Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Ярославский государственный университет им. П. Г. Демидова
Кафедра компьютерной безопасности и математических методов
обработки информации

О. В. Власова

Системы управления базами данных

Лабораторный практикум

Рекомендовано
Научно-методическим советом университета для студентов,
обучающихся по специальностям Прикладная математика
и информатика, Компьютерная безопасность

Ярославль 2010

УДК 004.65
ББК З 973.233–018.2
В 58

Рекомендовано
Редакционно-издательским советом университета
в качестве учебного издания. План 2009/10 года

Рецензент – кафедра компьютерной безопасности
и математических методов обработки информации

Власова, О. В. Системы управления базами данных: лабораторный практикум / О. В. Власова; Яросл. гос. ун-т им. П. Г. Демидова. – Ярославль : ЯрГУ, 2010. – 76 с.

В 58

Лабораторный практикум посвящен важнейшей составляющей широко разрабатываемых и используемых информационных систем организационного управления – базам данных (БД), создаваемым и функционирующим на основе систем управления базами данных (СУБД).

Содