиболее распространенным методом поддержания такой избыточной информации является ведение журнала изменений БД.

Поддержка языков БД

Для работы с базами данных используются специальные языки, называемые языками баз данных.

В современных СУБД обычно поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с БД, начиная от ее создания, и обеспечивающий базовый пользовательский интерфейс с базами данных. Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык SQL (Structured Query Language — язык структурированных запросов). Язык SQL позволяет определять схему реляционной БД и манипулировать данными.

Словарь данных

Одной из основополагающих идей рассмотренной выше трехуровневой архитектуры является наличие интегрированного системного каталога с данными о схемах,

пользователях, приложениях и т. д. Системный каталог, который еще называют словарем данных, является, таким образом, хранилищем информации, описывающей данные в базе данных. Предполагается, что каталог доступен как пользователям, так и функциям СУБД. Обычно в словаре данных: содержится следующая информация:

- имена, типы и размеры элементов данных;
- имена связей;
- накладываемые на данные ограничения поддержки целостности;
- имена пользователей, которым предоставлено право доступа к данным;
- внешняя, концептуальная и внутренняя схемы и отображения между ними;
- статистические данные, например частота транзакций и счетчики обращений к объектам базы данных.

Управление параллельным доступом

Одна из основных целей создания и использования СУБД заключается в том, чтобы множество пользователей могло осуществлять параллельный доступ к совместно обрабатываемым данным. Параллельный доступ сравнительно просто организовать, если все пользователи выполняют только чтение данных, поскольку в этом случае они не могут помешать друг другу. Однако когда два или больше пользователей одновременно получают доступ к базе данных, конфликт с нежелательными последствиями легко может возникнуть, например, если хотя бы один из них попытается обновить данные.

СУБД должна гарантировать, что при одновременном доступе к базе данных многих пользователей подобных конфликтов не произойдет.

Управление буферами оперативной памяти

СУБД обычно работают с БД значительного размера. Понятно, что если при обращении к любому элементу данных будет производиться обмен с внешней памятью, то вся система будет работать со скоростью устройства внешней памяти. Практически единственным способом реального увеличения этой скорости является буферизация данных в оперативной памяти. В развитых СУБД поддерживается собственный набор буферов оперативной памяти с собственной дисциплиной замены буферов.

Контроль доступа к данным

СУБД должна иметь механизм, гарантирующий возможность доступа к базе данных только санкционированных пользователей и защищающий ее от любого несанкционированного доступа.

В современных СУБД поддерживается один из двух широко распространенных подходов к вопросу обеспечения безопасности данных: избирательный подход или обязательный подход.

В большинстве современных систем предусматривается избирательный подход, при котором некий пользователь обладает различными правами при работе с разными объектами. Значительно реже применяется альтернативный, обязательный подход, где каждому объекту данных присваивается некоторый классификационный уровень, а каждый пользователь обладает некоторым уровнем допуска.

Поддержка целостности данных

Термин *целостность* используется для описания корректности и непротиворечивости хранимых в БД данных. Реализация поддержки целостности данных предполагает, что СУБД должна содержать сведения о тех правилах, которые нельзя нарушать при работе с данными, и обладать инструментами контроля за тем, чтобы данные и их изменения соответствовали заданным правилам.

2.3. ЯЗЫКИ БАЗ ДАННЫХ

В СУБД поддерживается несколько специализированных по своим функциям подъязыков. Их можно разбить на две категории:

- язык определения данных БД — ЯОД (DDL — Data Definition Language);
- язык манипулирования данными— ЯМД (DML — Data Manipulation , Language).

2.3.1. ЯЗЫК ОПРЕДЕЛЕНИЯ ДАННЫХ

Язык определения данных — описательный язык, с помощью которого описывается предметная область: именуются объекты, определяются их свойства и связи между объектами. Он используется главным образом для определения логической структуры БД.

Схема базы данных, выраженная в терминах специального языка определения данных, состоит из набора определений. Язык ЯОД используется как для определения новой схемы, так и для модификации уже существующей.

Результатом компиляции ЯОД — операторов является набор таблиц, хранимый в системном каталоге, в котором содержатся метаданные — т. е. данные, которые включают определения записей, элементов данных, а также другие объекты, представляющие интерес для пользователей или необходимые для работы СУБД. Перед доступом к реальным данным СУБД обычно обращается к системному каталогу.

2.3.2. ЯЗЫКИ МАНИПУЛИРОВАНИЯ ДАННЫМИ

Язык манипулирования данными содержит набор операторов манипулирования данными, т. е. операторов, позволяющих заносить данные в БД, удалять, модифицировать или выбирать существующие данные.

Множество операций над данными можно классифицировать следующим образом:.

1. операции селекции;
2. действия над данными:

 включение — ввод экземпляра записи в БД с установкой его связей;

 удаление — исключение экземпляра записи из БД с установкой новых связей;

 модификация — изменение содержимого экземпляра записи и коррекция связей при необходимости.

Языки манипулирования данными делятся на два типа. Это разделение обусловлено коренным различием в подходах к работе с данными, а следовательно, различием в базовых конструкциях в работе с данными.

Первый тип — это процедурный ЯМД.

Второй тип — это декларативный (непроцедурный) ЯМД.

К процедурным языкам манипулирования данными относятся и языки, поддерживающие операции реляционной алгебры, которую основоположник теории реляционных баз данных Э. Ф. Кодд ввел для управления реляционной базой данных. Реляционная алгебра — это процедурный язык обработки реляционных таблиц, где в качестве операндов выступают таблицы в целом.

Декларативные языки предоставляют пользователю средства, позволяющие указать лишь то, какие данные требуются. Решение вопроса о том, как их следует извлекать, берет на себя процессор данного языка, работающий с целыми наборами записей.

Реляционные СУБД обычно включают поддержку непроцедурных языков манипулирования данными — чаще всего это бывает язык структурированных запросов SQL или язык запросов по образцу QBE.

В настоящее время нормой является поддержка декларативного языка SQL, в основе которого лежит реляционное исчисление, также введенное Э Коддом. Этот язык стал стандартом для языков реляционных баз данных, что позволяет использовать один и тот же синтаксис и структуру команд при переходе от одной СУБД к другой

Следует отметить, что язык SQL имеет сразу два компонента: язык DDL (ЯОД) для описания структуры базы данных, и язык DML (ЯМД) для выборки и обновления данных.

Другим широко используемым языком обработки данных является язык QBE, который заслужил репутацию одного из самых простых способов извлечения информации из базы данных. Особенно это ценно для пользователей, не являющихся профессионалами в этой области Язык предоставляет графические средства создания запросов на выборку данных с использованием шаблонов Ответ на запрос также представляет собой графическую информацию

Часть непроцедурного языка ЯМД, которая отвечает за извлечение данных, называется *языком запросов* Язык запросов можно определить как высокоуровневый

узкоспециализированный язык, предназначенный для удовлетворения различных требований по выборке информации из базы данных.

2.4. АРХИТЕКТУРА МНОГОПОЛЬЗОВАТЕЛЬСКИХ СУБД

2.4.1. МОДЕЛИ ДВУХУРОВНЕВОЙ ТЕХНОЛОГИИ "КЛИЕНТ — СЕРВЕР"

Систему баз данных можно рассматривать как систему, где осуществлено распределение процесса выполнения по принципу взаимодействия двух программных процессов, один из которых в этой модели называется "клиентом", а другой, обслуживающий клиента, — *сервером* (машина, хранящая базы данных). Клиентский процесс запрашивает некоторые услуги, а серверный процесс обеспечивает их выполнение. При этом предполагается, что один серверный процесс может обслужить множество клиентских процессов (рис. 2.2).

Рис. 2.2. Структура системы БД с выделением клиентов и сервера

Сервер в простейшем случае — это собственно СУБД. Он поддерживает все основные функции СУБД и предоставляет полную поддержку на внешнем, концептуальном и внутреннем уровнях.

Клиенты — это различные приложения, которые выполняются над СУБД.

Обычно в приложении выделяются следующие группы функций:

- функции ввода и отображения данных;
- прикладные функции, определяющие основные алгоритмы решения задач приложения;
- функции обработки данных внутри приложения,
- функции управления информационными ресурсами;
- служебные функции, играющие роль связок между функциями первых четырех групп.

Если все пять компонентов приложения распределяются только между двумя процессами, которые выполняются на двух платформах: на клиенте и на сервере, то такая модель называется *двухуровневой.* Она имеет несколько основных разновидностей. Рассмотрим их.

Файловый сервер

Модель файлового сервера называется моделью *удаленного управления данными.* Данная модель предполагает следующее распределение функций - на клиенте располагаются почти все части приложения: презентационная часть приложения, прикладные функции, а также функции управления информационными ресурсами. Файловый сервер содержит файлы, необходимые для работы приложений и самой СУБД и поддерживает доступ к файлам (рис. 2.3).

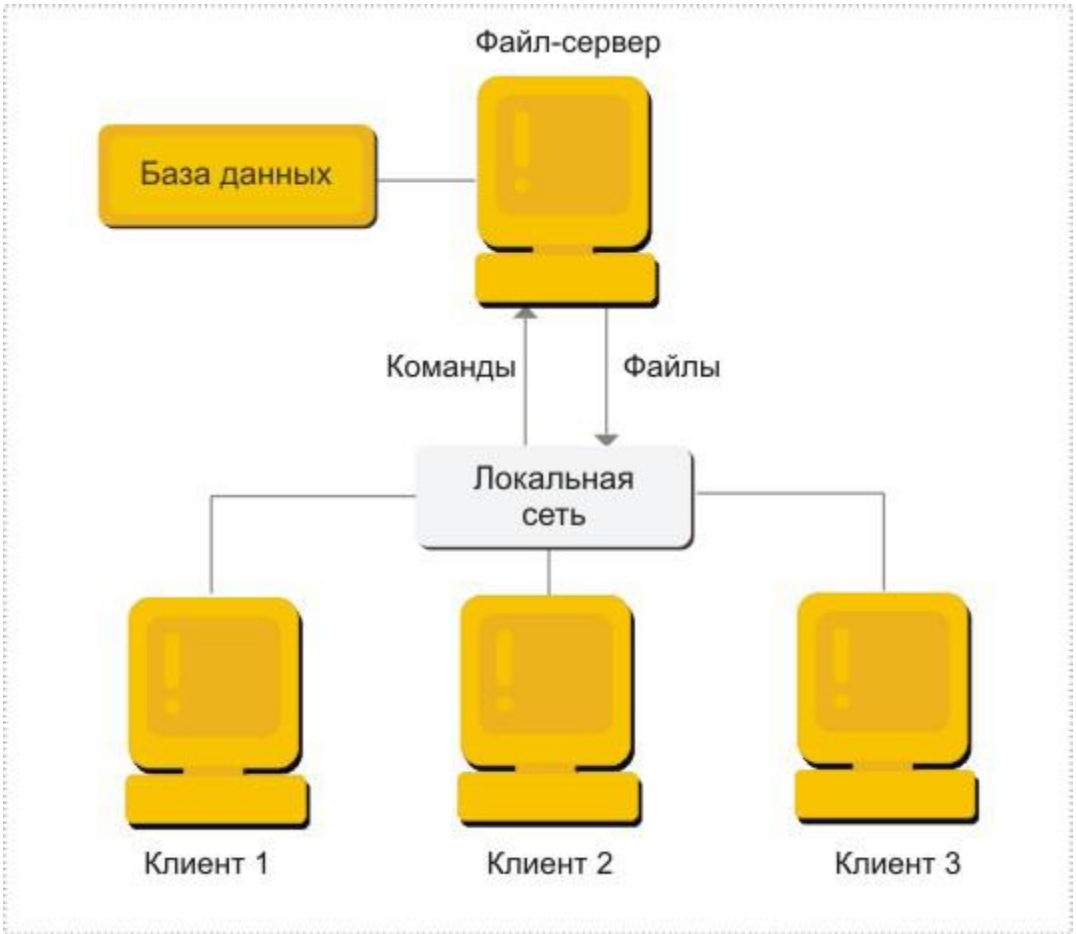

Рис. 2.3. Модель файлового сервера

Поскольку передача файлов представляет собой длительную процедуру, такой подход характеризуется значительным сетевым трафиком, что может привести к снижению производительности всей системы в целом.

Помимо этого недостатка использование файлового сервера несет еще и другие:

 на каждой рабочей станции должна находиться полная копия СУБД;

 управление параллельностью, восстановлением и целостностью усложняется, поскольку доступ к одним и тем же файлам могут осуществлять сразу несколько экземпляров СУБД;

 узкий спектр операций манипулирования данными, который определяется только файловыми командами;

 защита данных осуществляется только на уровне файловой системы.

Основное достоинство этой модели, заключается в том, что в ней уже осуществлено разделение монопольного приложения на два взаимодействующих процесса. При этом сервер может обслуживать множество клиентов, обращающихся к нему с запросами.

Модель удаленного доступа к данным

В модели удаленного доступа база данных также хранится на сервере. На сервере же находится и ядро СУБД. На клиенте располагаются части приложения, поддерживающие функции ввода и отображения данных и прикладные функции.

Клиент обращается к серверу с запросами на языке SQL. Структура модели удаленного доступа приведена на рис. 2.4.

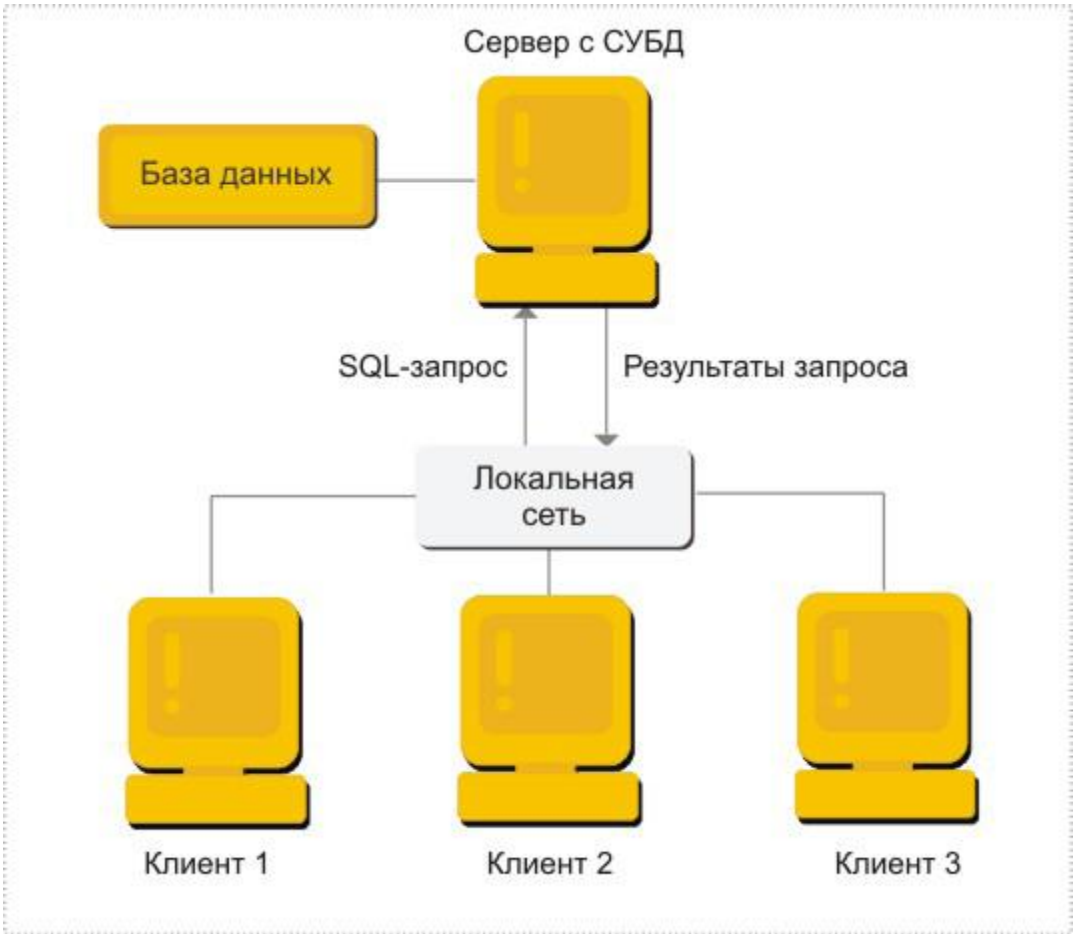

Рис. 2.4. Модель удаленного доступа

Сервер принимает и обрабатывает запросы со стороны клиентов, проверяет полномочия пользователей, гарантирует соблюдение ограничений целостности, выполняет обновление данных, выполняет запросы и возвращает результаты клиенту, поддерживает системный каталог, обеспечивает параллельный доступ к базе данных и ее восстановление. К тому же резко уменьшается загрузка сети, так как по ней от клиентов *к* серверу передаются не файловые команды, а запросы на SQL, и их объем существенно меньше. В ответ на запросы клиент получает только данные, соответствующие запросу, а не блоки файлов, как в модели файлового сервера.

Тем не менее, данная технология обладает и рядом недостатков:

 запросы на языке SQL при интенсивной работе клиентских приложений могут существенно загрузить сеть;

 презентационные и прикладные функции приложения должны быть повторены для каждого клиентского приложения;

 сервер в этой модели играет пассивную роль, поэтому функции управления информационными ресурсами должны выполняться на клиенте.

Модель сервера баз данных

Технологию "клиент — сервер" поддерживают большинство современных СУБД: Informix, Ingres, Sybase, Oracle, MS SQL Server. В основу данной модели добавлен механизм хранимых процедур и механизм триггеров.

Механизм *хранимых процедур* позволяет создавать подпрограммы, работающие на сервере и управляющие его процессами.

Таким образом, размещение на сервере хранимых процедур означает, что прикладные функции приложения разделены между клиентом и сервером. Трафик обмена информацией между клиентом и сервером резко уменьшается.

Централизованный контроль целостности базы данных в модели сервера баз данных выполняется с использованием механизма триггеров. Триггеры также являются частью БД.

Триггер — это особый тип хранимой процедуры, реагирующий на возникновение определенного события в БД. Он активизируется при попытке изменения данных — при операциях добавления, обновления и удаления. Триггеры определяются для конкретных таблиц БД.

Внедрение триггеров незначительно влияет на производительность сервера и часто используется для усиления приложений, выполняющих многокаскадные операции в БД.

В данной модели (рис. 2.5) сервер является активным, потому что не только клиент, но и сам сервер, используя механизм триггеров, может быть инициатором обработки данных в БД. Поскольку функции клиента облегчены переносом части прикладных функций на сервер, он в этом случае называется "тонким". При всех положительных качествах данной модели у нее все же есть один недостаток — очень большая загрузка сервера.

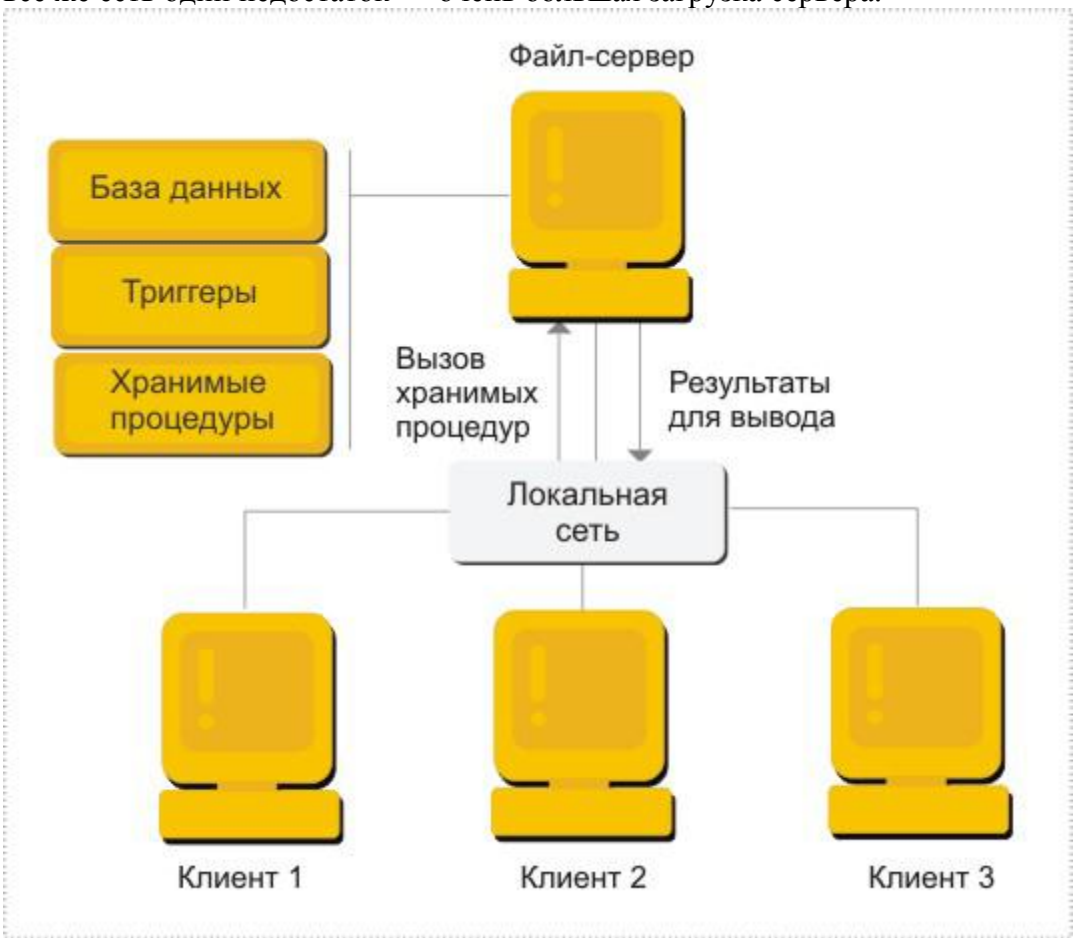

Рис. 2.5. Модель сервера БД

2.4.2. СЕРВЕР ПРИЛОЖЕНИЙ. ТРЕХУРОВНЕВАЯ МОДЕЛЬ

Эта модель является расширением двухуровневой модели и в ней вводится дополнительный промежуточный уровень между клиентом и сервером. Архитектура трехуровневой модели приведена на рис. 2.5.

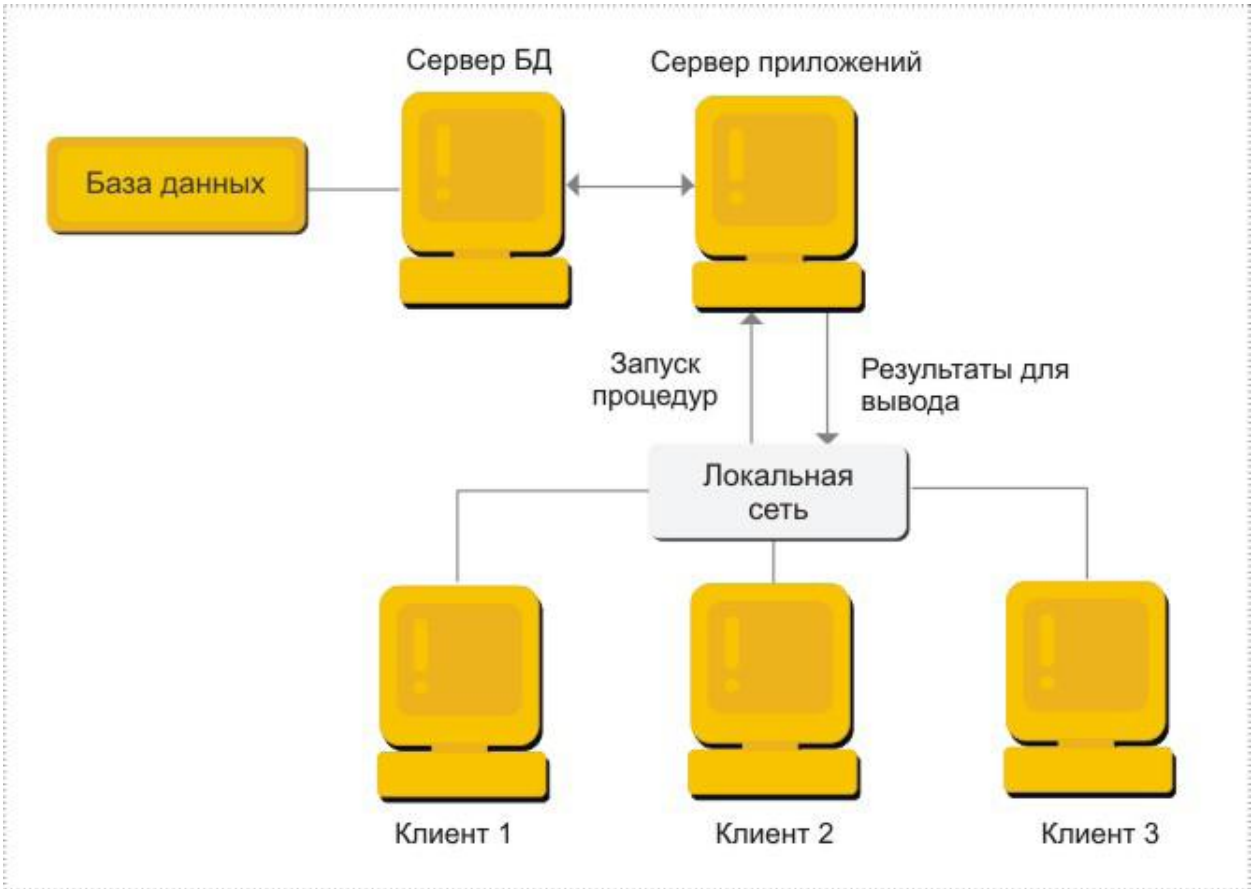

Рис. 2.5. Архитектура трехуровневой модели

Такая архитектура предполагает, что на *клиенте* располагаются: функции ввода и отображения данных, включая графический пользовательский интерфейс, локальные редакторы, коммуникационные функции, которые обеспечивают доступ клиенту в локальную или глобальную сеть.

Серверы баз данных в этой модели занимаются исключительно функциями управления информационными ресурсами БД: обеспечивают функции создания и ведения БД, поддерживают целостность БД, осуществляют функции создания резервных копий БД и восстановления БД после сбоев, управления выполнением транзакций и так далее.

Промежуточному уровню, который может содержать один или несколько серверов приложений, выделяются общие не загружаемые функции для клиентов: наиболее общие прикладные функции клиента, функции, поддерживающие сетевую доменную операционную среду, каталоги с данными, функции, обеспечивающие обмен сообщениями и поддержку запросов.

Преимущества трехуровневой модели наиболее заметны в тех случаях, когда клиенты выполняют сложные аналитические расчеты над базой данных.

3. МОДЕЛИ ДАННЫХ

Моделью данных называется формализованное описание структуры единиц информации и операций над ними в информационной системе.

Модель данных — это некоторая абстракция, в которой отражаются самые важные аспекты функционирования выделенной предметной области, а второстепенные — игнорируются. Модель данных включает в себя набор понятий для описания данных, связей между ними и ограничений, накладываемых на данные. В модели данных различают три главные составляющие:

- структурную часть, определяющую правила порождения допустимых для данной СУБД видов структур данных;
- управляющую часть, определяющую возможные операции над такими структурами;
- классы ограничений целостности данных, которые могут быть реализованы средствами этой системы.

Каждая СУБД поддерживает ту или иную модель данных.

По существу модель данных, поддерживаемая механизмами СУБД, полностью определяет множество конкретных баз данных, которые могут быть созданы средствами этой системы, а также способы модификации состоянии БД с целью отображения тех изменений, которые происходят в предметной области.

3.1. КЛАССИФИКАЦИЯ МОДЕЛЕЙ ДАННЫХ

В настоящее время описано много разнообразных моделей, построение которых преследует разные цели. Из множества опубликованных моделей данных можно выделить три категории:

- объектные модели данных;
- модели данных на основе записей;
- физические модели данных.

Применительно к трехуровневой архитектуре баз данных следует отметить, что первые две категории используются для описания данных на внешнем и концептуальном уровнях, а последняя категория — на внутреннем уровне.

Объектные модели данных

Среди объектных моделей следует выделить ER-модель, которая наиболее часто используется в методологии проектирования баз данных, а также объектно-ориентированную модель, последнее время широко используемую в технологиях баз данных. Объектно-ориентированная модель расширяет понятие объекта, включая в него не только атрибуты, характеризующие состояние объекта, но и связанные с ним действия.

Модели данных на основе записей

В модели данных на основе записей база данных состоит из нескольких записей фиксированного формата, которые могут иметь разные типы.

В большинстве коммерческих СУБД используются ставшие классическими два типа такого рода моделей данных: теоретико-графовые (ТГ) и теоретико-множественные (ТМ) модели данных.

К теоретико-графовым моделям относятся две разновидности:

- сетевые модели;
- иерархические модели.

В таких моделях данных предусматриваются характерные для подобного рода структур операции навигации и манипулирования данными.

Аппарат навигации в ТГ-моделях служит для установки тех объектов данных, к которым будет применяться очередная операции манипулирования данными.

Теоретико-множественные модели используют математический аппарат, реляционную алгебру (знаковая обработка множеств), реляционное исчисление. К моделям данного типа относятся реляционные модели.

В соответствии с реляционной моделью данных БД представляется в виде совокупности таблиц, над которыми могут выполняться операции, формируемые в терминах реляционной алгебры и реляционного исчисления.

Физическая модель данных

Физические модели данных описывают то, как данные хранятся в компьютере, представляя информацию о структуре записей, их упорядоченности и существующих путях доступа. Наиболее распространены из них следующие: обобщающая модель и модель памяти кадров.

3.2. СЕТЕВАЯ МОДЕЛЬ

Сети — естественный способ представления отношений между объектами, всевозможных их взаимосвязей. Сетевая модель опирается на математическую структуру, которая называется направленным графом. Направленный граф состоит из узлов, соединенных ребрами. В контексте моделей данных узлы представляют собой объекты в виде типов записей данных, а ребра — связи между объектами со степенью кардинальности "один к одному" или "один ко многим.

Иерархическая модель данных является частным случаем сетевой
модели.

3.2.1. СТРУКТУРЫ ДАННЫХ СЕТЕВОЙ МОДЕЛИ

В сетевой модели используются несколько различных типовых структур данных, главными из которых являются типы записей и наборы. Для построения этих структур применяются такие конструктивные элементы, как элемент данных и агрегат (рис. 3.1).

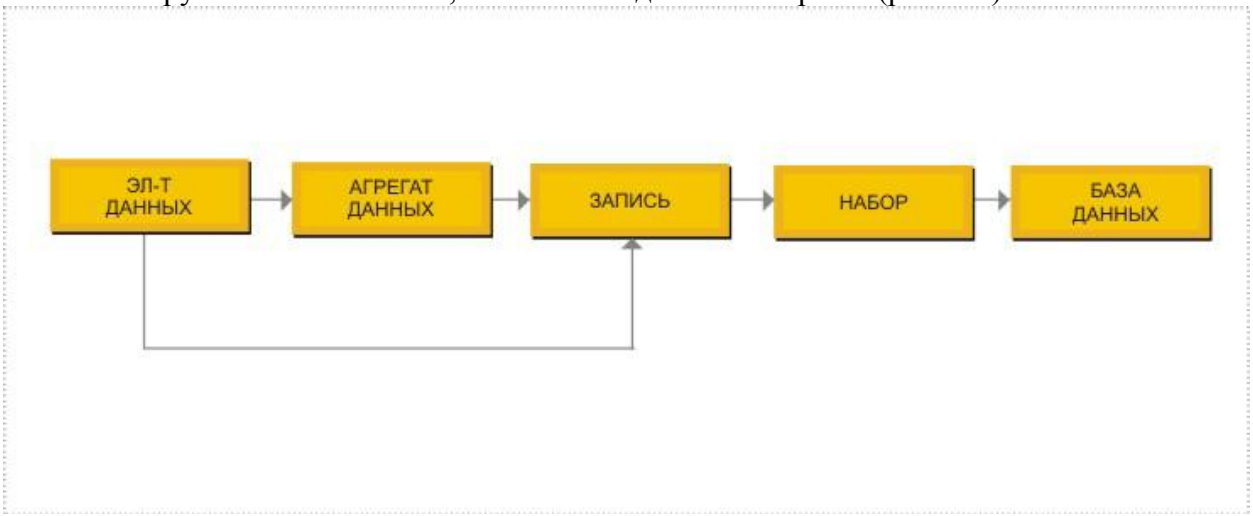

Рис. 3.1. Основные структуры сетевой модели данных

Элемент данных — это наименьшая поименованная информационная единица данных, доступная пользователю (аналог — поле в файловой системе). Элемент данных должен иметь свой тип (не структурный, простой).

Агрегат данных соответствует следующему уровню обобщения — поименованная совокупность элементов данных внутри записи или другого агрегата (рис. 3.2).

Рис. 3.2. Агрегат **Дата**

Запись — конечный уровень агрегации. Каждая запись представляет собой именованную структуру, содержащую один или более именованных элементов данных, каждый из которых обладает своим особым форматом.

Агрегат данных **Дата** входит в состав записи **Сотрудник** (рис. 3.3).

Тип записей — это совокупность логически связанных экземпляров записей. Тип записей моделирует некоторый класс объектов реального мира.
В качестве элемента данных могут быть использованы только простые типы, а в качестве агрегатов могут быть использованы сложные типы: вектор и повторяющаяся группа.

Рис. 3.3. Запись **Сотрудник**

Агрегат типа вектор соответствует линейному набору элементов данных (рис. 3.2). Агрегат типа повторяющаяся группа соответствует совокупности векторов данных. Так, например, в заказе может быть указано несколько видов товаров с числом повторений 10 (рис. 3.4).

Рис. 3.3. Запись **Заказ**

Набор — это поименованная двухуровневая иерархическая структура, которая содержит запись владельца и записи членов. Наборы выражают связи "один ко многим" или "один к одному" между двумя типами записей. Тип набора поддерживает работу с внутренними структурами типов записей.
Набор, приведенный на рис. 3.5, определяет тип записи-владельца **Отдел** и тип записи-члена набора **Сотрудник**, а также тип связи между ними "один ко многим" — с именем **Работает.** Имя набора — это метка, присвоенная стрелке. Связь типа "один ко многим" допускает возможность того, что с данным экземпляром записи-владельца может быть связан ноль, один, или несколько экземпляров записи-члена.

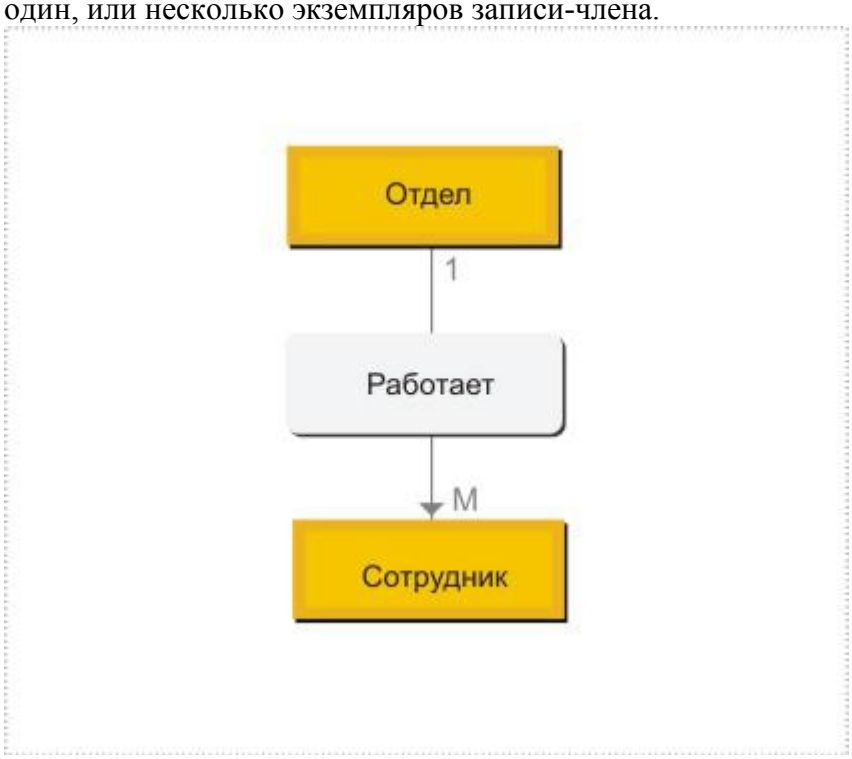

Рис. 3.4. Диаграмма типа набора **Работает**

Используя понятия сетевой модели данных, можно получить другое изображение такого набора (рис. 3.6), где представлены логические типы записей **Отдел** и **Сотрудник**, их структура и связь между типами записей **Работает.**

База данных в сетевой модели данных — это поименованная совокупность экземпляров записей различного типа и экземпляров наборов, содержащих связи между ними.

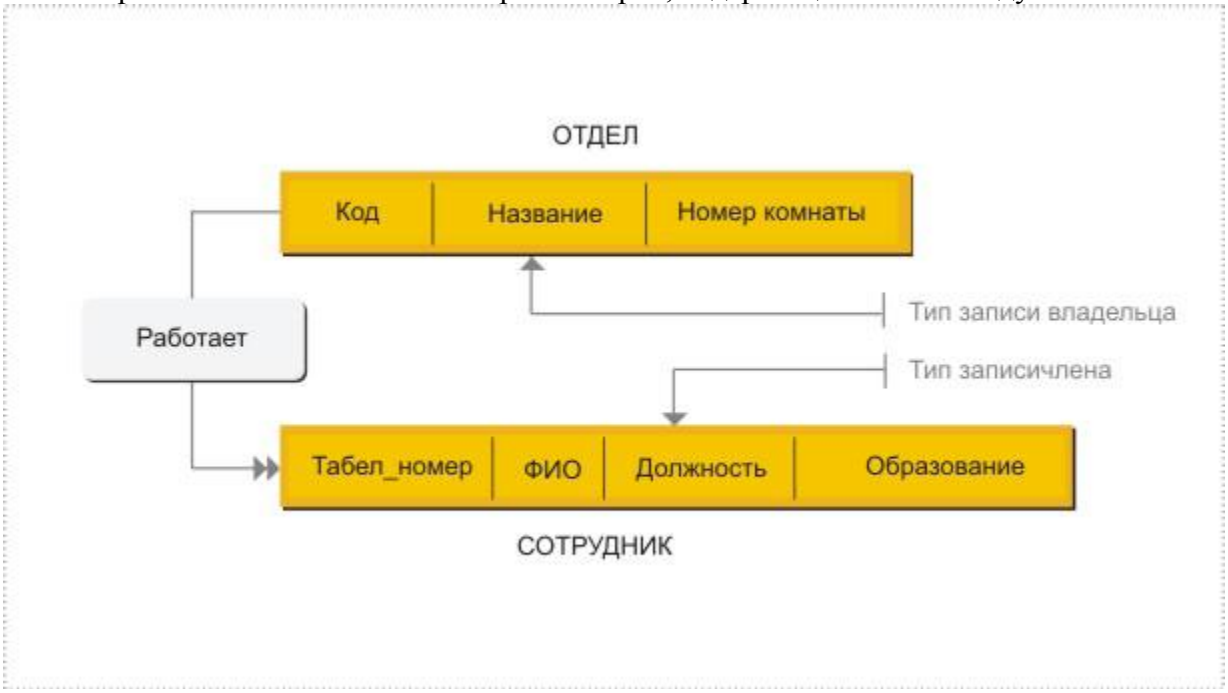

Рис. 3.5. Набор **Работает** между двумя типами записей **Отдел** и **Сотрудник**

3.2.2. ПРЕОБРАЗОВАНИЕ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ В СЕТЕВУЮ

Сетевая модель данных может быть без осложнений получена из концептуальной модели. Для этого надо предположить, что в последней используются только бинарные связи. Причем они должны принадлежать к типам "один к одному" или "один ко многим". При этом вместо сущностей концептуальной модели необходимо использовать типы записей сетевой модели, где имена сущностей становятся именами типов записей, атрибуты сущностей становятся полями записей, связь между сущностями превращается в связь между типами записей.

Бинарные связи, принадлежащие к типу "один ко многим", переносятся в сетевую модель следующим образом: тип записи со стороны "один" становится владельцем, а тип записи со стороны "много" становится типом записи-члена. Для связи типа "один к одному" выбор типа записи-владельца и типа записи-члена может быть осуществлен произвольно.

3.2.3. УПРАВЛЯЮЩАЯ ЧАСТЬ СЕТЕВОЙ МОДЕЛИ

Для манипулирования данными в сетевой модели данных определен ряд типичных операций, которые можно подразделить на две группы: навигационные операции и операции модификации.

Навигационные операции осуществляют перемещение по БД путем прохождения по связям, определенным в схеме БД. В результате таких операций определяется запись, которая называется текущей. К подобным операциям относятся:

- найти конкретную запись в наборе однотипных записей и сделать ее текущей;
- перейти от записи-владельца к записи-члену в некотором наборе;
- перейти к следующей записи в некоторой связи;
- перейти от записи-члена к владельцу по некоторой связи.

Операции модификации осуществляют как добавление новых экземпляров отдельных типов записей, так и экземпляров новых наборов, удаление экземпляров записей и наборов, модификацию отдельных компонентов самой записи. Для реализации этих операций в системе текущее состояние детализируется путем запоминания трех его составляющих:

текущего набора, текущего типа записи, текущего экземпляра типа записи. В такой ситуации возможны следующие операции:
 извлечь текущую запись в буфер прикладной программы для обработки;
 заменить в извлеченной записи значения указанных элементов данных на заданные новые их значения;
 запомнить запись из буфера в БД;
 создать новую запись;
 уничтожить запись;
 включить текущую запись в текущий экземпляр набора;
 исключить текущую запись из текущего экземпляра набора.
Поддержание ограничений целостности в сетевых моделях в принципе не требуется.

3.3. ИЕРАРХИЧЕСКАЯ МОДЕЛЬ ДАННЫХ

Первые системы управления базами данных использовали иерархическую модель данных, и во времени их появление предшествует появлению сетевой модели.

3.3.1. СТРУКТУРНАЯ ЧАСТЬ ИЕРАРХИЧЕСКОЙ МОДЕЛИ

Основными информационными единицами в иерархической модели данных являются сегмент и поле. Поле данных определяется как наименьшая неделимая единица данных, доступная пользователю. Для сегмента определяются тип сегмента и экземпляр сегмента. Экземпляр сегмента образуется из конкретных значений полей данных. Тип сегмента — это поименованная совокупность входящих в него типов полей данных.
Как и сетевая, иерархическая модель данных базируется на графовой форме построения данных, и на концептуальном уровне она является просто частным случаем сетевой модели данных. В иерархической модели данных вершине графа соответствует тип сегмента или просто сегмент, а дугам — типы связей предок — потомок. В иерархических структурах сегмент — потомок должен иметь в точности одного предка.
Иерархическая модель представляет собой связный неориентированный граф древовидной структуры, объединяющий сегменты. Иерархическая БД состоит из упорядоченного набора деревьев (рис.3.7).

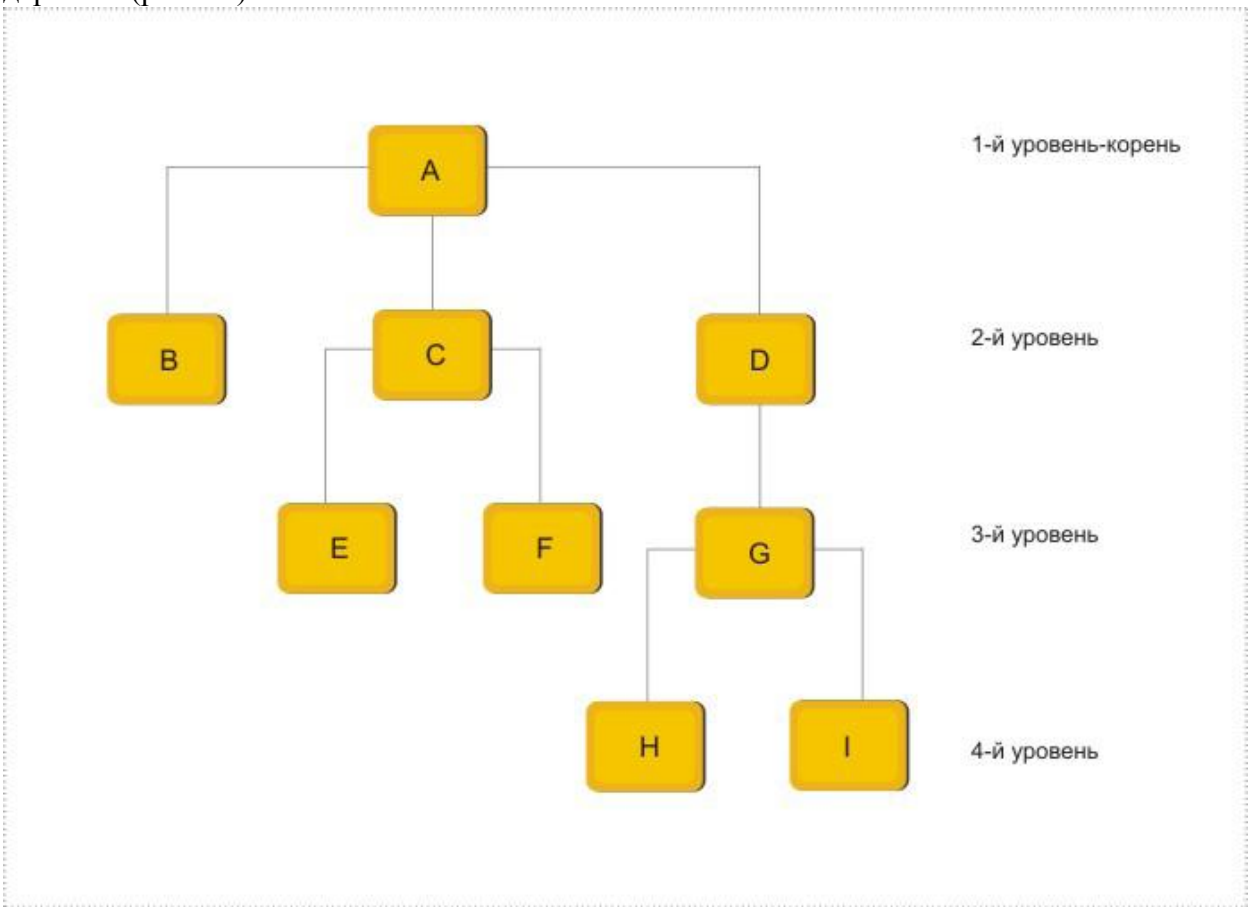

Рис. 3.7. Иерархическая модель данных

3.3.2. ПРЕОБРАЗОВАНИЕ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ В ИЕРАРХИЧЕСКУЮ МОДЕЛЬ ДАННЫХ

Преобразование концептуальной модели в иерархическую структуру данных во многом схоже с преобразованием ее в сетевую модель, но и имеет некоторые отличия в связи с тем, что иерархическая модель требует организации всех данных в виде дерева.

Преобразование связи типа "один ко многим" между предком и потомком осуществляется практически автоматически в том случае, если потомок имеет одного предка, и происходит это следующим образом. Каждый объект с его атрибутами, участвующий в такой связи, становится логическим сегментом. Между двумя логическими сегментами устанавливается связь типа "один ко многим". Сегмент со стороны "много" становится потомком, а сегмент со стороны "один" становится предком.

Ситуация значительно усложняется, если потомок в связи имеет не одного, а двух и более предков. Так как подобное положение является невозможным для иерархической модели, то отражаемая структура данных нуждается в преобразованиях, которые сводятся к замене одного дерева, например, двумя (если имеется два предка). В результате такого преобразования в базе данных появляется избыточность, так как единственно возможный выход из этой ситуации — дублирование данных.

3.3.3. УПРАВЛЯЮЩАЯ ЧАСТЬ ИЕРАРХИЧЕСКОЙ МОДЕЛИ

В рамках иерархической модели выделяют языковые средства описания данных (ЯОД) и средства манипулирования данными (ЯМД).

Каждая физическая база описывается набором операторов, обусловливающих как ее логическую структуру, так и структуру хранения БД. При этом способ доступа устанавливает способ организации взаимосвязи физических записей. Определены следующие способы доступа:

- иерархически последовательный;
- иерархически индексно-последовательный;
- иерархически прямой;
- иерархически индексно-прямой;
- индексный.

Помимо задания имени БД и способа доступа описания должны содержать определения типов сегментов, составляющих БД, в соответствии с иерархией, начиная с корневого сегмента. Каждая физическая БД содержит только один корневой сегмент, но в системе может быть несколько физических БД.

Среди операторов манипулирования данными можно выделить операторы поиска данных, операторы поиска данных с возможностью модификации, операторы модификации данных. Набор операций манипулирования данными в иерархической БД невелик, но вполне достаточен.

Примеры типичных операторов поиска данных:

- найти указанное дерево БД;
- перейти от одного дерева к другому;
- найти экземпляр сегмента, удовлетворяющий условию поиска;
- перейти от одного сегмента к другому внутри дерева;
- перейти от одного сегмента к другому в порядке обхода иерархии.

Примеры типичных операторов поиска данных с возможностью модификации:

- найти и удержать для дальнейшей модификации единственный экземпляр сегмента, удовлетворяющий условию поиска;
- найти и удержать для дальнейшей модификации следующий экземпляр сегмента с теми же условиями поиска;
- найти и удержать для дальнейшей модификации следующий экземпляр для того же родителя.

Примеры типичных операторов модификации иерархически организованных данных, которые выполняются после выполнения одного из операторов второй группы (поиска данных с возможностью модификации):

- вставить новый экземпляр сегмента в указанную позицию;
- обновить текущий экземпляр сегмента;
- удалить текущий экземпляр сегмента.

В иерархической модели автоматически поддерживается целостность ссылок между предками и потомками. Основное правило: никакой потомок не может существовать без своего родителя.

4. РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

4.1. ИСТОРИЯ ВОПРОСА

Создателем реляционной модели является сотрудник фирмы IBM доктор Э. Ф. Кодд. Будучи по образованию математиком, Э. Кодд предложил использовать для обработки данных аппарат теории множеств. В статье "A Relational Model of Data for Large Shared Data Banks", вышедшей в свет в 1970 году, он показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как отношение (relation).

Положив теорию отношений в основу реляционной модели, Э. Кодд обосновал реляционную замкнутость отношений и ряда некоторых специальных операций, которые применяются сразу ко всему множеству строк отношения, а не к отдельной строке. Указанная реляционная замкнутость означает, что результатом выполнения операций над отношениями является также отношение, над которым в свою очередь можно осуществить некоторую операцию. Из этого следует, что в данной модели можно оперировать реляционными выражениями, а не только отдельными операндами в виде простых имен таблиц.

Одним из основных преимуществ реляционной модели является ее однородность. Все данные рассматриваются как хранимые в таблицах и только в таблицах. Каждая строка такой таблицы имеет один и тот же формат.

К числу достоинств реляционного подхода можно отнести:

 наличие небольшого набора абстракций, которые позволяют сравнительно просто моделировать большую часть распространенных предметных областей и допускают точные формальные определения, оставаясь интуитивно понятными;

 наличие простого и в то же время мощного математического аппарата, опирающегося главным образом на теорию множеств и математическую логику и обеспечивающего теоретический базис реляционного подхода к организации БД;

 возможность ненавигационного манипулирования данными без необходимости знания конкретной физической организации баз данных во внешней памяти.

Кодд сформулировал двенадцать приведенных ниже правил, которым должна соответствовать настоящая реляционная база данных.

1. Правило информации. Вся информация в базе данных должна быть представлена исключительно на логическом уровне и только одним способом — в виде значений, содержащихся в таблицах.

2. Правило гарантированного доступа. Логический доступ ко всем и каждому элементу данных (атомарному значению) в реляционной базе данных должен обеспечиваться путем использования комбинации имени таблицы, первичного ключа и имени столбца.

3. Правило поддержки недействительных значений. В настоящей реляционной базе данных должна быть реализована поддержка недействительных значений, которые отличаются от строки символов нулевой длины, строки пробельных символов и от нуля или любого другого числа и используются для представления отсутствующих данных независимо от типа этих данных.

4. Правило динамического каталога, основанного на реляционной модели. Описание базы данных на логическом уровне должно быть представлено в том же виде, что и основные данные, чтобы пользователи, обладающие соответствующими правами, могли работать с ним с помощью того же реляционного языка, который они применяют для работы с основными данными.

4. Правило исчерпывающего подъязыка данных. Реляционная система может поддерживать различные языки и режимы взаимодействия с пользователем (например, режим вопросов и ответов). Однако должен существовать, по крайней мере, один язык, операторы которого можно представить в виде строк символов в соответствии с некоторым четко определенным синтаксисом и который в полной мере поддерживает следующие элементы:

 определение данных;

 определение представлений;

 обработку данных (интерактивную и программную);

 условия целостности;

 идентификацию прав доступа;
 границы транзакций (начало, завершение и отмена).

5. Правило обновления представлений. Все представления, которые теоретически можно обновить, должны быть доступны для обновления.

7. Правило добавления, обновления и удаления. Возможность работать с отношением как с одним операндом должна существовать не только при чтении данных, но и при добавлении, обновлении и удалении данных.

8. Правило независимости физических данных. Прикладные программы и утилиты для работы с данными должны на логическом уровне оставаться нетронутыми при любых изменениях способов хранения данных или методов доступа к ним.

6. Правило независимости логических данных. Прикладные программы и утилиты для работы с данными должны на логическом уровне оставаться нетронутыми при внесении в базовые таблицы любых изменений, которые теоретически позволяют сохранить нетронутыми содержащиеся в этих таблицах данные.

10. Правило независимости условий целостности. Должна существовать возможность определять условия целостности, специфические для конкретной реляционной базы данных, на подъязыке реляционной базы данных и хранить их в каталоге, а не в прикладной программе.

11. Правило независимости распространения. Реляционная СУБД не должна зависеть от потребностей конкретного клиента.

12. Правило единственности. Если в реляционной системе есть низкоуровневый язык (обрабатывающий одну запись за один раз), то должна отсутствовать возможность использования его для того, чтобы обойти правила и условия целостности, выраженные на реляционном языке высокого уровня (обрабатывающем несколько записей за один раз).

4.2. СТРУКТУРНАЯ ЧАСТЬ РЕЛЯЦИОННОЙ МОДЕЛИ

4.2.1. ОТНОШЕНИЕ

Реляционная база данных — это конечный (ограниченный) набор отношений. Отношения используются для представления сущностей, а также для представления связей между сущностями. Отношение — это двумерная таблица, имеющая уникальное имя и состоящая из строк и столбцов, где строки соответствуют записям, а столбцы — атрибутам. Каждая строка в таблице представляет некоторый объект реального мира или соотношения между объектами.

Атрибут — это поименованный столбец отношения. Свойства сущности, его характеристики определяются значениями атрибутов. Порядок следования атрибутов не влияет на само отношение.

Пусть имеется отношение r. Схемой отношения r называется конечное множество имен атрибутов R = {A1, A2, ... An}. Заголовки столбцов отношения содержат имена его атрибутов и, следовательно, все вместе отражают его схему.

Схема отношения **ПРЕПОДАВАТЕЛЬ** может быть представлена следующим образом: **{Таб_ном_преп, Фамилия, Должность}**

Или

ПРЕПОДАВАТЕЛЬ
Таб_ном_преп
Фамилия
Должность

Тогда заголовок отношения **ПРЕПОДАВАТЕЛЬ** примет вид

Таб_ном_преп	**Фамилия**	**Должность**

Отношение строится с учетом ряда факторов. Каждому имени атрибута Ai, 1 >= i <= n ставится в соответствие множество допустимых для соответствующего столбца значений. Это множество Di, называется доменом данного имени атрибута.

Каждая строка отношения является множеством значений, взятых по одному из домена каждого имени атрибута. Домены являются произвольными непустыми конечными или счетными множествами и образуют множество:
D = D1 ∪ D2 ∪ ... ∪ Dn.
Отношение r со схемой R — это конечное множество отображений {t1, t2, ... tp*)* из R в D. Причем каждое отображение t∈ r должно удовлетворять следующему ограничению:
t(Ai*)* принадлежит Di где 1 <= i <= n.
Эти отображения называются *кортежами.* Каждый кортеж отношения отображает экземпляр сущности, а атрибут отношения отображает атрибут сущности.
Множество кортежей называется *телом отношения.* Тело отношения отражает состояние сущности, поэтому во времени оно постоянно меняется. Тело отношения характеризуется *кардинальным числом,* которое равно количеству содержащихся в нем кортежей.
Одной из главных характеристик отношения является его степень. *Степень отношения* определяется количеством атрибутов, которое в нем присутствует. Эта характеристика отношения имеет еще названия: ранг и арность. Отношение с одним атрибутом называется унарным, с двумя атрибутами — бинарным, с тремя — тернарным, с n атрибутами n-арным. Определение степени отношения осуществляется по заголовку отношения.
Все названные характеристики отношения обозначены на рис. 4.1

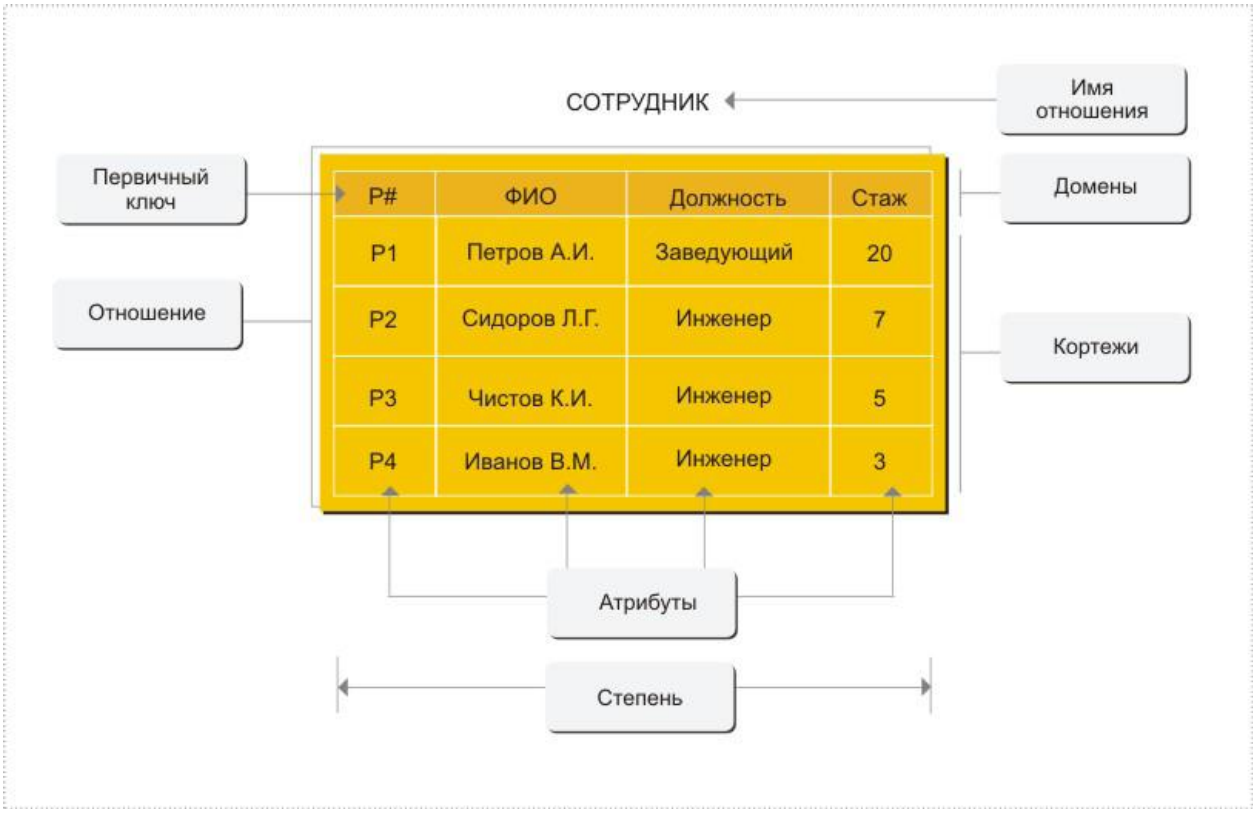

Рис. 4.1. Отношение **СОТРУДНИК**

4.2.2. СВОЙСТВА И ВИДЫ ОТНОШЕНИЙ

Отношение по структуре подобно таблице, но таблице, обладающей определенными свойствами. Сведем воедино все свойства отношения.
 Отношение имеет имя, которое отличается от имен всех других отношений.
 Отношение представляется в виде табличной структуры.
 Каждый атрибут имеет уникальное имя, его значения берутся из одного и того же домена.
 Каждый компонент кортежа является простым, атомарным значением, не состоящим из группы значений.
 Упорядочение атрибутов теоретически несущественно, однако оно может влиять на эффективность доступа к кортежам.
 Все строки (кортежи) должны быть различны.

 Теоретически порядок следования кортежей не имеет значения. В реляционной теории встречается несколько видов отношений, но не все они поддерживаются реальными системами. Различают:
 именованное отношение — это переменная отношения, определенная в СУБД посредством специальных операторов;
 базовое отношение — это именованное отношение, являющееся частью базы данных;
 производное отношение — это отношение, определенное посредством реляционного выражения через базовые отношения;
 представление — это именованное виртуальное производное отношение, представленное в системе исключительно через определение в терминах других именованных отношений;
 снимки — это отношения, подобные представлениям, но они сохраняются, доступны для чтения и периодически обновляются;
 результат запроса — это неименованное производное отношение, получаемое в результате запроса, которое для сохранения необходимо преобразовать в именованное отношение;
 хранимое отношение — это отношение, которое поддерживается в физической памяти.

4.2.3. РЕЛЯЦИОННЫЕ КЛЮЧИ

В отношении могут существовать несколько одиночных или составных атрибутов, которые однозначно идентифицируют кортеж отношения. Это — потенциальные ключи.
Говорят, что множество атрибутов К = {Ai, Aj, ..., Ak} отношения r является потенциальным ключом r тогда и только тогда, когда удовлетворяются два независимых от времени условия:
 уникальность: в произвольный заданный момент времени никакие два различных кортежа r не имеют одного и того же значения для Ai, Aj, ..., Ак;
 минимальность: ни один из атрибутов Ai, Aj, ... Ак не может быть исключен из К без нарушения уникальности.
Отношение может иметь несколько потенциальных ключей. Ключ, содержащий два и более атрибута, называется составным ключом. Каждое отношение обладает хотя бы одним возможным ключом, поскольку в отношении не может быть одинаковых кортежей, а это значит, что, по меньшей мере, комбинация всех его атрибутов удовлетворяет условию уникальности. Потенциальные ключи, позволяя гарантированно выделить точно один кортеж, обеспечивают основной механизм адресации на уровне кортежей реляционной модели.
Один из возможных ключей (выбранный произвольным образом) принимается за его первичный ключ. Обычно первичным ключом назначается тот возможный ключ, которым проще всего пользоваться при повседневной работе. Остальные возможные ключи, если они есть, называются альтернативными ключами. Для индикации связи между отношениями используются внешние ключи.
Внешний ключ — это набор атрибутов одного отношения, являющийся потенциальным ключом другого отношения.
Благодаря наличию связок между потенциальными и внешними ключами обеспечивается взаимосвязь кортежей определенных отношений. Отношение, содержащее внешний ключ, называется дочерним или ссылающимся отношением. А отношение, содержащее связанный с внешним ключом потенциальный ключ, — родительским или целевым отношением.
Отношения не могут рассматриваться как статические объекты, так как они предназначены для отражения некоторой части реального мира, а эта часть реального мира может изменяться во времени. Поэтому и отношения изменяются во времени: кортежи могут добавляться, удаляться или модифицироваться. Тем не менее, предполагается, что сама схема отношения инвариантна во времени. Отношение должно восприниматься как множество возможных состояний, которые может принимать отношение.

Пример

Пусть рассматривается концептуальная модель, приведенная на рис. 4.2. Пример относится к предметной области, которую можно назвать "Преподавательская деятельность". Данная модель содержит две сущности: **ЛЕКТОР** и **ПРЕДМЕТ**, между которыми установлена связь **ЧИТАЕТ** типа "многие ко многим". Характеристики сущностей представлены изображенными на рисунке атрибутами. Связь **ЧИТАЕТ** не имеет собственных атрибутов.

Для преобразования концептуальной модели в реляционную модель разработан ряд технологий, знакомство с которыми состоится несколько позже.

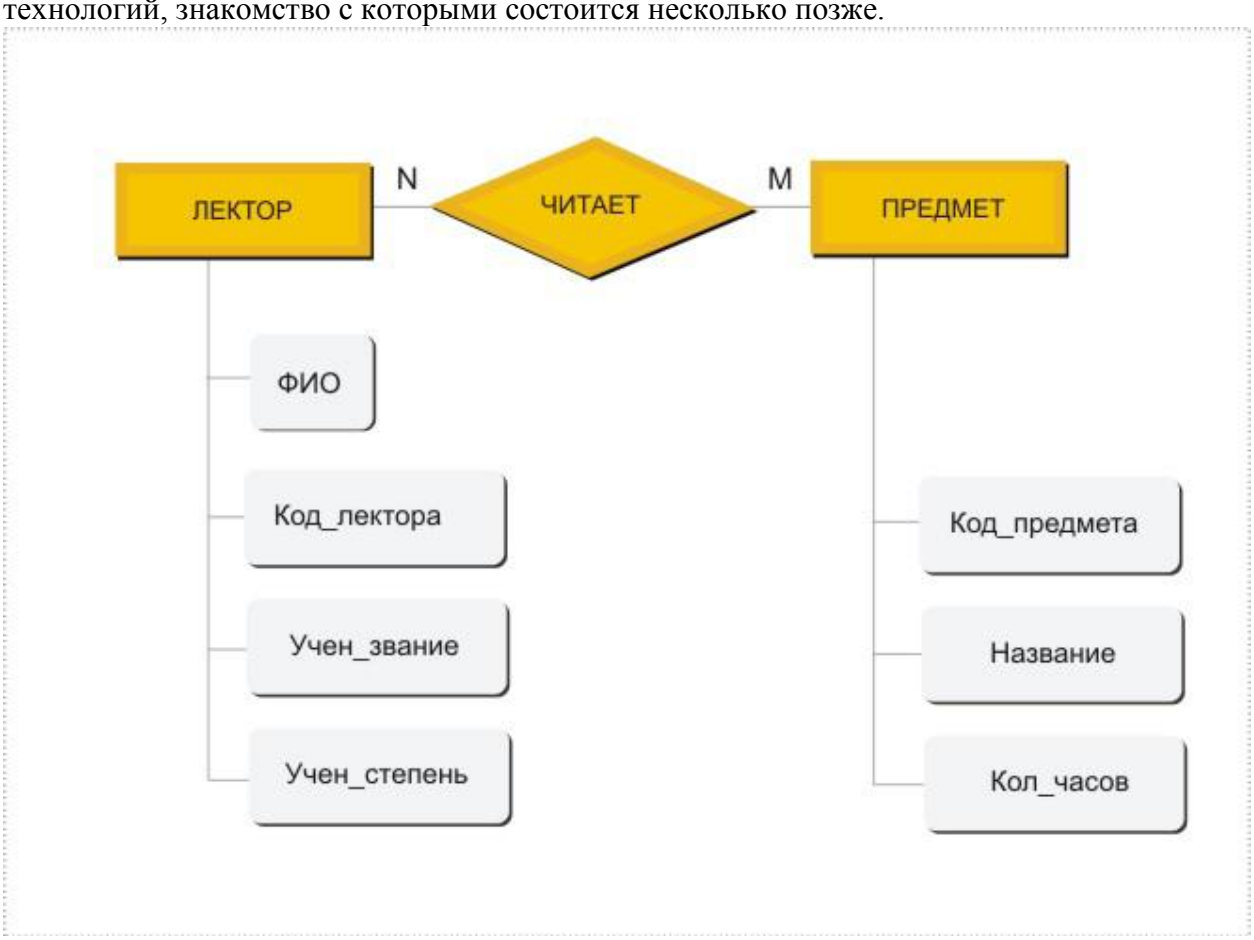

Рис. 4.2. Концептуальная модель для примера

В данный момент, не вдаваясь в подробности причин принятого решения, просто приведем реляционную схему, соответствующую указанной концептуальной модели. Она включает в себя три отношения: **ЛЕКТОР, ПРЕДМЕТ, ЧИТАЕТ**. Схемы отношений и связи между ними изображены на рис. 4.3.

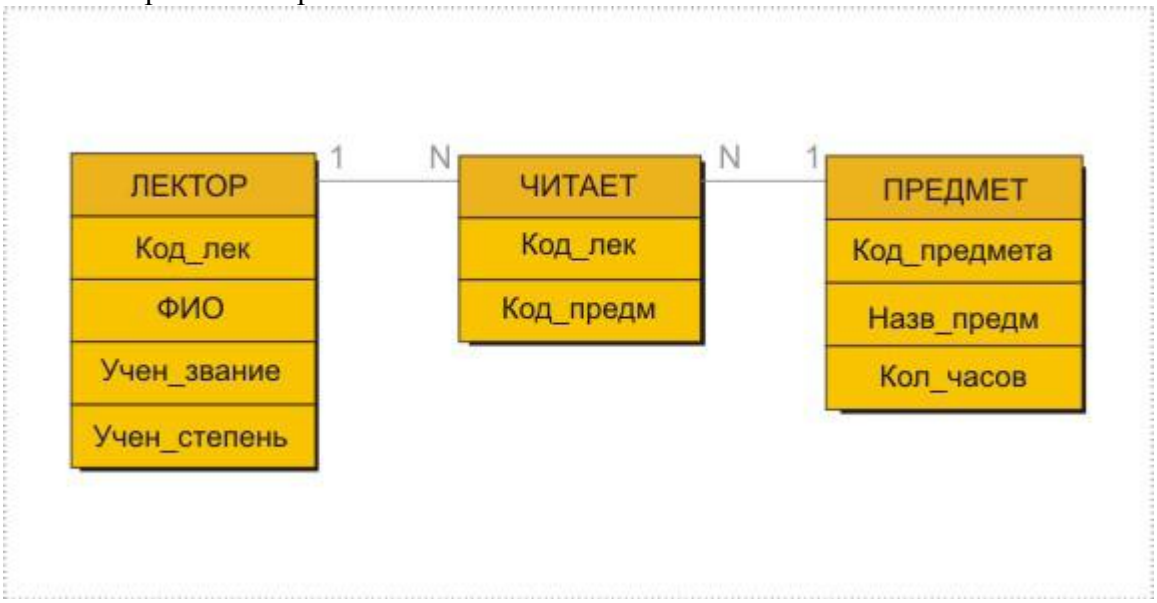

Рис. 4.3. Реляционная схема базы данных для примера

На рис. 4.4 даны соответствующие отношения, заполненные кортежами. Эти отношения имеют следующие характеристики:

 ЛЕКТОР — 4-арное отношение с первичным ключом **Код_лек** с кардинальным числом, равным четырем; атрибуты определены на следующих доменах: **Код_лек** — {целые: 1...4},

ФИО— {возможные фамилии и инициалы}, **Уч_степень** — {к.т.н., д.т.н., нет степени}, **Уч_звание** — {Доцент, Профессор, Нет_звания};

 ПРЕДМЕТ — тернарное отношение с первичным ключом **Код_предм**. с кардинальным числом, равным шести; атрибуты определены на следующих доменах: **Код_предм** — {символьный}. **Назв_предм** — {Информатика, Программирование, Физика, ООП, Базы данных, Базы данных}, **Кол_во_час** — {целые: 54, 102, 36};

 ЧИТАЕТ — бинарное отношение с составным первичным ключом **Код_лек, Код_предм,** с кардинальным числом, равным шести, в котором присутствуют первичные ключи только читающих лекторов и первичные ключи только читаемых предметов.

ЛЕКТОР

Код_лек	ФИО	Уч_степень	Уч_звание
01	Петров А.И.	К.т.н.	Доцент
02	Сидоров Л.Г.	К.т.н.	Доцент
03	Чистов К.И.	Д.т.н.	Профессор
04	Иванов В.М.	К.т.н.	Доцент

ЧИТАЕТ

Код_прдм	Код Лек
01	02
02	02
03	04
04	01
05	03
06	01

ПРЕДМЕТ

Код_прдм	Назв_предм	Кол_во_час
01	Информатика	54
02	Базы данных	102
03	ООП	34
04	Программир-е	45
05	Физика	45
06	Структуры	45

Рис. 4.4. Реляционные отношения модели для примера

Все приведенные отношения являются нормализованными, поскольку атрибуты всех отношений имеют атомарные значения. Во всех трех отношениях отсутствуют дублирующие кортежи.

4.3. ОБНОВЛЕНИЕ ОТНОШЕНИЙ

Для обновления отношений необходимо иметь возможность выполнять следующие операции;

 добавление кортежа;

 удаление кортежа;

 изменение кортежа.

Рассмотрим их по порядку.

Операция добавления для отношения r со схемой (A1, A2, ..., An) имеет вид:

ADD (r: Al=dl, A2=d2, ..., An=dn).

Если порядок атрибутов фиксирован, возможна более короткая запись:

ADD (r: dl, d2,…, dn).

Выполнение этой операции может стать невозможным в ряде случаев:

 добавляемый кортеж не соответствует схеме определенного отношения;

 некоторые значения кортежей не принадлежат соответствующим доменам;

 описанный кортеж совпадает по ключу с кортежем, уже находящимся в отношении.

Операция удаления предназначена для удаления кортежей. Она может быть записана следующим образом:

DEL (r; Al=dl, A2=d2,…, An=dn),
или для упорядоченных атрибутов:
DEL (r; dl, d2,…, dn) .
Для удаления некоторого кортежа часто достаточно указать значение некоторого ключа:
DEL (r; ключ) .
Если указанный кортеж в отношении отсутствует, то отношение остается неизменным.
Операция изменения предназначена для модификации части кортежа. Для отношения r ее можно при {C1, C2, ... ,Cp} ∈ {A1, A2, ..., An} определить так:
CH (r; Al=dl, A2=d2, ..., An=dn; Cl=el, C2=e2, ..., Cp=ep).
Если K = {B1, B2, ..., Bk} является ключом, то запись данной операции может быть сокращена:
CH (r; Bl=dl, B2=d2, ..., Bk=dk; Cl=el, C2=e2, ..., Cp=ep).
Возможные ошибки в данном случае те же, что и у предыдущих операций:
 указанный кортеж не существует;
 изменения имеют неправильный формат;
 используемые значения не принадлежат соответствующим доменам.

4.4. ЦЕЛОСТНОСТЬ БАЗЫ ДАННЫХ

Поддержка целостности базы данных реализуется посредством ряда ограничений, накладываемых на данные.
Первый тип ограничений проистекает из того факта, что каждый атрибут определяется на своем домене, или наоборот: домен атрибута задает множество значений, которые может принимать атрибут. Указанное ограничение называется ограничением атрибута.
Важными понятиями в теории реляционных баз данных являются категорная целостность и целостность на уровне ссылок.
Категорная целостность ограничивает набор значений первичных ключей базовых отношений. Такого рода целостность заключается в следующем: кортеж не может записываться в БД до тех пор, пока значения его ключевых атрибутов не будут полностью определены. Иными словами: никакой ключевой атрибут любого кортежа отношения не может содержать отсутствующего значения, обозначаемого определителем NULL.
Целостность на уровне ссылок. При построении отношений для связывания строк одной таблицы со строками другой таблицы используются внешние ключи. База данных, в которой все непустые внешние ключи ссылаются на текущие значения ключей другого отношения, обладает целостностью на уровне ссылок.
Как правило, определение условия целостности данных осуществляется при установлении взаимосвязи между родительским и дочерним отношениями; при этом пользователю часто предоставляется возможность самому решить, каким путем сохранять целостность базы данных и насколько жестко должно соблюдаться условие целостности при различных операциях изменения данных.
Так, например, при изменении значения первичного ключа в родительском отношении часто разрешены следующие варианты действий, из которых два первых — обеспечивают нахождение базы данных в целостном непротиворечивом состоянии.
 При изменении значений полей первичного ключа в родительском отношении автоматически осуществляется каскадное изменение всех соответствующих значений в дочернем отношении.
 Не позволяется изменять значения полей первичного ключа в родительском отношении, если в дочернем отношении имеется хотя бы одна запись, содержащая ссылку на изменяемую запись.
 Позволяется изменять значения полей первичного ключа в родительском отношении, независимо от существования связанных записей в дочернем отношении. Целостность данных при этом не поддерживается. При удалении записи в родительском отношении также возможны несколько вариантов действий.
 При удалении записи в родительском отношении автоматически осуществляется каскадное удаление всех записей из дочернего отношения, связанных с удаляемой записью.

 Не позволяется удалять записи в родительском отношении, если в дочернем отношении имеется хотя бы одна запись, содержащая ссылку на удаляемую запись. При попытке удаления записи возникает ошибка, которую можно обработать программно.
 Позволяется удалять записи в родительском отношении независимо от существования связанных записей в дочернем отношении. Очевидно, что целостность данных при этом не поддерживается.

При добавлении новой записи в дочернее отношение или редактировании в нем существующей записи возможно выбрать один из вариантов:
 не позволяется вводить запись, если значение индексного выражения дочернего отношения не соответствует одной из записей в родительском отношении;
 при вводе данных в дочернее отношение не анализируется значение индексного выражения. Целостность данных при этом не поддерживается.

Еще один вид ограничений, накладываемый на информацию, содержащуюся в отношениях, связан с теми бизнес-правилами, которые существуют в данной организации или в моделируемой предметной области. Это так называемые дополнительные правила поддержки целостности данных, определяемые пользователем или администратором данных, которые называются корпоративными ограничениями целостности. Приведем примеры таких правил:
 студентам нельзя планировать занятия продолжительностью более восьми часов в день;
 студенты весь день должны заниматься в одном корпусе вуза, чтобы не требовалось дополнительное время на переходы из одного здания в другое

5. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ
5.1. ИЗБЫТОЧНОСТЬ ДАННЫХ И АНОМАЛИИ ОБНОВЛЕНИЯ В БД

Избыточность данных в БД относится к нежелательным явлениям, поскольку ведет к увеличению объема памяти, необходимого для физического хранения отношений. Избыточность вызывается, прежде всего, дублированием данных.

Вот характерный пример отношения (табл. 5.1), содержащего нежелательную избыточность:

Таблица 5.1. Отношение **СТУДЕНТ**

Ном_зач_кн	ФИО_студента	Код_группы	ФИО_старосты	Куратор
20-Т-201	Иванов С.И.	20-Т-11	Рябов В.С.	Доц. Фок И.И.
20-Т-215	Петров Я.Р.	20-Т-12	Сизов М.М.	Доц. Докин С.С.
20-Т-217	Рябов В.С	20-Т-11	Рябов В.С.	Доц. Фок И.И.
20-Т-211	Сенова А.Л.	20-Т-11	Рябов В.С.	Доц. Фок И.И.

В данном отношении с первичным ключом **Ном_зач_кн** в каждом кортеже о каждом студенте из одной и той же группы повторяются сведения о коде группы, старосте и кураторе.

При работе с отношениями, содержащими избыточные данные, могут возникнуть проблемы, которые называются аномалиями обновления.

Различают три вида аномалий в базе данных:

- аномалии включения;
- аномалии удаления;
- аномалии модификации.

Аномалии включения

В приведенном выше отношении аномалии включения возникают при попытке создать новую группу и ввести ее в отношение при том условии, что в нее еще не зачислен ни один студент. Ввод такой информации в подобной ситуации требует присвоения значения NULL всем атрибутам описания студента, в том числе и атрибуту **Ном_зач_кн**, который является первичным ключом данного отношения. Но реализация такой попытки приведет к нарушению категорней целостности, а значит, система ее обязана отклонить.

Результатом анализа является вывод о том, что в отношении табл. 5.1 присутствуют аномалии включения, а, следовательно, это отношение должно быть преобразовано таким образом, чтобы от них избавиться.

Структура отношений, содержащая ту же информацию, что и отношение **СТУДЕНТ**, но лишенная аномалий включения, представлена в табл. 5.2 и 5.3.

Таблица 6.2 Отношение **СТУДЕНТ**

Ном_зач_кн	ФИО_студента	Код_группы
20-Т-201	Иванов С.И.	20-Т-11
20-Т-215	Петров Я.Р.	20-Т-12
20-Т-217	Рябов В.С.	20-Т-11
20-Т-211	Сенова А.Л.	20-Т-11

Таблица 6.3 Отношение **ГРУППА**

Код_группы	ФИО_старосты	Куратор
20-Т-11	Рябов В.С.	Доц. Фок И.И.
20-Т-12	Сизов М.М.	Доц. Докин С.С.

Аномалии удаления

Вернемся к анализу отношения, представленного в табл. 5.1. При удалении из этого отношения кортежа:

20-Т-215 Петров Я.Р. 20-Т-12 Сизов М.М. Доц. Докин С.С.

из базы данных будут удалены все сведения о группе 20-Т-12. Такая ситуация представляет собой аномалию удаления.
Для исключения из базы данных аномалии удаления это отношение должно быть преобразовано. Причем преобразования должны быть проведены точно такие же, какие были проведены для исключения аномалии включения.

Аномалии модификации
Такая аномалия возникает при попытке изменить что-либо касающееся сведений о группе обучения студента. Допустим, что в группе 20-Т-11 решили назначить нового старосту, например, Сенову А.Л.
В такой ситуации необходимо просмотреть все кортежи отношения и в каждом кортеже значение атрибута **ФИО_старосты** заменить Рябов В.С. на Сенова А.Л.
Появление аномалии модификации можно заблокировать, если опять же прибегнуть к преобразованию отношения из табл. 5.1. Эти преобразования точно такие же, которые были использованы для исключения аномалий включения и удаления. Действительно, смена старосты группы требует изменения значения атрибута **ФИО_старосты** только в одном кортеже отношения табл. 5.3.

Проблема обратимости
Чтобы исключить различного рода аномалии из отношения, его подвергают процессу декомпозиции. При решении задачи декомпозиции возникают две проблемы.
Первая проблема — это проблема обратимости, заключающаяся в возможности восстановления исходной схемы, а именно — восстановления любого кортежа исходного отношения, используя кортежи полученных в результате декомпозиции отношений. Декомпозиция, удовлетворяющая этому требованию, называется декомпозицией, гарантирующей отсутствие потерь.
Вторая проблема связана с сохранением зависимостей. Следует напомнить, что проектирование базы данных включает в себя и определение ограничений, накладываемых на ее отношения. В процессе декомпозиции получаются новые отношения и ограничения, которые приписываются им, должны быть такими, чтобы были сохранены исходные ограничения.
Все сказанное означает, что проводимая декомпозиция должна сохранять эквивалентность схем при замене одной схемы на другую, т. е. нужна декомпозиция, гарантирующая отсутствие потерь и сохранение зависимостей.

5.2. НОРМАЛИЗАЦИЯ ОТНОШЕНИЙ

Для устранения рассмотренных выше недостатков и применяется процесс нормализация отношений. Данный процесс — это формальный метод анализа отношений на основе их первичных или потенциальных ключей и существующих функциональных зависимостей. Он включает ряд формальных правил, используемых для проверки всех отношений базы данных. Различают:
 1НФ — первую нормальную форму;
 2НФ — вторую нормальную форму;
 ЗНФ — третью нормальную форму;
 НФБК — нормальную форму Бойса — Кодда;
 4НФ — четвертую нормальную форму;
 5НФ — пятую нормальную форму.
Каждая нормальная форма налагает определенные ограничения на данные. Эти ограничения вводятся в каждом конкретном отношении, и соблюдение этих ограничений в отношении связано уже с наличием нормальной формы.
 1НФ, 2НФ, ЗНФ— ограничивают зависимость непервичных атрибутов от ключей.
 НФБК — ограничивает зависимость первичных атрибутов.
 4НФ — формулирует ограничения на виды многозначных зависимостей.
 5НФ — вводит другие типы зависимостей: зависимости соединений.

Процесс перехода от нормальной формы более низкого уровня к нормальной форме более высокого уровня и называется нормализацией отношений (НО).
Для реляционных баз данных необходимо, чтобы все отношения базы данных обязательно находились в 1НФ. Нормальные формы более высокого порядка могут использоваться разработчиками по своему усмотрению. Однако следует стремиться к тому, чтобы довести уровень нормализации базы данных хотя бы до ЗНФ, тем самым, исключив из базы данных избыточность данных и аномалии обновления.

5.2.1. ФУНКЦИОНАЛЬНЫЕ ЗАВИСИМОСТИ

Функциональная зависимость определяется для атрибутов, находящихся в одном и том же отношении в 1НФ. Функциональная зависимость является семантическим свойством атрибутов отношения.
Пусть дана схема отношения R(A1, A2, ..., An).
Атрибут A2 функционально зависит от атрибута A1 если каждому значению A1 соответствует единственное значение A2 (т. е. каждый кортеж, имеющий одно и то же значение A1, должен иметь одно и то же значение A2)
Обозначается подобная ситуация так: A1 → A2. Левая часть функциональной зависимости называется детерминантом, а правая часть — зависимой частью. Отсутствие функциональной зависимости обозначается A -/→ B.
Рассмотренная выше функциональная зависимость — это F- зависимость.
Дадим более общее формальное определение функциональных зависимостей.
Пусть r-отношение со схемой R, а X, Y— подмножества R. Отношение r удовлетворяет функциональной зависимости X → Y, если для любых двух кортежей t1 и t2 в r t1(X) = t2(X), то t1(Y) = t2(Y).

5.2.2. АКСИОМЫ ВЫВОДА

Для отношения r(R) в любой момент существует некоторое семейство F-зависимостей, которым это отношение удовлетворяет. Причем одно состояние отношения может удовлетворять F-зависимости, а другое — нет. Требуется выявить семейство F-зависимостей F, которому удовлетворяют все допустимые состояния r.
Множество функциональных зависимостей, применимых к отношению r(R), конечно, так как существует только конечное число подмножеств множества R. Таким образом, можно найти все F- зависимости, которым удовлетворяет r, перебрав все возможные. Время поиска можно сократить в том случае, если известны некоторые F- зависимости из F. В подобной ситуации сокращение времени поиска происходит часто из-за того, что остальные F-зависимости можно получить, использую аксиомы вывода.
Аксиома вывода — это правило, устанавливающее, что если отношение удовлетворяет определенным F-зависимостям, то оно должно удовлетворять к некоторым другим F-зависимостям.
Ниже приведены шесть аксиом вывода для F-зависимостей. В их формулировках используется обозначение r для отношения на R, и W, X, У, Z для подмножеств R.
Запишем кратко все аксиомы.
- F1. Рефлексивность: X → X.
- F2. Пополнение: X → Y влечет за собой XZ → Y.
- F3. Аддитивность: X → Y и X → Z влечет за собой X → YZ.
- F4. Проективность: X → YZ влечет за собой X → Y.
- F5-Транзитивность: X → Y и Y → Z влечет за собой X → Z.
- F5. Псевдотранзитивность: X → Y и YZ → W влечет за собой XZ →W.

5.2.3. ПЕРВАЯ НОРМАЛЬНАЯ ФОРМА

Отношение находится в первой нормальной форме, если все его атрибуты имеют простые (атомарные) значения. Другими словами, значения в домене каждого атрибута отношения не являются ни списками, ни множествами простых или сложных значений.
Определить понятия атомарности трудно. Значение, атомарное в одном приложении, может быть неатомарным в другом. Можно руководствоваться общим принципом, что значение не

атомарно, если в приложении оно используется по частям. Рассмотрим пример отношения, представленного в табл. 5.4.

Таблица 5.4. Отношение **РОЖДЕНИЕ**

Имя	Дата рождения
Анна	5 марта 1986
Александр	25 января 1987
Ольга	1 ноября 1987
Федор	14 сентября 1986

Если значение атрибута **Дата рождения** предполагается использовать целиком, то в этом случае данное отношение находится в 1НФ. Если бы потребовалось выделить и отдельно использовать, скажем, год, число, месяц, то это отношение не находилось бы в 1НФ, так как требуемые данные являются только частями значения атрибута **Дата рождения**. Чтобы перевести такое отношение в 1НФ, атрибут **Дата рождения** должен быть разбит на части так, как показано в табл. 5.5.

Таблица 5.5. Отношение **РОЖДЕНИЕ**

Имя	День рождения	Месяц рождения	Год рождения
Анна	5	Март	1986
Александр	25	Январь	1987
Ольга	1	Ноябрь	1987
Федор	14	Сентябрь	1986

Или, например, табл. 5.6 является ненормализованной, и она не находится в 1НФ потому, что включает величины, являющиеся совокупностью атомарных значений. Чтобы получить отношение РОД, находящееся в 1НФ, необходимо его представить так, как это сделано в табл. 5.7.

Таблица 5.6. Ненормализованная таблица РОД

Имя	Пол
{Александр, Федор}	Мужской
Ольга	Женский

Таблица 5.7. Отношение РОД

Имя	Пол
Александр	Мужской
Федор	Мужской
Ольга	Женский

5.2.4. ВТОРАЯ НОРМАЛЬНАЯ ФОРМА

Вторая и третья нормальные формы возникли в результате стремления избежать аномалий обновления данных при работе с БД и избавиться от информационной избыточности в отношениях.

Вторая нормальная форма применяется к отношениям с составными ключами, т. е. к таким отношениям, первичный ключ которых состоит из двух или более атрибутов. Отношение, у которого первичный ключ включает только один атрибут, всегда находится во 2НФ.

Дадим определения новых понятий.

Определение 1. Атрибут называется посторонним для функциональной зависимости $X \rightarrow Y$, если он может быть удален из правой или левой части функциональной зависимости без изменений транзитивного замыкания F^+ множества F.

Определение 2. Функциональная зависимость $X \rightarrow Y$ называется редуцированной слева, если X не содержит атрибута Z, постороннего для функциональной зависимости $X \rightarrow Y$. Если функциональная зависимость $X \rightarrow Y$ редуцирована слева, то Y является полностью зависящим от X. В противном случае Y частично зависит от X.

Определение 3. Для данной схемы отношения R атрибут A в R и множество функциональных зависимостей F на R атрибут A называется первичным в R относительно F, если A содержится в каком-нибудь ключе схемы R. В противном случае A называется непервичным в R.

Итак, схема отношения R находится во 2НФ относительно F, если она находится в 1НФ, и каждый непервичный атрибут функционально полно зависит от первичного ключа.
Схема всей БД имеет 2НФ относительно F, если каждая ее схема отношения находится относительно F во 2НФ.
Рассмотрим отношение **КОНСУЛЬТАЦИИ_ДИПЛОМНИКОВ** со схемой:
(Таб_Ном_преп, Ном_зач_кн, Дата, ФИО_преп, Должность, ФИО_студ, Тема_диплома, Время, Аудитория, Вместимость).
Это отношение содержит информацию о том, какой преподаватель консультирует определенного дипломника, а также дату, время консультации и аудиторию с ее вместимостью, где она должна проводиться. Обозначим основные зависимости этого отношения.
(Таб_Ном_преп, Ном_зач_кн, Дата) → (ФИО_преп, Должность, ФИО_студ, Тема_липлома, Время, Аудитория, Вместимость).
Описательные атрибуты преподавателя **ФИО_преп, Должность** зависят только от части первичного ключа. Данную ситуацию определяет зависимость:
Таб-Ном_преп → (ФИО_преп, Должность).
Описательные атрибуты студента **ФИО_студ, Тема_диплома** также зависят только от части первичного ключа и не зависят от остальных атрибутов ключа, то есть имеется зависимость вида:
Ном_зач_кн → (ФИО_студ. Тема_диплома).
Отсутствие полной функциональной зависимости каждого непервичного атрибута отношения от первичного ключа, как и в других рассмотренных здесь примерах, является источником аномалий обновления и вносит свою долю избыточности в базу данных.
Устранение данных отрицательных явлений осуществляется путем декомпозиции исходного отношения на три со следующими схемами:
ПРЕПОДАВАТЕЛЬ (Таб_Ном_преп, ФИО_преп, Должность);
СТУДЕНТ (Ном_зач_кн, ФИО_студ, Тема_диплома);
КОНСУЛЬТАЦИИ (Таб_Ном_преп, Ном_зач_кн, Дата, Время, Аудитория, Вместимость).

5.2.5. ТРЕТЬЯ НОРМАЛЬНАЯ ФОРМА

Рассмотрим транзитивную зависимость следующего типа: если A → B, B -/→ A (B не является ключом), B → C, то A → C.
В этом случае считается, что C транзитивно зависит от A.
Транзитивная зависимость вызвана наличием в отношении двух семантических зависимостей различных типов.
Схема отношения находится в третьей нормальной форме относительно множества функциональных зависимостей F, если она находится в 1НФ и ни один из непервичных атрибутов в R не является транзитивно зависимым от ключа для R.
Схема всей БД находится в ЗНФ относительно F, если каждая схема отношения находится в ЗНФ относительно F.
Рассмотрим схему базы данных примера предыдущего раздела.
ПРЕПОДАВАТЕЛЬ (Таб_Ном_преп, ФИО_преп, Должность);
СТУДЕНТ (Ном_зач_кн, ФИО_студ, Тема_диплома);
КОНСУЛЬТАЦИИ (Таб_Ном_преп, Ном_зач_кн, Дата, Время, Аудитория, Вместимость).
Последнее отношение содержит транзитивную зависимость:
(Таб_Ном_преп, Ном_зач_кн, Дата) → Аудитория →Вместимость.
Следовательно, это отношение не находится в ЗНФ со всеми вытекающими из этого последствиями, и прежде всего следующими:
 если аудитория исключается из расписания консультаций, то о ней вообще теряются сведения;
 если аудитория перестроена и в результате изменилась ее вместимость, то придется просмотреть все кортежи и провести модификацию значений атрибута.

Для устранения транзитивной зависимости необходимо провести декомпозицию последнего отношения, удалив из него транзитивно- зависимый атрибут и поместив его в новое отношение вместе с копией того атрибута, от которого он зависит.
Таким образом, база данных этого примера, лишенная транзитивных зависимостей, в ЗНФ будет выглядеть так:
ПРЕПОДАВАТЕЛЬ (Таб_Ном_преп, ФИО_преп, Должность);
СТУДЕНТ (Ном_зач_кн, ФИО_студ, Тема_диплома);
КОНСУЛЬТАЦИИ (Таб_Ном_преп, Ном_зач_кн, Дата, Время, Аудитория);
АУДИТОРИЯ (Аудитория, Вместимость).
При проектировании структуры реляционной базы данных считается корректной установка, что любая БД должна находиться как минимум в ЗНФ. На практике третья нормальная форма схем отношений достаточна в большинстве случаев, и приведением к третьей нормальной форме процесс проектирования реляционной базы данных обычно заканчивается.

5.2.5. НОРМАЛЬНАЯ ФОРМА БОЙСА — КОДДА

Следует отметить, что определение для ЗНФ было дано Коддом для ситуаций с упрощающим картину допущением того, что отношение имеет только один потенциальный ключ, который, естественно, и является первичным ключом. Естественно, что не все отношения могут быть уложены в данные довольно жесткие рамки. Более обобщающими являются случаи, когда в наличии имеются следующие условия:
 отношение имеет два (или более) потенциальных ключа;
 два потенциальных ключа являются составными;
 два потенциальных ключа перекрываются, т. е. имеют, по крайней мере, один общий атрибут.
Схема отношения R находится в НФБК относительно множества F- зависимостей тогда и только тогда, когда детерминанты являются потенциальными ключами.
Допустим, что при проектировании базы данных **ПОСТАВКИ_ТОВАРОВ** рассматривается отношение:
ПОСТАВКА (Индекс_поставщ, Имя_поставщ, Индекс_товара, Колич_товара).
Допустим также, что значения атрибута **Имя_поставщ** уникальны и могут быть использованы наряду с атрибутом **Индекс_поставщ** для идентификации поставщика. В такой ситуации можно выделить два составных потенциальных ключа:
(Индекс поставщ, Индекс_товара);
(Имя_поставщ, Иидекс_товара).
В рассматриваемом отношении есть два атрибута **Индекс_поставщ** и **Имя_поставщ**, которые идентифицируют один и тот же экземпляр сущности, а, значит, они определяют друг друга. В таком случае отношение содержит два детерминанта, но эти детерминанты не являются потенциальными ключами отношения. Сложившаяся картина противоречит данному выше определению НФБК, и следовательно, данное отношение не находится в НФБК.
Для схемы отношения, не находящейся в НФБК, можно провести декомпозицию в схему БД в НФБК. Из исходного отношения убирается и переносится в новое отношение зависимая часть вместе с копией детерминанта.
Для рассматриваемого примера решение проблемы можно осуществить, разбив исходное отношение на два. Причем поскольку два детерминанта **Индекс_поставщ** и **Имя_поставщ** определяют друг друга, то возможны два равносильных варианта декомпозиции, приводящей к НФБК.
Первый вариант получается, если учитывается зависимость
Индекс_поставщ → Имя_поставщ,
в результате чего имеем следующих два отношения:
ПОСТАВКА (Иидекс_поставщ, Индекс_товара, Колич_товара);
ПОСТАВЩИК (Индекс_поставщ, Имя_поставщ),
Второй вариант исходит из зависимости
Имя_поставщ → Индекс_поставщ,

в результате чего получаем альтернативную группу отношений;
ПОСТАВКА (Имя_поставщ, Индекс_товара, Колич_товара);
ПОСТАВЩИК (Индекс_поставщ, Имя_поставщ).

5.3. ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

Концептуальная модель данных состоит из ряда компонентов: сущностей, связей, атрибутов. При переходе к реляционной схеме базы данных каждый из этих компонентов должен быть проанализирован и, если это окажется необходимым, то даже и преобразован. Изменения, вносимые в процессе преобразования, должны быть такими, чтобы их результат полностью отвечал требованиям, выдвигаемым реляционной моделью данных.
Таким образом, фаза логического проектирования предполагает следующие действия:
 преобразование концептуальной модели данных в логическую модель, в результате которого будет определена схема реляционной модели данных;
 проверка модели с помощью концепций последовательной нормализации;
 проверка поддержки целостности данных.
Рассмотрим последовательно каждое действие.

5.2.1. ПРЕОБРАЗОВАНИЕ СУЩНОСТЕЙ И АТРИБУТОВ

Общий подход к преобразованию сущностей концептуальной модели ПрО в отношения реляционной базы данных состоит в следующем:
 построить набор предварительных отношений и указать первичные ключи для каждого отношения;
 подготовить список всех представляющих интерес атрибутов (тех из них, которые не были перечислены в диаграмме в качестве первичных ключей сущностей) и назначить каждый из этих атрибутов одному из предварительных отношений с тем условием, чтобы эти отношения находились в НФБК. На этом шаге для каждого отношения должны быть определены межатрибутные функциональные зависимости, с помощью которых проверяется соответствие отношений НФБК. Если полученные отношения в итоге не находятся в НФБК или если некоторым атрибутам не находится логически обоснованных мест в предварительных отношениях, то в этих случаях диаграммы необходимо пересмотреть.

5.2.2. ПРЕОБРАЗОВАНИЕ БИНАРНЫХ СВЯЗЕЙ

Каждая сущность преобразуется в определенное отношение, а значит, связь между сущностями преобразуется в связь между отношениями.
Напомним, что связи между отношениями в реляционной модели данных реализуются посредством механизма первичных и внешних ключей. Чтобы этот механизм действовал, необходимо в первую очередь определиться с тем, которое из двух отношений является родительским, а которое — дочерним. Родительским считается такое отношение, которое передает копию набора значений своего первичного ключа другому отношению, где этот набор значений будет представлять внешний ключ. Последнее отношение в этом случае будет являться дочерним отношением.
Рассмотрим бинарную связь **ЧИТАЕТ** между сущностями **ПРЕПОДАВАТЕЛЬ** и **КУРС** (рис. 5.1).

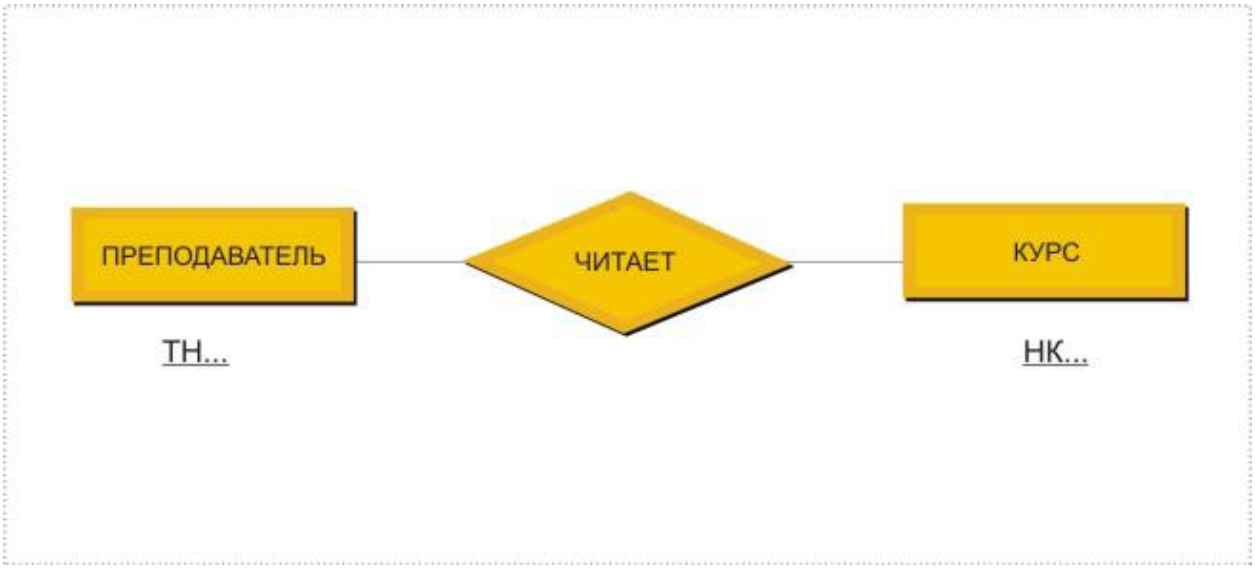

Рис. 5.1. Диаграмма бинарной связи

Как известно, эта связь может быть изображена с помощью диаграммы, которая содержит всю информацию, необходимую для генерации соответствующих отношений РБД.

Сущности **ПРЕПОДАВАТЕЛЬ** и **КУРС** однозначно идентифицируются с помощью **ТН** — табельного номера преподавателя и **НК** — номера курса.

Напомним, что, если все элементы данной сущности должны участвовать в связи, то такое участие называется обязательным. Например, если каждый преподаватель должен читать один какой-либо курс, то класс принадлежности такой сущности — обязательный. Класс принадлежности сущностей, а также мощность связи между сущностями являются факторами, определяющими структуру проектируемой БД.

5.2.3. ПРЕДВАРИТЕЛЬНЫЕ ОТНОШЕНИЯ ДЛЯ БИНАРНЫХ СВЯЗЕЙ ТИПА 1:1

Предварительные отношения для бинарных связей с показателем кардинальности, равным 1 : 1 (рис. 5.2) могут быть получены вследствие просмотра нескольких логических альтернатив и выбора из них наиболее подходящей.

Пусть в проектируемой БД должна храниться следующая информация:

ТН — номер преподавателя;

Ф_И_О — фамилия, имя, отчество преподавателя;

Тел_П — телефон преподавателя;

НК — номер курса;

Назв_К — название курса.

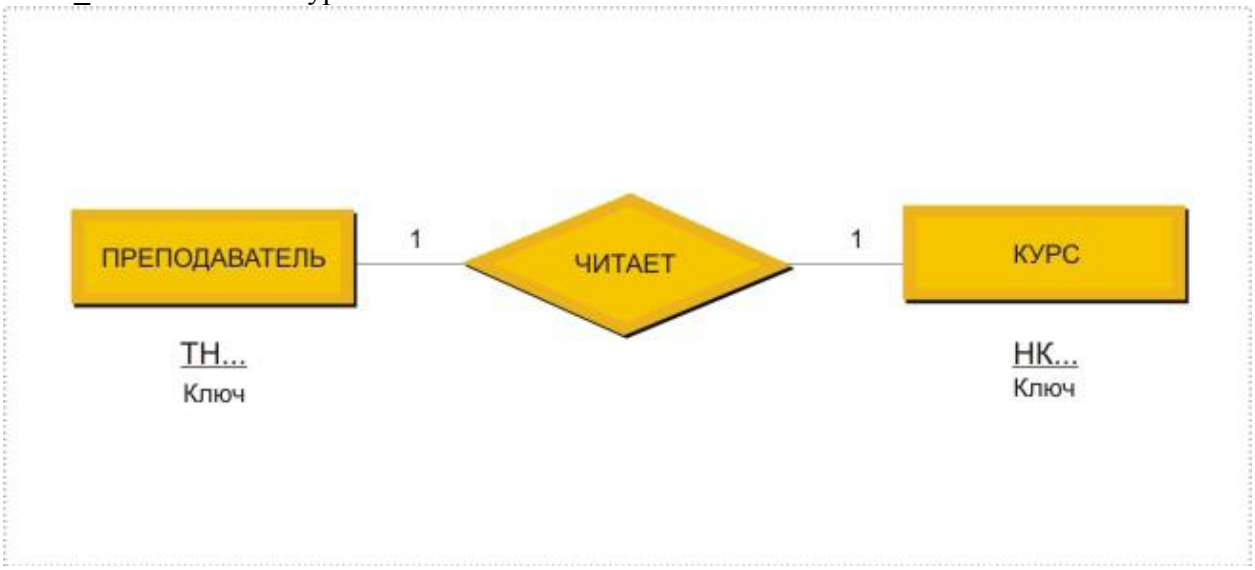

Рис. 5.2. Диаграмма бинарной связи типа 1 : 1

Считая, что класс принадлежности является обязательным для обеих сущностей, получаем отношение:

ПРЕПОДАВАТЕЛЬ (ТН, Ф_И_О, Тел_П, НК, Назв_К).
Первичным ключом этого отношения может быть ключ любой из двух сущностей. Ситуация будет другая, если класс принадлежности одной из сущностей (**ПРЕПОДАВАТЕЛЬ**) — обязательный, второго (**КУРС**) — нет.
В такой ситуации требуется построение двух отношений: по одному под каждую сущность. При этом ключ каждой сущности должен служить первичным ключом для соответствующего отношения. Сущность, для которой класс принадлежности является необязательным, преобразуется в родительское отношение, а сущность, участвующая в связи с обязательным классом принадлежности, преобразуется в дочернее отношение.
Для связи полученных отношений ключ родительского отношения добавляется в качестве атрибута (внешнего ключа) в дочернее отношение. В результате получаем следующую реляционную схему:
ПРЕПОДАВАТЕЛЬ (ТН, Ф_И_О, Тел_П, НК);
КУРС (НК, Назв_К).
Если класс принадлежности для обеих сущностей — необязательный, то лучшим решением является определение трех отношений — по одному для каждой сущности и одного для связи:
ПРЕПОДАВАТЕЛЬ (ТН, Ф_И_О, Тел_П);
КУРС (НК, Назв_К);
ЧИТАЕТ (ТН, НК)
Отношение **ПРЕПОДАВАТЕЛЬ** содержит сведения обо всех преподавателях, а отношение **КУРС** — обо всех курсах.
Отношение для связи должно иметь среди своих атрибутов по одному ключу от каждой сущности.

5.2.4. ПРЕДВАРИТЕЛЬНЫЕ ОТНОШЕНИЯ ДЛЯ БИНАРНЫХ СВЯЗЕЙ ТИПА 1: N

Пусть в рассмотренной выше концептуальной модели существует бинарная связь типа 1: N (рис. 5.3).
Для таких связей существует только два правила. Вариант определяется классом принадлежности N-связной сущности, класс принадлежности 1-связной сущности не влияет на конечный результат в обоих случаях.

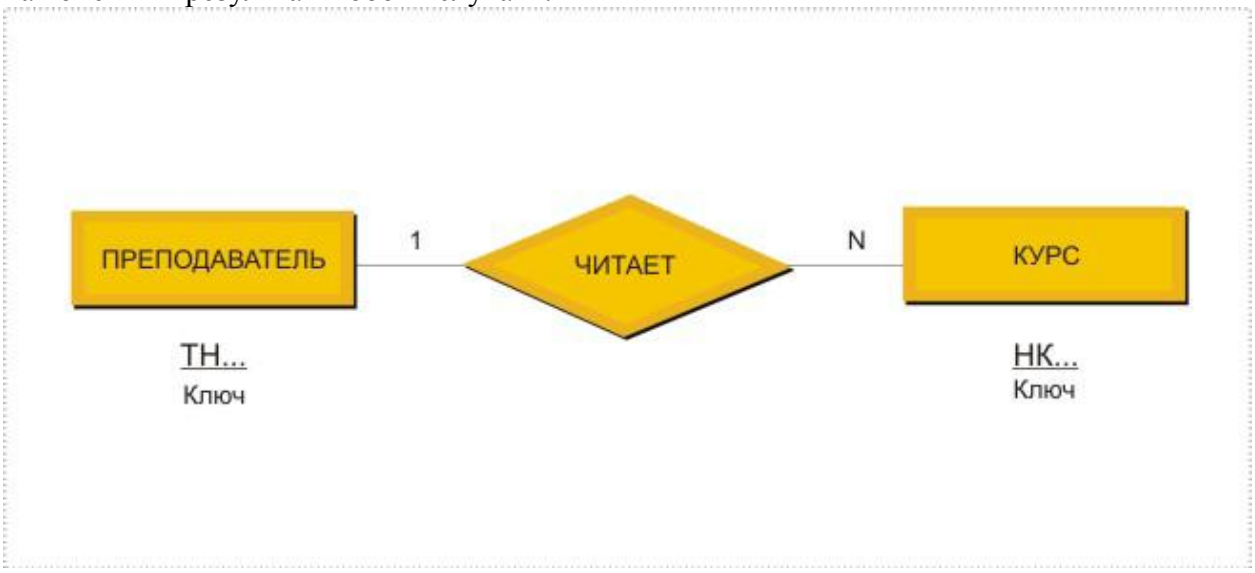

Рис. 5.3. Диаграмма бинарной связи типа 1 : N
Первый вариант. **ПРЕПОДАВАТЕЛЬ** — класс принадлежности необязательный, **КУРС** — обязательный.
Решение задачи становится возможным, если использовать два отношения, по одному на каждую сущность, при условии, что ключ каждой сущности служит в качестве первичного ключа для соответствующего отношения. В качестве родительского назначается отношение, соответствующее "единичной" стороне связи, а в качестве дочернего назначается отношение, представляющее "множественную" сторону связи. Для представления этой связи необходимо

скопировать первичный ключ родительского отношения в дочернее отношение, где данный ключ должен быть описан как внешний. Окончательно в БД войдут два отношения:

ПРЕПОДАВАТЕЛЬ (ТН, Ф_И_О, Тел_П);
КУРС (НК, Назв_К, ТН).

Второй вариант. Класс принадлежности для обеих сущностей — необязательный. Решение — в формировании трех отношений: по одному на каждую сущность (причем ключ каждой сущности служит первичным ключом соответствующего отношения) и одного отношении для связи. Отношение для связи должно иметь среди своих атрибутов ключ каждой сущности:

КУРС (НК, Назв_К);
ПРЕПОДАВАТЕЛЬ (ТН, Ф_И_О, Тел_П);
ЧИТАЕТ (ТН, НК)__

5.2.5. ПРЕДВАРИТЕЛЬНЫЕ ОТНОШЕНИЯ ДЛЯ БИНАРНЫХ СВЯЗЕЙ ТИПА M: N

При такой степени связи требуется три отношения вне зависимости от класса принадлежности обеих сущностей: по одному для каждой сущности, причем ключ каждой сущности используется в качестве первичного ключа соответствующего отношения, и одного отношения для связи. Последнее должно иметь в числе своих атрибутов ключ каждой сущности. Единственно допустимый вариант в сложившейся ситуации — представить БД тремя отношениями:

КУРС (НК, Назв_К);
ПРЕПОДАВАТЕЛЬ (ТН, Ф_И_О, Тел_П);
ЧИТАЕТ (ТН, НК).

Вся информация о курсе будет содержаться в отношении **КУРС**, информация о преподавателе — в отношении **ПРЕПОДАВАТЕЛЬ**, а отношение **ЧИТАЕТ** будет содержать только экземпляры связи, имеющиеся в модели.

Использование прямого преобразования концептуальных связей в логические структуры оказывается очень полезным при моделировании составных сущностей.

6. ЯЗЫК SQL

6.1. ОПЕРАТОР ВЫБОРА SELECT. ФОРМИРОВАНИЕ ЗАПРОСОВ К БАЗЕ данных

Назначение оператора SELECT СОСТОИТ В выборке и отображении данных одной или нескольких таблиц БД.

Синтаксис оператора SELECT:

SELECT [DISTINCT| ALL] {* | [<СПИСОК СТОЛБЦОВ>]}
FROM <СПИСОК ТАБЛИЦ>
[WHERE <предикат-условие выборки или соединения;>]
[GROUP BY <список полей результата>]
[HAVING <предикат-условие для группы>]
[ORDER BY <список полей, по которым требуется упорядочить ВЫВОД>]

Поясним каждую фразу данного оператора.

Фраза SELECT:

• наличие ключевого слова ALL (по умолчанию) означает, что в результирующую таблицу включаются все строки, удовлетворяющие условиям запроса, что может привести к появлению в результирующей таблице одинаковых строк;

• ключевое слово DISTINCT предназначено для приведения таблицы в соответствие с принципами теории отношений, где предполагается отсутствие дубликатов строк;

• символ "*" определяет очень часто встречаемую ситуацию, когда в результирующий набор включаются все столбцы из исходной таблицы запроса.

Во фразе FROM задается перечень исходных таблиц запроса.

Во фразе WHERE определяются условия отбора строк результата или условия соединения строк исходных таблиц, подобно операции условного соединения в реляционной алгебре. В качестве условий отбора могут быть использованы следующие операторы:

• сравнения " = , <>, >, <, >=, <=" — для сравнения результатов вычисления двух выражений; более сложные выражения строятся с помощью логических операторов AND, OR, NOT;

• BETWEEN A AND B — предикат истинен, когда вычисленное значение выражения попадает в заданный диапазон;

• IN — предикат истинен тогда, когда сравниваемое значение входит в множество заданных значений;

• LIKE и NOT LIKE — предикаты, смысл которых противоположен, требуют задания шаблона, с которым сравнивается заданное значение;

• IS NULL — предикат, применяющийся для выявления равенства значения некоторого атрибута неопределенному значению:

• EXIST и NOT EXIST, используются во встроенных подзапросах.

Во фразе GROUP BY задается список полей группировки.

Во фразе HAVING задаются предикаты-условия, накладываемые на каждую группу.

Во фразе ORDER BY задается список полей упорядочения результата, то есть список полей, который определяет порядок сортировки в результирующей таблице.

6.1.1. ПРОСТЫЕ ЗАПРОСЫ

Рассмотрим ряд простых запросов.

Запрос 1

Вывести номера телефонов кафедр университета.

Результат такого запроса должен содержать только два столбца:

Name_kaf и Nom_telef, поэтому сам запрос должен выглядеть следующим образом:

SELECT Name_kaf, Nom_telef
FROM kafedra;

Результирующая таблица приведена ниже:

Name_kaf	Nom_telef
Физики	23-34-24
Общей математики	23-65-43
Истории	23-78-72
Графики	23-99-77
Прикладной математики	23-66-62

Запрос 2

Вывести сведения о кафедре Графики. Запрос будет выглядеть следующим образом:

```
SELECT *
FROM kafedra
WHERE Name_kaf = 'Графики';
```

Ответ на такой запрос будет содержать только одну строку:

Kod_kaf	Name_kaf	Nom_telef	Nom_Auditoria	Col_sotr	Zav_kaf
004	Графики	23-99-77	385	18	Фирсов С.С.

Запрос 3

Вывести сведения о кафедрах университета, находящихся па первом этаже, учитывая тот факт, что номера аудиторий первого этажа лежат в диапазоне от 1 до 96.

Запрос будет выглядеть следующим образом:

```
SELECT *
FROM kafedra
WHERE Nom_Auditoria BETWEEN 1 AMD 99;
```

Результат запроса:

Kod_kaf	Name_kaf	Nomtetef	Nom_Auditoria	Col_sotr	Zavjcaf
002	Общей математики	23-65-43	003	22	Махов
005	Прикладной	23-66-62	028	24	Ляхова

Запрос 4

Вывести сведения о кафедрах университета в виде, отсортированном по столбцу Name_kaf в порядке возрастания.

Запрос будет выглядеть следующим образом:

```
SELECT *
FROM kafedra
ORDER BY Name_kaf ASC;
```

Результат данного запроса:

Kod_kaf	Name_kaf	Nom_telef	Nom_Auditoria	Col_sotr	Zav__kal
004	Графики	23-99-77	385	18	Фирсов
003	Истории	23-78-72	465	16	Росс
002	Общей ма	23-65-43	003	22	Махов
005	Прикладной	23-66-62	028	24	Ляхова
001	Физики	23-34-24	132	25	Иванов Т.М.

6.1.2. АГРЕГАТНЫЕ ФУНКЦИИ ЯЗЫКА

В стандарте языка SQL определено несколько агрегатных 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.

В качестве операнда данных функций может использоваться наименование только одного столбца, и все они возвращают единственное значение. С функциями SUM и AVG могут использоваться только числовые поля. С функциями COUNT, MAX и MIN могут использоваться как числовые, так и символьные поля. При вызове всех перечисленных выше функций, кроме функции COUNT (*), осуществляется исключение всех пустых значений» только после этого операция применяется к оставшимся значениям столбца. Функция COUNT (*) призвана осуществлять подсчет всех строк таблицы независимо от того, какие значения в них находятся.

Запрос 5

Подсчитать и вывести общее число кафедр университета. Запрос будет выглядеть следующим образом:

```
SELECT COUNT (*) AS count
FROM kafedra;
```

Ответ на данный запрос будет выглядеть:

count
5

Запрос 6

Определить среднее число сотрудников, работающих на кафедрах университета.

Запрос будет выглядеть следующим образом:

```
SELECT AVG(Col_sotr) AS avg
FROM kafedra;
```

Ответ на запрос:

avg
21

6.1.3. ГРУППИРОВАНИЕ РЕЗУЛЬТАТОВ

Часто встречаются ситуации, когда в отчет необходимо поместить и промежуточные результаты, опирающиеся на вычисления обобщенных групповых значений. Для применения агрегатных функций в подобных случаях предполагается предварительная операция группировки. Суть операции группировки состоит в том, что все множество строк таблицы разбивается на группы, в каждой из которых собираются строки, имеющие одинаковые значения атрибутов, которые заданы в списке группировки. Обработка такой информации реализуется путем применения агрегатных функций уже к каждой отдельной группе и выдаче полученных итогов.

В языке SQL для осуществления операции группировки в оператор SELECT включается фраза GROUP BY. Запрос, в котором присутствует фраза GROUP BY, называется группирующим запросом, а столбцы, перечисленные в этой фразе, называются группирующими столбцами.

В дальнейшем в качестве примера будем работать с двумя БД: НИР и Сессия.

БД НИР состоит из одной таблицы, в которой хранится информация о производимых выплатах специалистам за проделанную работу по определенным этапам НИР: R= (**ФИО, Этап, Начисления**).

Пусть таблица содержит следующие данные.

r

ФИО	Этап	Начисления (руб)
Семенов Т.Т.	Этап 1	1000
Просов С.М.	Этап 1	2000
Мехова И.И.	Этап 1	500
Семенов Т.Т.	Этап 2	500
Просов С.М.	Этап 2	500
Мехова И.И.	Этап 2	1000
Просов С.М.	Этап 3	1000
Мехова И.И.	Этап 3	1000
Чемцов Я.Ю.	Этап 3	2000
Чемцов Я.Ю.	Этап 4	2000
Яров И.М.	Этап 4	3000

БД Сессия включает в себя сводную таблицу, где представлены экзаменационные оценки студентов, полученные ими в сессию по определенным дисциплинам:

S = (**ФИО, Дисциплина, Оценка**);

s

ФИО	Дисциплина	Оценка
Мур С.М.	Физика	4
Цуканов Т.Т.	Физика	5
Думская М.Т.	Физика	3
Дрозд Г.Р.	Физика	4
Мур С.М.	История	4
Цуканов Т.Т.	История	5
Думская М.Т.	История	3
Цуканов Т.Т.	Математика	5
Думская М.Т.	Математика	4
Дрозд Г.Р.	Математика	5
Петрова С.О.	Электротехника	5
Часов И.И.	Электротехника	4
Иванова Я.С.	Электротехника	5
Крисс Р.О.	Электротехника	3
Часов И.И.	Иностр. язык	5
Иванова Я.С.	Иностр. язык	4
Часов И.И.	Экономика	4
Иванова Я.С.	Экономика	4
Крисс Р.О.	Экономика	5
Фирсова Л.Р.	Экономика	3

Сформируем к базам данных несколько запросов.

Запрос 7

БД НИР. Для каждого специалиста определить сумму, выплаченную за работу по данному этапу, и количество сделанных ему выплат.

Для формирования запроса включим в предложение SELECT следующую информацию: **ФИО, COUNT (Начисления) AS count, SUM (Начисления) AS sum,** где в качестве имен для двух вычисляемых столбцов используются псевдонимы. Группировку будем производить по столбцу **ФИО**. Для того чтобы проще было просматривать результаты, выводимые данные представим в отсортированном по столбцу **ФИО** виде

```
SELECT ФИО, COUNT (Начисления) AS count, SUM (Начисления) AS sum
FROM r
GROUP BY ФИО
ORDER BY ФИО;
```

Результат запроса:

ФИО	count	sum
Мехова И.И.	3	2500
Просов С.М.	3	3500
Семенова Т.Т.	2	1500
Чемцов Я.Ю.	2	4000
Яров И.М.	1	3000

Запрос 8

БД Сессия. Для каждой дисциплины определить количество студентов, сдавших экзамен.

Запрос будет выглядеть следующим образом:

```
SELECT Дисциплина, COUNT (*) AS count
FROM s
GROUP BY Дисциплина
ORDER BY Дисциплина;
```

Результат запроса:

Дисциплина	count
Иностр.язык	2
История	3
Математика	3
Физика	4
Экономика	4
Электротехника	4

Запрос 9

БД НИР. В условиях предыдущего запроса вывести информацию, касающуюся только тех специалистов, которым производились начисления более одного раза.

Для вывода такой информации в текст предыдущего запроса необходимо добавить фразу HAVING COUNT (Начисления! > 1. И в этом случае весь запрос примет вид

```
SELECT ФИО, count (Начисления) AS count, SUM (Начисления)
AS sum
FROM r
GROUP BY ФИО
HAVING COUNT(Начисления) > 1
ORDER BY ФИО;
```

Результаты выполнения запроса представлены ниже.

ФИО	count	sum
Мехова И.И.	3	2500
Просов С.М.	3	3500
Семенов Т.Т.	2	1500
Чемцов Я.Ю.	2	4000

6.1.4. ВЛОЖЕННЫЕ ЗАПРОСЫ

Стандарт языка позволяет в тело одного оператора SELECT внедрять другой оператор SELECT. Если внутренний оператор запроса помешен в предложения WHERE или HAVING внешнего оператора SELECT, то создается ситуация вложенных запросов (подзапросов).

Запрос 10

БД НИР. Вывести список платежей, где величина единовременных начислении превысила среднее значение.

Запрос будет выглядеть следующим образом:

```
SELECT ФИО, Этап, Начисления
FROM r
WHERE Начисления > (SELECT avg(Начисления) FROM r);
```

Результат запроса:

ФИО	Этап	Начисления (руб)
Просов С. М.	Этап 1	2000
Чемцов Я.Ю.	Этап 3	2000
Чемцов Я.Ю.	Этап 4	2000
Яров И.М.	Этап 4	3200

6.1.5. Многотабличные запросы

При работе с базами данных потребности пользователей не ограничиваются только реализацией простых запросов данных из одной таблицы. Во многих случаях для получения ответа на запрос необходимо объединить информацию из нескольких исходных таблиц. Для того чтобы осуществить такое объединение в результирующей таблице, необходимо выполнить операцию соединения, при которой объединение информации из двух таблиц происходит посредством образования пар связанных строк, выбранных из каждой таблицы. Таблицам можно присвоить имена- псевдонимы, что бывает полезно для осуществления операции соединения таблицы с самой собою и в ряде других ситуаций.

Если в операторе SELECT указано более одного имени таблицы, неявно подразумевается, что над перечисленными таблицами осуществляется операции декартова произведения. Самый простой запрос SELECT такого рода без необязательных частей выглядит следующим

образом:

```
SELECT *
FROM rl, r2;
```

и соответствует декартову произведению таблиц r1 и r2.

Выражение

```
SELECT rl.A, r2.B
```

FROM rl, r2;
соответствует проекции декартова произведения двух таблиц на два столбца A из таблицы r1 и B из таблицы r2.
Рассмотрим базу данных, в которой хранится информация о производимых выплатах специалистам за проделанную работу по определенным этапам НИР. Пусть она состоит из трех отношений r1, r2 и r3. Будем считать, что они представлены таблицами r1, r2 и r3 соответственно.
R1 = **(ФИО, Отдел);**
R2= **(Отдел, Этап);**
R3= **(ФИО, Этап, Начисления).**

r1

ФИО	Отдел
Семенов Т.Т.	03
Просов СМ.	03
Мехова И.И.	03
Чемцов Я.Ю.	04
Яров И.М.	04

r2

Отдел	Этап
03	Этап 1
03	Этап 2
03	Этап 3
04	Этап 3
04	Этап 4

r3

ФИО	Этап	Начисления (руб)
Семенов Т. Т.	Этап 1	1000
Просов СМ.	Этап 1	2000
Мехова И.И.	Этап 1	500
Семенов Т. Т.	Этап 2	500
Просов СМ.	Этап 2	500
Мехова И.И.	Этап 2	1000
Просов СМ.	Этап 3	1000
Мехова И.И.	Этап 3	1000
Чемцов Я.Ю.	Этап 3	2000
Чемцов Я.Ю.	Этап 4	2000
Яров И.М.	Этап 4	3000

Для запросов будет использоваться и расширенная база данных Сессия, представленная таблицами со схемами:
S1 = (**ФИО, Дисциплина, Оценка**) — содержащей сведения о результатах сессии;
S2= (**ФИО, Группа**) — содержащей сведения о составе групп;
S3 = (**Группа, Дисциплина**) — содержащей перечень экзаменов, подлежащих сдаче.

s1

ФИО	Дисциплина	Оценка
МурС.М.	Физика	4
Цуканов Т.Т.	Физика	5
Думская М.Т.	Физика	3
Дрозд Г.Р.	Физика	4
МурС.М.	История	4
Цуканов Т.Т.	История	5
Думская М.Т.	История	3
Цуканов Т.Т.	Математика	5
Думская М.Т.	Математика	4
Дрозд Г.Р.	Математика "	5
Петрова СО.	Экономика	5
Часов И.И.	Электротехника	4
Иванова Я. С.	Электротехника	5
Крисе Р.О.	Электротехника	3
Часов И.И.	Иностр_язык	5
Иванова Я.С.	Иностр_язьгк	4
Часов И.И.	Экономика	4
Иванова Я.С.	Экономика	4
Крисе Р.О.	Экономика	5
Фирсова Л.Р.	Экономика	3

s2

ФИО	Группа
МурС.М.	02-КТ-21
Цуканов Т.Т.	02-КТ-21
Думская М.Т.	02-КТ-21
Дрозд Г.Р.	02-КТ-21
Петров С.О.	02-КТ-12
Часв И.И.	02-КТ-12
Иванова Я.С.	02-КТ-12
Крисс Р.О.	02-КТ-12
Фирсова Л.Р.	02-КТ-12

s3

Группа	Дисциплина
02-КТ-21	Физика
02-КТ-21	История
02-КТ-21	Математика
02-КТ-12	Экономика
02-КТ-12	Электротехника
02-КТ-12	Иностр. язык

Запрос 11

БД НИР. Вывести список сотрудников отдела 03, которые участвовали в выполнении Этапа_3.

Запрос будет выглядеть следующим образом:

```
SELECT r3.ФИО, r3.Этап
FROM rl, r3
WHERE rl.Отдел = '03' AND
rl.ФИО = r3.ФИО AND
r.Этап = 'Этап_3';
```

Результат запроса:

ФИО	Этап
ПросовС.М.	Этап_3
Мехова И.И.	Этап_3

Запрос 12

Вывести группы, в которых по одной дисциплине на экзаменах получено больше одной пятерки.

Запрос будет выглядеть следующим образом:

```
SELECT s2.Группа
FROM s1, s2
```

WHERE s1.ФИО = s2.ФИО AND
s1.Оценка = 5
GROUP BY s2.Группа , s1.Дисциплина
HAVING count (*)> 1;
Результатом выполнения раздела HAVING является сгруппированная таблица, содержащая только те группы строк, для которых результат вычисления условия поиска есть TRUE.
Результат запроса:

Группа
02-КТ-21
02-КТ-12

Дадим пример запроса, где будет использован предикат NOT EXISTS. Его возможности удобно проиллюстрировать в тексте многотабличного запроса.
Запрос 13
Вывести список тех студентов, кто должен был сдавать экзамен по истории, но пока еще не сдавал,
Запрос будет выглядеть следующим образом:
SELECT ФИО
FROM s2,S3
WHERE s2.Группа=s3.Группа AND
Дисциплина = 'История' AND NOT EXISTS (SELECT ФИО
FROM SI
WHERE ФИО = a.ФИО AND
Дисциплина = 'История');
Результат запроса:

ФИО
Дрозд Г. Р.

Напомним, что предикат EXISTS истинен, когда подзапрос не пуст, то есть содержит хотя бы один кортеж, в противном случае предикат EXISTS ложен. Предикат ют EXISTS — истинен только тогда, когда подзапрос пуст. Обработка такого запроса состоит в том, что для каждого студента, обучающегося в группе, студентам которой необходимо сдавать экзамен по истории, проверяется истинность предиката NOT EXISTS. ИСТИННОСТЬ его устанавливается по факту присутствия во внутреннем запросе значений. Если подзапрос пуст, то данный студент еще не сдавал экзамен по истории, предикат кот EXISTS имеет значение TRUE, И ФИО студента помешается в результирующую таблицу вывода.

6.2. ОПЕРАТОРЫ МАНИПУЛИРОВАНИЯ ДАННЫМИ
6.2.1. ОПЕРАТОР ВВОДА ДАННЫХ INSERT
Оператор ввода данных INSERT имеет следующий синтаксис:
INSERT INTO имя таблицы [(<список столбцов>)] VALUES (<список значений>)
Подобный синтаксис позволяет ввести только одну строку в таблицу. Например, введем нового студента в таблицу s2 БД Сессия:
INSERT INTO s2 (ФИО, Группа) VALUES ('Сидоров П.П.', '02-КТ-21');
Задание списка столбцов необязательно тогда, когда, как в данном случае, вводится строка с заданием значений всех столбцов. При таком вводе предполагается, что информация будет вводиться в том порядке, в котором они описаны в операторе CREATE TABLE. Так как в рассматриваемом примере вводится полная строка, то можно не задавать список столбцов, ограничиться только заданием перечня значений, в этом случае оператор ввода будет выглядеть следующим образом:
INSERT INTO S2 VALUES ('Сидоров П.П.', '02-КТ-21');
Между списком имен столбцов и списком значений должно быть следующее соответствие:
• количество элементов в обоих списках должно быть одинаковым;

• между положением элементов в списках должно быть строгое соответствие, которое определяется слева направо: первый элемент одного списка соответствует первому элементу второго списка и т. д.;
• типы данных соответствующих элементов списков должны быть одинаковые и принадлежать к одному и тому же домену.

6.2.2. ОПЕРАТОР УДАЛЕНИЯ ДАННЫХ DELETE

Оператор удаления данных позволяет удалить одну или несколько строк из таблицы в соответствии с условиями, которые задаются для удаляемых строк. Синтаксис оператора DELETE следующий:

```
DELETE FROM имя_таблицы
[WHERE условия_Отбора]
```

Условия отбора определяют, какие строки должны быть удалены. Если условия отбора не задаются, то из таблицы удаляются все существующие в ней строки. Однако это не означает, что удаляется вся таблица. Исходная таблица остается, но она остается пустой, незаполненной.

Например, если нам надо удалить результаты прошедшей сессии, то мы можем удалить все строки из отношения s1 следующим оператором:

```
DELETE FROM s1;
```

Условия отбора в части WHERE имеют тот же вид, что и условия фильтрации в операторе SELECT. Эти условия определяют, какие строки из исходного отношения будут удалены. Например, исключение по какой- либо причине студента Крисса Р.О. из таблицы s2 можно выполнить оператором:

```
DELETE FROM s2
WHERE ФИО = 'Крисс Р.О.';
```

6.2.3. ОПЕРАЦИЯ ОБНОВЛЕНИЯ ДАННЫХ UPDATE

Операция обновления данных UPDATE требуется тогда, когда требуется изменить содержимое базы данных. Данный оператор, также как и другие операторы обновления информации БД, применяется к одной конкретной таблице и имеет следующий формат:

```
UPDATE имя_таблицы
SET имя_столбца1 = новое_значение1 [,имя__стол0ца2 =
новое_значение2...]
[WHERE условие_отбора]
```

Здесь в предложении UPDATE указывается имя обновляемой таблицы, в предложении SET указываются имена столбцов и новые данные. Новые данные должны быть совместимы с теми данными, которые они призваны заменить.

Часть WHERE является необязательной, также как и в операторе DELETE. Она играет здесь ту же роль, что и в операторе DELETE, — позволяет отобрать строки, к которым будет применена операция модификации. Если условие отбора не задается, то операция модификации будет применена ко всем строкам таблицы.

Рассмотрим операцию обновления данных таблицы базы данных НИР. Предположим, что решено все начисления специалистам увеличить на 10%. Операция обновления информации в связи с этим будет выглядеть следующим образом:

```
UPDATE r3
SET Начисления = Начисления * 1.1;
```

В том случае, когда модификацию информации необходимо производить выборочно, требуется использование предложения WHERE. Допустим, что в БД Сессия следует произвести изменение данных, поскольку студентка Думская М.Т. пересдала экзамен по физике и получила оценку "отлично" вместо "удовлетворительно". Для решения поставленной задачи необходимо выполнить следующую операцию:

```
UPDATE sI SET s1.Оценка = 5
WHERE sI.ФИО = 'Думская М.Т.' AND
s1.Дисциплина = 'ФИЗИКА';
```

6.3. ОПЕРАТОРЫ ОПРЕДЕЛЕНИЯ ДАННЫХ

6.3.1. СОЗДАНИЕ ТАБЛИЦ

Создание таблицы осуществляется посредством оператора CREATE TABLE. ЕГО упрошенная версия выглядит следующим образом:

CREATE TABLE Имя_таблииы
(Имя_столбца Тип_данных [NULL | NOT NULL] [,...]}

Оператор такого вида приведет к созданию таблицы с именем <Имя_таблицы>, которая будет содержать столько столбцов, сколько их задано в операторе. При определении столбца необходимо задать его имя, тип данных, к которому будут относиться значения этого столбца, а также определить, можно ли в качестве значения рассматриваемого столбца использовать ключевое слово NULL. Ключевым словом NULL помечается такой столбец, который может содержать неопределенные значения. Определения столбцов первичных ключей отношений всегда должны
содержать ключевые слова NOT NULL.

Для того чтобы создать таблицу s1 БД СЕССИЯ, необходимо использовать оператор вида

CREATE TABLE s1 (
ФИО VARCHAR (20) NOT NULL,
Дисциплина VARCHAR (20) NOT NULL,
Оценка SMALLINT NOT NULL);

Полное описание оператора CREATE TABLE ДОЛЖНО включать средства поддержки целостности данных. Такие средства представляют собой спецификаторы, позволяющие задать ограничения для предотвращения попыток нарушить согласованность данных. Базовое определение оператора CREATE TABLE имеет следующий формат:

CREATE TABLE имя_таблицы
({ имя_столбца тип_даных [NOT NULL] [UNIQUE]
[DEFAULT значение по умолчанию]
[CHECK (условие проверки на допустимость) [,...] }
[PRIMARY KEY (список столбцов),]
{[UNIQUE (список столбцов),] [,...]}
{[FORING KEY {список столбцов внешних ключей)
REFERENCES имя родительской таблицы [(список столбцов ключей-кандидатов)],
[MATCH {PARTIAL | FULL}
[ON UPDATE правило ссылочной целостности]
[ON DELETE правило ссылочной целостности]] [,...]}
{[CHECK (условие проверки на допустимость)] [,...]})

Перепишем оператор создания таблицы s1 БД Сессия следующим образом:

CREATE TABLE sI (
ФИО VARCHAB (20) NOT NULL,
Дисциплина VARCHAR (20) NOT NULL,
Оценка SMALLINT NOT NULL);
PRIMARY KEY (ФИО, Дисциплина),
FORING KEY ФИО REFERENCES S2
ON UPDATE CASCADE
ON DELETE CASCADE);

Учитывая то, что операторы языка SQL транслируются в режиме интерпретации, создавать таблицы необходимо в определенном порядке: вначале родительские, а затем дочерние. В противном случае появятся сообщения об ошибке в том случае, когда в определении дочерней таблицы будут присутствовать ссылки на еще не существующую родительскую таблицу.

6.3.2. ОБНОВЛЕНИЕ ТАБЛИЦ

В уже созданную таблицу изменения могут быть внесены с помощью оператора ALTER TABLE, который имеет следующий обобщенный формат:

ALTER TABLE имя_таблицы
[ADD COLUMN] имя_столбца тип_данных [NOT NULL] [UNIQUE]

[DEFAULT значение по умолчанию]
[CHECK (условие проверки на допустимость)]]
[DROP [COLUMN] имя_столбца [RISTRICT | CASCADE]]
[ADD CONSTRAINT имя_ ограничения ограничение]
[DROP CONSTRAINT имя ограничения [RISTRICT I CASCADE]]
[ALTER [COLUMN] SET DEFAULT значение по умолчанию]
[ALTER [COLUMN] DROP DEFAULT]
В данном формате предусмотрены возможности для выполнения ряда действий:
• добавить новый столбец в существующую таблицу — ADD COLUMN;
• удалить столбец из существующей таблицы — DROP COLUMN;
• добавить в определение таблицы новое ограничение — ADD CONSTRAINT;
• удалить из определения таблицы существующее ограничение — DROP CONSTRAINT;
• задать для существующего столбца значение по умолчанию — ALTER [COLUMN] SET DEFAULT;
• отменить установленное для столбца значение по умолчанию ALTER [COLUMN] DROP DEFAULT.
Добавить в таблицу s1 столбец Группа, содержащий символьный тип данных, можно с помощью оператора:

```
ALTER TABLE s1
ADD Группа varchar (7) NOT NULL;
```

6.3.2. УДАЛЕНИЕ ТАБЛИЦ

Ставшая ненужной таблица может быть удалена из базы данных оператором

```
DROP TABLE имя таблицы [RISTRICT I CASCADE].
```

Ключевые слова RISTRICT И CASCADE используются для определения условий удаления таблицы в том случае, если в базе данных присутствуют ее дочерние таблицы. Ключевое слово RISTRICT при наличии в базе данных зависимых от удаляемой таблицы объектов вызовет отмену удаления. Ключевое слово CASCADE в этой ситуации вызовет автоматическое удаление всех объектов базы данных, существование которых зависит отданной таблицы.
Удалим таблицу s1:

```
DROP TABLE s1;
```

4.3. Лабораторные работы

№ п/п	*Номер раздела дисциплины*	*Наименование лабораторной работы*	*Объем (час.)*	*Вид занятия в интерактивной, активной, инновационной формах, (час.)*
1	**4.**	Введение в Microsoft Access.	3	-
2	**4.**	Основы работы с таблицами.	3	-
3	**4.**	Работа с запросами.	3	-
4	**5.**	Создание отчетов.	3	-
5	**5.**	Работа с формами.	3	-
6	**5.**	Кнопочные формы.	3	-
7	**3.**	Проектирование базы данных.	3	-
8	**6.**	Создание базы данных при помощи языка SQL.	3	-
9	**6.**	Создание простых запросов при помощи языка SQL.	4	-
10	**6.**	Создание сложных запросов при помощи языка SQL.	4	-
		ИТОГО	**35**	-

4.4. Практические занятия

№ п/п	*Номер раздела дисциплины*	*Наименование практической работы*	*Объем (час.)*	*Вид занятия в интерактивной, активной, инновационной формах, (час.)*
1	**3.**	Основные модели данных.	6	-
2	**5.**	Этапы и методы проектирования баз данных.	6	-
3	**6.**	Основы языка SQL.	5	-
		ИТОГО	**17**	-

4.5. Контрольные мероприятия: не предусмотрены

5. МАТРИЦА СООТНЕСЕНИЯ РАЗДЕЛОВ УЧЕБНОЙ ДИСЦИПЛИНЫ К ФОРМИРУЕМЫМ В НИХ КОМПЕТЕНЦИЯМ И ОЦЕНКЕ РЕЗУЛЬТАТОВ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Компетенции / *№, наименование разделов дисциплины*	*Кол-во часов*	*Компетенции*			*Σ комп.*	$t_{ср}$, *час*	*Вид учебных занятий*	*Оценка результатов*
		ОПК		*ПК*				
		6	*9*	*5*				
1	2	3	4	5	6	7	8	9
1. Введение в базы данных	18	+	+	+	3	6	Лк, СРС	ЗАЧЕТ
2. Архитектура СУБД	18	+	+	+	3	6	Лк, СРС	ЗАЧЕТ
3. Модели данных	34	+	+	+	3	11,3	Лк, ЛР, ПЗ, СРС	ЗАЧЕТ
4. Реляционная модель данных	34	+	+	+	3	11,3	Лк, ЛР, СРС	ЭКЗАМЕН
5. Проектирование базы данных	39	+	+	+	3	13	Лк, ЛР, ПЗ, СРС	ЭКЗАМЕН
6. Язык SQL	37	+	+	+	3	12,3	Лк, ЛР, ПЗ, СРС	ЭКЗАМЕН
всего часов	**180**	**60**	**60**	**60**	**3**	**48**		

6. ПЕРЕЧЕНЬ УЧЕБНО-МЕТОДИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

1. Дейт, К. Дж. Введение в системы баз данных : учебное пособие / К. Дж. Дейт. – 8-е изд. – Москва : Вильямс, 2005. – 1328 с.
2. Хомоненко, А. Д. Microsoft Access. Быстрый старт / А. Д. Хомоненко, В. В. Гридин. – Санкт-Петербург : БХВ- Петербург, 2003. – 304 с.

7. ПЕРЕЧЕНЬ ОСНОВНОЙ И ДОПОЛНИТЕЛЬНОЙ ЛИТЕРАТУРЫ, НЕОБХОДИМОЙ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

№	*Наименование издания*	*Вид заня-тия*	*Количество экземпляров в библиотеке*, шт.	*Обеспечен-ность*, (экз./ чел.)
1	2	3	4	5
Основная литература				
1	Советов, Б. Я. Базы данных: теория и практика : учебник для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 2-е изд. – М. : Юрайт, 2014. – 463 с. – (Бакалавр. Базовый курс).	Лк, кр, ПЗ, ЛР	15	1
2	Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин . – Москва : Юрайт, 2016. – 213 с. – (Бакалавр. Академический курс)	Лк, кр, ПЗ, ЛР	10	0,67
Дополнительная литература				
3	Ульянов, А. Д. Реляционные базы данных в СУБД Microsoft Access : лабораторный практикум / А. Д. Ульянов. – Братск : БрГУ, 2015. – 53 с. – Б. ц.	ЛР	23	1

8. ПЕРЕЧЕНЬ РЕСУРСОВ ИНФОРМАЦИОННО – ТЕЛЕКОММУНИКАЦИОННОЙ СЕТИ «ИНТЕРНЕТ» НЕОБХОДИМЫХ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

1.Электронный каталог библиотеки БрГУ
http://irbis.brstu.ru/CGI/irbis64r_15/cgiirbis_64.exe?LNG=&C21COM=F&I21DBN=BOOK&P21DBN=BOOK&S21CNR=&Z21ID=.
2. Электронная библиотека БрГУ
http://ecat.brstu.ru/catalog .
3. Электронно-библиотечная система «Университетская библиотека online» http://biblioclub.ru .
4. Электронно-библиотечная система «Издательство «Лань»
http://e.lanbook.com .
5. Информационная система "Единое окно доступа к образовательным ресурсам" http://window.edu.ru .
6. Научная электронная библиотека eLIBRARY.RU http://elibrary.ru .
7. Университетская информационная система РОССИЯ (УИС РОССИЯ) https://uisrussia.msu.ru/ .
8. Национальная электронная библиотека НЭБ
http://xn--90ax2c.xn--p1ai/how-to-search /.

9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Даются рекомендации при подготовке к лабораторным и практическим работам.

9.1. Методические указания для обучающихся по выполнению лабораторных работ/практическим работам

Лабораторная работа №1

Введение в СУБД Access

Цель работы:

Получить базовых навыков работы с СУБД Access. Создание базы данных из готового шаблона. Заполнение базы данных. Создание отчетов по готовой базе данных.

Задание:

1. Создать базу данных «Учащиеся» из предложенных шаблонов в СУБД Access.
2. Заполнить информацию, с помощью предложенных форм, как минимум о 10 учениках.
3. Создать все предложенные отчеты.

Порядок выполнения:

Запустить программу Microsoft Access. Выбрать из предложенных шаблонов выбрать «Учащиеся». В созданной базе данных заполнить информацию о 10 учениках. Создать все предложенные отчеты по этой базе данных

Форма отчетности:

Отчет набирается на компьютере и сдается в печатном виде. В отчете должны присутствовать:

1. Номер варианта
2. Цель работы
3. Задание
4. Поэтапное выполнение всех заданий варианта
5. Вывод.

Задания для самостоятельной работы:

Предусмотрены вариантом студента.

Рекомендации по выполнению заданий и подготовке к лабораторной работе

Ознакомиться с теоретическим материалом, представленным в четвертом разделе данной дисциплины.

Основная литература

1. Советов, Б. Я. Базы данных: теория и практика : учебник для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 2-е изд. – М. : Юрайт, 2014. – 463 с. – (Бакалавр. Базовый курс).

Дополнительная литература

1. Ульянов А.Д. Реляционные базы данных в СУБД Microsoft Access : лабораторный практикум. – Братск: Изд-во БрГУ, 2015. – 53 с.

Контрольные вопросы для самопроверки

1. Какие поля являются обязательными к заполнению?
2. Как формируются отчеты?
3. Есть ли несоответствие отчетов и заполненной ранее информации?

Лабораторная работа №2

Основы работы с таблицами

Цель работы:

Получение навыков работы по созданию структуры таблиц, модификации структуры таблиц, заполнению таблиц. Создание ключевых полей, индексированных полей, установка связей между таблицами. Удаление информации из связанных таблиц и восстановление этой

информации.

Задание:

1. Создать структуры таблиц, ключевые и индексные поля.
2. Заполнить таблицы данными.
3. Установить связи .
4. Сделать каскадное удаление данных.

Порядок выполнения:

Запустить программу Microsoft Access. Создать новую базу данных. По варианту выбрать предметную область. Создать и заполнить таблицы. Установить связи между таблицами. Проверить работу каскадного удаления данных

Форма отчетности:

Отчет набирается на компьютере и сдается в печатном виде. В отчете должны присутствовать:

1. Номер варианта
2. Цель работы
3. Задание
4. Поэтапное выполнение всех заданий варианта
5. Вывод.

Задания для самостоятельной работы:

Предусмотрены вариантом студента.

Рекомендации по выполнению заданий и подготовке к лабораторной работе/ семинару/ практическому занятию

Ознакомиться с теоретическим материалом, представленным в четвертом разделе данной дисциплины.

Основная литература

1. Советов, Б. Я. Базы данных: теория и практика : учебник для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 2-е изд. – М. : Юрайт, 2014. – 463 с. – (Бакалавр. Базовый курс).

Дополнительная литература

1. Ульянов А.Д. Реляционные базы данных в СУБД Microsoft Access : лабораторный практикум. – Братск: Изд-во БрГУ, 2015. – 53 с.

Контрольные вопросы для самопроверки

1. Какие типы данных были использованы в работе и почему.
2. Какие бывают типы данных и чем они отличаются друг от друга.
3. Для чего необходимо ключевое поле.
4. В каких случаях необходимо составное ключевое поле.
5. Какие бывают виды связей.
6. От чего зависит вид связи.

Лабораторная работа №3

Работа с запросами.

Цель работы:

Получение навыков работы по созданию запросов.

Задание:

1. Создать запрос на выборку информации из основной таблицы.

2. Создать запрос на выборку информации из связанных таблиц.
3. Создать параметрический запрос.
4. Создать запрос для выбора информации для создания сложного отчета.
5. Создать 2 запроса самостоятельно, используя другие операторы запросов.

Порядок выполнения:
Открыть созданную ранее базу данных. Изучить предложенные операторы запросов. Изучить работу конструктора запроса. По примеру создать запросы на выборку информации, параметрические запросы и т.д. Самостоятельно создать 2 запроса с операторами запроса не представленными в методическом пособии.

Форма отчетности:
Отчет набирается на компьютере и сдается в печатном виде. В отчете должны присутствовать:
1. Номер варианта
2. Цель работы
3. Задание
4. Поэтапное выполнение всех заданий варианта
5. Вывод.

Задания для самостоятельной работы:
Предусмотрены вариантом студента.

Рекомендации по выполнению заданий и подготовке к лабораторной работе
Ознакомиться с теоретическим материалом, представленным в четвертом разделе данной дисциплины.

Основная литература

1. Советов, Б. Я. Базы данных: теория и практика : учебник для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 2-е изд. – М. : Юрайт, 2014. – 463 с. – (Бакалавр. Базовый курс).

Дополнительная литература

1. Ульянов А.Д. Реляционные базы данных в СУБД Microsoft Access : лабораторный практикум. – Братск: Изд-во БрГУ, 2015. – 53 с.

Контрольные вопросы для самопроверки

1. Как формируются запросы в Access?
2. Какие операторы вы использовали в работе?
3. Написать запрос по заданию преподавателя.
4. Как работают сложные запросы?

Лабораторная работа №4
Работа с запросами.

Цель работы:
Получение навыков работы по созданию отчетов.

Задание:
1. Создать простой отчет на основе примера, отображающий результаты обработки информации для прикладной области, выбранной в соответствии с вариантом задания.
2. Создать второй отчет самостоятельной.

Порядок выполнения:
Открыть созданную ранее базу данных. Изучить предложенный пример создания отчета. На

основе этого примера создать отчет для своей базы данных. Самостоятельно создать второй отчет, в качестве примеров можно использовать отчеты из 4 лабораторной работы данной курса.

Форма отчетности:
Отчет набирается на компьютере и сдается в печатном виде. В отчете должны присутствовать:

1. Номер варианта
2. Цель работы
3. Задание
4. Поэтапное выполнение всех заданий варианта
5. Вывод.

Задания для самостоятельной работы:
Предусмотрены вариантом студента.

Рекомендации по выполнению заданий и подготовке к лабораторной работе
Ознакомиться с теоретическим материалом, представленным в пятом разделе данной дисциплины.

Основная литература

1. Советов, Б. Я. Базы данных: теория и практика : учебник для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 2-е изд. – М. : Юрайт, 2014. – 463 с. – (Бакалавр. Базовый курс).

Дополнительная литература

1. Ульянов А.Д. Реляционные базы данных в СУБД Microsoft Access : лабораторный практикум. – Братск: Изд-во БрГУ, 2015. – 53 с.

Контрольные вопросы для самопроверки

1. Как разделить поле заголовка и поле значение?
2. Как перенести одно из полей в другую часть рабочей области?
3. Какие функции можно использовать в отчетах?
4. Рассказать какую информацию предоставляет каждый отчет в вашей работе.

Лабораторная работа №5
Работа с формами

Цель работы:
Получение навыков работы по созданию экранных форм.

Задание:

1. Создать форму для ввода информации в таблицы в удобном для пользователя формате.
2. Создать сложную форму, объединив формы, созданные для разных таблиц.

Порядок выполнения:
Открыть созданную ранее базу данных. Изучить предложенные примеры создания форм. На основе этих примеров создать собственные формы. У одной из созданных форм изменить оформление на свой вкус.

Форма отчетности:
Отчет набирается на компьютере и сдается в печатном виде. В отчете должны присутствовать:

1. Номер варианта
2. Цель работы

3. Задание
4. Поэтапное выполнение всех заданий варианта
5. Вывод.

Задания для самостоятельной работы:
Предусмотрены вариантом студента.

Рекомендации по выполнению заданий и подготовке к лабораторной работе

Ознакомиться с теоретическим материалом, представленным в пятом разделе данной дисциплины.

Основная литература

1. Советов, Б. Я. Базы данных: теория и практика : учебник для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 2-е изд. – М. : Юрайт, 2014. – 463 с. – (Бакалавр. Базовый курс).

Дополнительная литература

1. Ульянов А.Д. Реляционные базы данных в СУБД Microsoft Access : лабораторный практикум. – Братск: Изд-во БрГУ, 2015. – 53 с.

Контрольные вопросы для самопроверки

1. Для чего необходимы формы?
2. Какие элементы являются обязательными в форме?
3. Чем отличаются Объединенные формы и Связанные формы?
4. Каким образом производится заполнения нового элементы таблицы при помощи формы?
5. Как можно отредактировать уже имеющийся элемент таблицы при помощи формы?

Лабораторная работа №6
Работа с кнопочными формами

Цель работы:
Получение навыков работы по созданию кнопочных форм.

Задание:
1. Создать кнопочную форму для работы с созданными объектами базы данных (таблицы, формы, отчеты, формы). Предусмотреть выход из БД.

Порядок выполнения:
Открыть созданную ранее базу данных. Изучить предложенный пример создания кнопочных форм. На основе этого примера создать собственную кнопочную форму. Проверить работу каждой созданной кнопки.

Форма отчетности:
Отчет набирается на компьютере и сдается в печатном виде. В отчете должны присутствовать:
1. Номер варианта
2. Цель работы
3. Задание
4. Поэтапное выполнение всех заданий варианта
5. Вывод.

Задания для самостоятельной работы:
Предусмотрены вариантом студента.

Рекомендации по выполнению заданий и подготовке к лабораторной работе

Ознакомиться с теоретическим материалом, представленным в пятом разделе данной дисциплины.

Основная литература

1. Советов, Б. Я. Базы данных: теория и практика : учебник для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 2-е изд. – М. : Юрайт, 2014. – 463 с. – (Бакалавр. Базовый курс).

Дополнительная литература

1. Ульянов А.Д. Реляционные базы данных в СУБД Microsoft Access : лабораторный практикум. – Братск: Изд-во БрГУ, 2015. – 53 с.

Контрольные вопросы для самопроверки

1. Какие можно создать кнопки в кнопочной форме?
2. Как создать кнопки для выполнения запросов?
3. Как добавить Диспетчер кнопочных форм, если её сразу нет в программе?
4. Для чего создаются кнопочные формы в базе данных?

Лабораторная работа №7

Проектирование базы данных.

Цель работы:

Изучить принципы построения различных вариантов инфологических моделей данных, сравни полученные варианты.

Задание:

1. Разработать несколько вариантов инфологической модели предметной области.
2. Построить Сетевую, Иерархическую и Реляционную модель базы данных.

Порядок выполнения:

1. Изучить теоретические основы
2. По предметной области построить несколько инфологических моделей
3. Построить Сетевую, Иерархическую и Реляционную модель данных и сравнить их

Форма отчетности:

Отчет сдается в печатном виде. В отчете должны присутствовать:

1. Цель работы
2. Задание
3. Поэтапное выполнение всех заданий варианта
4. Вывод.

Задания для самостоятельной работы:

Изучить теоретические данные по теме лабораторной работы.

Рекомендации по выполнению заданий и подготовке к лабораторной работе

Ознакомиться с теоретическим материалом, представленным в третьем разделе данной дисциплины.

Основная литература

1. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин . – Москва : Юрайт, 2016. – 213 с. – (Бакалавр. Академический курс)

Контрольные вопросы для самопроверки

1. Что такое инфологические модели данных.

2. Избыточность данных
3. Аномалии данных
4. Сетевая модель данных
5. Иерархическая модель данных
6. Реляционная модель данных

Лабораторная работа №8

Создание базы данных при помощи языка SQL.

Цель работы:

Изучить создание и заполнение таблиц при помощи языка SQL.

Задание:

1. Изучить оператор создания таблиц в языке SQL.
2. Изучить оператор заполнения таблиц в языке SQL.

Порядок выполнения:

1. Изучить теоретические основы
2. Создать таблицы при помощи языка SQL
3. Заполнить таблицы при помощи языка SQL

Форма отчетности:

Отчет сдается в печатном виде. В отчете должны присутствовать:

1. Цель работы
2. Задание
3. Поэтапное выполнение всех заданий варианта
4. Заключение.

Задания для самостоятельной работы:

Изучить теоретические данные по теме лабораторной работы.

Рекомендации по выполнению заданий и подготовке к лабораторной работе

Ознакомиться с теоретическим материалом, представленным в шестом разделе данной дисциплины.

Основная литература

1. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин . – Москва : Юрайт, 2016. – 213 с. – (Бакалавр. Академический курс)

Контрольные вопросы для самопроверки

1. Особенности создания таблиц при помощи языка SQL.
2. Создание таблицы с несколькими ключевыми полями при помощи языка SQL.
3. Особенности заполнения таблиц при помощи языка SQL.
4. Вид оператора заполнения таблицы, если необходимо заполнить не все столбцы.

Лабораторная работа №9

Создание простых запросов при помощи языка SQL.

Цель работы:

Изучить создание простых запросов при помощи языка SQL

Задание:

1. Изучить работу оператора Select.
2. Изучить работу оператора From.
3. Изучить работу оператора Where.
4. Создание связи при помощи языка SQL.

5. Изучить работу операторов Is Null, Is Not Null, Like, UNION.

Порядок выполнения:
1. Изучить теоретические основы
2. Создать несколько простых запросов с использованием Select, From.
3. Создать связь между таблицами при помощи языка SQL.
4. Создать несколько запросов с условием.

Форма отчетности:
Отчет сдается в печатном виде. В отчете должны присутствовать:
1. Цель работы
2. Задание
3. Поэтапное выполнение всех заданий варианта
4. Заключение.

Задания для самостоятельной работы:
Изучить теоретические данные по теме лабораторной работы.

Рекомендации по выполнению заданий и подготовке к лабораторной работе
Ознакомиться с теоретическим материалом, представленным в шестом разделе данной дисциплины.

Основная литература

1. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин . – Москва : Юрайт, 2016. – 213 с. – (Бакалавр. Академический курс)

Контрольные вопросы для самопроверки

1. Принцип работа оператора Select.
2. Принцип работа оператора From.
3. Как создать связь между таблицами при помощи языка SQL.
4. Принцип работы оператора условия Where.
5. Принцип работы операторов Is Null, Is Not Null, Like, UNION.

Лабораторная работа №10

Создание сложных запросов при помощи языка SQL.

Цель работы:
Изучить создание сложных запросов при помощи языка SQL
Задание:
1. Создать несколько группирующих запросов**,** в которых определяются условия, причем сначала выполняются вычисления, а затем происходит отбор.
2. Создать несколько группирующих запросов**,** в которых определяются условия, причем сначала происходит отбор, а затем выполняются вычисления.
3. Создать несколько вложенных запросов.

Порядок выполнения:
1. Изучить теоретические основы
2. Создать несколько группирующих запросов**,** в которых определяются условия, причем сначала выполняются вычисления, а затем происходит отбор.
3. Создать несколько группирующих запросов**,** в которых определяются условия, причем сначала происходит отбор, а затем выполняются вычисления.
4. Создать несколько вложенных запросов.

Форма отчетности:

Отчет сдается в печатном виде. В отчете должны присутствовать:

1. Цель работы
2. Задание
3. Поэтапное выполнение всех заданий варианта
4. Заключение.

Задания для самостоятельной работы:
Изучить теоретические данные по теме лабораторной работы.

Рекомендации по выполнению заданий и подготовке к лабораторной работе
Ознакомиться с теоретическим материалом, представленным в шестом разделе данной дисциплины.

Основная литература

1. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин . – Москва : Юрайт, 2016. – 213 с. – (Бакалавр. Академический курс)

Контрольные вопросы для самопроверки

1. Принцип работа операторов avg, sum, min, max.
2. Принцип работа операторов Group by, Order by.
3. Принцип работа оператора Having.
4. Принцип работа оператора Exist.
5. Принцип работа оператора Distinct.

Практическое занятие №1
Основные модели данных

Цель работы:
Приобрести общие представление о моделях данных СУБД

Задание:

1. Познакомиться с основными моделями данных

Порядок выполнения:
Изучить теоретические данные. Изучить теоретические сведенья о сетевой модели данных. Изучить теоретические сведенья об иерархической модели данных. Изучить теоретические сведенья о реляционной модели данных. Привести по одному примеру баз дынных в каждой из изученных моделях.

Форма отчетности:
Отчет сдается в печатном виде. В отчете должны присутствовать:

1. Цель работы
2. Задание
3. Поэтапное выполнение всех заданий варианта
4. Вывод.

Задания для самостоятельной работы:
Предусмотрены вариантом студента.

Рекомендации по выполнению заданий и подготовке к практическому занятию
Ознакомиться с теоретическим материалом, представленным в третьем разделе данной дисциплины.

Основная литература

1. Советов, Б. Я. Базы данных: теория и практика : учебник для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 2-е изд. – М. : Юрайт, 2014. – 463 с. – (Бакалавр. Базовый курс).
2. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин . – Москва : Юрайт, 2016. – 213 с. – (Бакалавр. Академический курс).

Контрольные вопросы для самопроверки

1. Особенности иерархической модели данных.
2. Особенности сетевой модели данных.
3. Особенности реляционной модели данных.

Практическое занятие №2
Этапы и методы проектирование баз дынных.

Цель работы:
Приобрести навыки проектирования баз данных.
Задание:

1. Изучить основные понятия проектирования баз данных.
2. Изучить основные этапы проектирования баз данных.

Порядок выполнения:
Изучить теоретические данные. Изучить основные этапы проектирования баз данных. Изучить основы CASE технологий проектирования. Изучить диаграмму потоков данных. Изучить методологию SADT. Привести пример проектирования базы данных в любом из рассмотренных способов.

Форма отчетности:
Отчет сдается в печатном виде. В отчете должны присутствовать:

1. Цель работы
2. Задание
3. Поэтапное выполнение всех заданий варианта
4. Заключение.

Задания для самостоятельной работы:
Предусмотрены вариантом студента.

Рекомендации по выполнению заданий и подготовке к практическому занятию
Ознакомиться с теоретическим материалом, представленным в пятом разделе данной дисциплины.

Основная литература

1. Советов, Б. Я. Базы данных: теория и практика : учебник для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 2-е изд. – М. : Юрайт, 2014. – 463 с. – (Бакалавр. Базовый курс).
2. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин . – Москва : Юрайт, 2016. – 213 с. – (Бакалавр. Академический курс).

Контрольные вопросы для самопроверки

1. Назвать основные этапы проектирования баз данных.
2. Особенности CASE технологий проектирования.

3. Кто разработал диаграмму потоков данных.
4. В чем отличие методологии SADT от других технологий.

Практическое занятие №3
Основы языка SQL.

Цель работы:
Приобрести навыки работы с языком SQL.

Задание:
1. Изучить основы языка SQL .

Порядок выполнения:
Изучить историю возникновения языка SQL. Познакомиться с основными свойствами языка SQL. Изучить основные операторы языка. Привести примеры создания запросов при помощи языка SQL.

Форма отчетности:
Отчет сдается в печатном виде. В отчете должны присутствовать:
1. Цель работы
2. Задание
3. Поэтапное выполнение всех заданий варианта
4. Заключение.

Задания для самостоятельной работы:
Предусмотрены вариантом студента.

Рекомендации по выполнению заданий и подготовке к практическому занятию
Ознакомиться с теоретическим материалом, представленным в шестом разделе данной дисциплины.

Основная литература

1. Советов, Б. Я. Базы данных: теория и практика : учебник для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 2-е изд. – М. : Юрайт, 2014. – 463 с. – (Бакалавр. Базовый курс).
2. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин . – Москва : Юрайт, 2016. – 213 с. – (Бакалавр. Академический курс).

Контрольные вопросы для самопроверки

1. Кто создал язык SQL.
2. Когда впервые был опубликован язык SQL.
3. Назвать операторы создания и уничтожения отношений.

9.2. Методические указания по выполнению контрольной работы

Работа посвящена разработке информационного обеспечения задачи предметной области. Задание включает в себя следующие разделы:
1. Анализ предметной области.
2. Разработка информационного обеспечения задачи.

2.1. Информационный анализ входной информации, необходимой для решения задачи и выделение информационных объектов предметной области

2.1.1.Структура предметной области.
2.2. Определение связей информационных объектов и построение частной информационно-логической модели.
2.2.1. Альтернативное описание предметной области
2.2.2. ER- модель.
2.2.3. Генерация схем отношений.
2.3. Определение логической структуры базы данных.
Расчет производиться каждым студентом индивидуально, по вариантам.

10. ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ ПРИ ОСУЩЕСТВЛЕНИИ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

Информационно-коммуникационные технологии (ИКТ) – преподаватель использует для:

- получения информации при подготовке к занятиям,
- создания презентационного сопровождения лекций;
- интерактивного общения;
- ОС Windows 7 Professional;
- Microsoft Office 2007 Russian Academic OPEN No Level ;
- Антивирусное программное обеспечение KasperskySecurity

11. ОПИСАНИЕ МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЙ БАЗЫ, НЕОБХОДИМОЙ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

Вид занятия	***Наименование аудитории***	***Перечень основного оборудования***	***№ ЛР или ПЗ***
1	**3**	**4**	**5**
ЛР	Лекционный кабинет/дисплейный класс	Персональные компьютеры	ЛР 1-10
ПЗ	Лекционный кабинет/дисплейный класс	Персональные компьютеры	ПЗ 1-3
СР	ЧЗЗ	-	-

Приложение 1

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

1. Описание фонда оценочных средств (паспорт)

№ компетенции	Элемент компетенции	Раздел	Тема	ФОС
ОПК-6	Способность осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий.	**1.** Введение в базы данных	1.1. Основные понятия и определения	Вопрос к зачету. Экзаменационный билет
			1.3. Базы данных.	
		2. Архитектура СУБД	2.1. Трехуровневая архитектура базы данных	Вопрос к зачету. Экзаменационный билет
		3. Модели данных	3.1. Классификация моделей данных.	Вопрос к зачету. Экзаменационный билет
		4. Реляционная модель данных	4.1. История вопроса	Экзаменационный билет
			4.3. Обновление отношений.	
		5. Проектирование базы данных.	5.1. Избыточность данных и аномалии обновления в БД.	Экзаменационный билет
		6. Язык SQL	6.1.Оператор выбора SELECT. Формирование запросов к базе.	Экзаменационный билет
ОПК-9	Способность использовать навыки работы с компьютером, владеть методами информационных технологий, соблюдать основные требования информационной безопасности	**1.** Введение в базы данных	1.2. Современное состояние технологий баз данных.	Вопрос к зачету. Экзаменационный билет
		2. Архитектура СУБД	2.2. Функции СУБД.	Вопрос к зачету. Экзаменационный билет
			2.4. Архитектура многопользовательских СУБД.	
		3. Модели данных	3.2. Сетевая модель	Вопрос к зачету. Экзаменационный билет
		4. Реляционная модель данных	4.2. Структура часть реляционной модели.	Экзаменационный билет
		5. Проектирование базы данных.	5.2. Нормализация отношений.	Экзаменационный билет
		6. Язык SQL	6.2. Операторы манипулирования данными.	Экзаменационный билет
ПК-5	Способность осуществлять сбор и анализ исходных данных для расчета и	**1.** Введение в базы данных	1.4. Системы управления базами данных.	Вопрос к зачету. Экзаменационный билет
		2. Архитектура СУБД	2.3. Языки СУБД.	Вопрос к зачету. Экзаменационный билет

	проектирования систем и средств автоматизации и управления	**3.** Модели данных	3.3. Иерархическая модель данных.	Вопрос к зачету. Экзаменационный билет
		4. Реляционная модель данных	4.4. Целость базы данных.	Экзаменационный билет
		5. Проектирование базы данных.	5.3. Проектирование реляционной базы данных.	Экзаменационный билет
		6. Язык SQL	6.3. Операторы определения данных.	Экзаменационный билет

2. Вопросы к зачету

№ п/п	Компетенции		Вопросы к зачету	№ и наименование раздела
	Код	Определение		
1	2	3	4	5
1	ОПК-6	Способность осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий.	**1.**Определение базы данных	**1.** Введение в базы данных
			2.Определение предметной области	
			3.Определение банка данных	
			4. Негативные последствия избыточности в базах данных	
			5. Привести пример связи между сущностями	
			1. Назвать три уровня архитектуры базы данных	**2.** Архитектура СУБД
			2. Чем характерен внешний уровень?	
			3. Что такое концептуальное проектирование?	
			1. Объектные модели данных	**3.** Модели данных
			2. Физическая модель данных	
			3. Модели данных на основе записей	
2	ОПК-9	Способность использовать навыки работы с компьютером, владеть методами информационных технологий, соблюдать основные требования информационной безопасности Способность осуществлять поиск, хранение, обработку и	**1.** Что входит в банк данных?	**1.** Введение в базы данных
			2. Назвать основные современные принципы организации баз данных	
			3. Что содержится в словаре данных?	
			1. Перечислить основные функции СУБД	**2.** Архитектура СУБД
			2. Кратко рассказать о функции управления транзакциями	
			3. Кратко рассказать о функции восстановление базы данных	
			4. Перечислить двухуровневые модели СУБД	
			5. Сервер приложений	
			1. История вопроса	**3.** Модели данных

		анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий.	**2.** Основные структуры сетевой модели	
			3. Элемент данных	
			4. Запись	
			5. Набор	
3	ПК-5	Способность осуществлять сбор и анализ исходных данных для расчета и проектирования систем и средств автоматизации и управления	**1.** Определение СУБД	**1.** Введение в базы данных
			2. Современные СУБД	
			3. Дать обобщенную характеристику современных СУБД	
			1. Язык определения данных	**2.** Архитектура СУБД
			2. Язык манипулирования данными	
			1. В виде чего можно представить иерархическую модель данных?	**3.** Модели данных
			2. Какого типа связей не может быть в иерархической модели данных?	

3. Экзаменационные вопросы

№ п/п	Компетенции		ЭКЗАМЕНАЦИОННЫЕ ВОПРОСЫ	№ и наименование раздела
	Код	Определение		
1	2	3	4	5
1	ОПК-6	Способность осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий.	**1.** Основные понятия и определения.	**1.** Введение в базы данных
			2. Базы данных.	
			3. Трехуровневая архитектура базы данных.	**2.** Архитектура СУБД
			4. Классификация моделей данных.	**3.** Модели данных
			5. История вопроса.	**4.** Реляционная модель данных
			6. Обновление отношений.	
			7. Избыточность данных и аномалии обновления в БД.	**5.** Проектирование базы данных.
			8. Преобразование бинарных связей.	
			9.Оператор выбора SELECT. Формирование запросов к базе.	**6.** Язык SQL
			10. Операторы Group by, Order by.	
2	ОПК-9	Способность использовать навыки работы с компьютером, владеть методами	**1.** Современное состояние технологий баз данных.	**1.** Введение в базы данных
			2. Системы управления базами данных.	
			3. Привести пример ER-диаграммы базы данных.	

		информационных технологий, соблюдать основные требования информационной безопасности Способность осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий.	**4.** Архитектура многопользовательских СУБД.	**2.** Архитектура СУБД
			5. Модели двухуровневой технологии «клиент-сервер».	
			6. Сетевая модель.	**3.** Модели данных
			7. Структуры данных сетевой модели.	
			8. Преобразование концептуальной модели в сетевую.	
			9. Структура часть реляционной модели.	**4.** Реляционная модель данных
			10. Свойства и виды отношений.	
			11. Первая и вторая нормальные формы.	**5.** Проектирование базы данных.
			12. Третья и нормальная форма Бойса-Кодда.	
			13. Операторы манипулирования данными.	**6.** Язык SQL
			14. Агрегатные функции языка.	
3	ПК-5	Способность осуществлять сбор и анализ исходных данных для расчета и проектирования систем и средств автоматизации и управления	**1.** Целостность, восстанавливаемость, безопасность, эффективность баз данных.	**1.** Введение в базы данных
			2. Языки СУБД.	**2.** Архитектура СУБД
			3. История создания языка SQL.	
			4. Функции СУБД.	
			5. Иерархическая модель данных.	**3.** Модели данных
			6. Управляющая часть иерархической модели.	
			7. Достоинства и недостатки реляционной модели данных.	**4.** Реляционная модель данных
			8. Целость базы данных.	
			9. Проектирование реляционной базы данных.	**5.** Проектирование базы данных.
			10. Нормализация отношений.	
			11. Операторы определения данных.	**6.** Язык SQL
			12. Вложенные запросы.	

4. Описание показателей и критериев оценивания компетенций

Показатели	Оценка	Критерии
Знать (ОПК-6): - оценивать производительность вычислительных машин и систем, выбирать вычислительные средства для проектирования устройств и систем управления; (ОПК-9): - Базовое устройство персонального компьютера. (ПК-5): - Основные информационные процессы происходящие в персональном компьютере. **Уметь** (ОПК-6): – Применять принципы и методы построения моделей, методы анализа, синтеза и оптимизации при создании и исследования средств и систем управления; (ОПК-9): – Использовать персональный компьютер для самостоятельной работы. (ПК-5): – Использовать сбор и анализ исходный данных для расчета и проектирования систем управления базами данных. **Владеть** (ОПК-3): – Навыками работы с современными аппаратными и программными средствами исследования и проектирования систем управления; (ОПК-9): – Достаточным уровнем использования универсальных пакетов прикладных компьютерных программ (ПК-5): – Навыками работы с современными аппаратными и программными средствами сбора и анализа информации.	**отлично**	Студент должен во время ответа показать знания: баз данных, СУБД, различных моделей, основных терминов используемые в научно-технической литературе по базам данных. Студент должен иметь навыки владения: использования универсальных программных продуктов на ПК, понимания материала и способности высказывания мыслей на научно-техническом языке. Студент во время ответа должен продемонстрировать умения: использования навыков анализа основных понятий в теории систем управления базами данных.
	хорошо -	Ответ содержит неточности. Дополнительные вопросы требуется, но студент с ними справляется отлично.
	удовлетворительно	Ответил только на один вопрос, либо слабо ответил на оба вопроса. На дополнительные вопросы отвечает неуверенно.
	неудовлетворительно	На оба вопроса студент отвечает неубедительно. На дополнительные вопросы преподавателя также не может ответить.
	зачтено	Во время ответа на зачете студент продемонстрировал уверенное знание материала и ответил на вопросы преподавателя.
	не зачтено	На поставленные вопросы студент не ответил. На дополнительные вопросы преподавателя также не может ответить.

4. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и опыта деятельности

Дисциплина системы управления базами данных направлена на ознакомление с базами данных, и их практическим применением в современных системах телекоммуникаций; на получение теоретических знаний и практических навыков использования различных СУБД и языка запросов SQL, и их дальнейшего использования в практической деятельности.

Изучение дисциплины системы управления базами данных предусматривает:

- лекции,
- лабораторные работы,
- практические занятия,
- контрольную работу,
- самостоятельную работу студента,
- зачет,
- экзамен.

В ходе освоения раздела 1 «Введение в базы данных» студенты должны изучить: основные понятия и определения дисциплины и способы их применения.

В ходе освоения раздела 2 «Архитектура СУБД» студенты должны изучить: различные виды архитектур построения СУБД, основные функции СУБД и основные языки управления СУБД.

В ходе освоения раздела 3 «Модели данных» студенты должны изучить: классификацию моделей данных, сетевую и иерархическую модели.

В ходе освоения раздела 4 «Реляционная модель данных» студенты должны изучить: историю возникновения реляционной модели, структурную часть модели, обновление отношений и целостность данных в данной модели данных.

В ходе освоения раздела 5 «Проектирование базы данных» студенты должны изучить: избыточность данных и аномалии обновлений в БД, нормализацию отношений, проектирование реляционной базы данных, и применить полученные знания на практике.

В ходе освоения раздела 6 «Язык SQL» студенты должны изучить: написание запросов, как работают различные операторы языка SQL, и способы определения данных при помощи языка SQL.

В процессе проведения лабораторных работ происходит закрепление знаний, формирование умений и навыков реализации представления о работе с базами данных и использование языка SQL.

В процессе проведения практических работ происходит закрепление знаний, формирование умений и навыков проектирования различных моделей данных и формирования запросов к ним.

При подготовке к экзамену рекомендуется особое внимание уделить следующим вопросам: достоинства и недостатки реляционной модели данных, сетевая модель, функции СУБД и нормализация отношений.

Работа с литературой является важнейшим элементом в получении знаний по дисциплине. Прежде всего, необходимо воспользоваться списком рекомендуемой по данной дисциплине литературой. Дополнительные сведения по изучаемым темам можно найти в периодической печати и Интернете.

Приложение 2

АННОТАЦИЯ
рабочей программы дисциплины
Системы управления базами данных

1. Цель и задачи дисциплины

Целью изучение дисциплины является: изучение теории баз данных, формирование практических навыков проектирования информационных систем на основе баз данных, формирование практических навыков создания реляционных баз данных в современных СУБД, формирование практических навыков по использованию языка запросов SQL, формирование практических навыков работы с инструментальными средствами быстрой разработки приложений.

Задачей изучения дисциплины является: подготовка студента к самостоятельному решению теоретических и прикладных задач связанных с проектированием информационных систем на основе баз данных.

2. Структура дисциплины

2.1 Распределение трудоемкости по отдельным видам учебных занятий, включая самостоятельную работу: Лк – 35 часов, ЛР – 35 часов, ПЗ – 17 часов, СРС – 57 часов.
Общая трудоемкость дисциплины составляет 180 часов, 5 зачетных единиц

2.2 Основные разделы дисциплины:

1. Введение в базы данных
2. Архитектура СУБД
3. Модели данных
4. Реляционная модель данных
5. Проектирование базы данных
6. Язык SQL

3. Планируемые результаты обучения (перечень компетенций)

Процесс изучения дисциплины направлен на формирование следующей компетенции:

ОПК-6 - Способность осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий;

ОПК-9 - Способность использовать навыки работы с компьютером, владеть методами информационных технологий, соблюдать основные требования информационной безопасности

ПК-5 - Способность осуществлять сбор и анализ исходных данных для расчета и проектирования систем и средств автоматизации и управления

4. Вид промежуточной аттестации: зачет, экзамен

Приложение 3

Протокол о дополнениях и изменениях в рабочей программе
на 20___-20___ учебный год

1. В рабочую программу по дисциплине вносятся следующие дополнения:
_____
_____

2. В рабочую программу по дисциплине вносятся следующие изменения:

_____
_____
_____

Протокол заседания кафедры №____ от «___» _____ 20 __ __ г.,
(разработчик)

Заведующий кафедрой _____ _____
(подпись) *(Ф.И.О.)*

Приложение 4

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ ПО ДИСЦИПЛИНЕ

1. Описание фонда оценочных средств (паспорт)

№ компетенции	Элемент компетенции	Раздел	Тема	ФОС
ОПК-6	Способность осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий.	**3.** Модели данных	Классификация моделей данных.	*Отчеты по лабораторным работам, практическим занятиям.*
		4. Реляционная модель данных	История вопроса	*Отчеты по лабораторным работам.*
			Обновление отношений.	
		5. Проектирование базы данных.	Избыточность данных и аномалии обновления в БД.	*Отчеты по лабораторным работам, практическим занятиям.*
		6. Язык SQL	Оператор выбора SELECT. Формирование запросов к базе.	*Отчеты по лабораторным работам, практическим занятиям.*
ОПК-9	Способность использовать навыки работы с компьютером, владеть методами информационных технологий, соблюдать основные требования информационной безопасности.	**3.** Модели данных	Сетевая модель.	*Отчеты по лабораторным работам, практическим занятиям.*
		4. Реляционная модель данных	Структура часть реляционной модели.	*Отчеты по лабораторным работам,*
		5. Проектирование базы данных.	Проектирование реляционной базы данных.	*Отчеты по лабораторным работам, практическим занятиям, контрольная работа.*
		6. Язык SQL	Операторы манипулирования данными.	*Отчеты по лабораторным работам, практическим занятиям.*

ПК-5	Способность осуществлять сбор и анализ исходных данных для расчета и проектировани я систем и средств автоматизации и управления	**3.** Модели данных	Иерархическая модель данных.	*Отчеты по лабораторным работам, практическим занятиям.*
		4. Реляционная модель данных	Целость базы данных.	*Отчеты по лабораторным работам, практическим занятиям.*
		5. Проектирован ие базы данных.	Нормализация отношений.	*Отчеты по лабораторным работам, практическим занятиям, контрольная работа.*
		6. Язык SQL	Операторы определения данных.	*Отчеты по лабораторным работам, практическим занятиям.*

2. Описание показателей и критериев оценивания компетенций

Показатели	Оценка	Критерии
Знать (ОПК-6): - оценивать производительность вычислительных машин и систем, выбирать вычислительные средства для проектирования устройств и систем управления; (ОПК-9): - Базовое устройство персонального компьютера. (ПК-5): - Основные информационные процессы происходящие в персональном компьютере. **Уметь** (ОПК-6): – Применять принципы и методы построения моделей, методы анализа, синтеза и оптимизации при создании и исследования средств и систем управления; (ОПК-9): – Использовать персональный компьютер для самостоятельной работы. (ПК-5): – Использовать сбор и анализ исходный данных для расчета и проектирования систем управления базами данных. **Владеть** (ОПК-3): – Навыками работы с современными	**зачтено**	Во время защиты лабораторных работ и практических работ студент ответил на поставленные преподавателем вопросы.
	не зачтено	Во время защиты лабораторных работ и практических работ студент не смог дать ответы на поставленные преподавателем вопросы. Либо отчет имеет ряд замечаний.

аппаратными и программными средствами исследования и проектирования систем управления; (ОПК-9): – Достаточным уровнем использования универсальных пакетов прикладных компьютерных программ (ПК-5): Навыками работы с современными аппаратными и программными средствами сбора и анализа информации.		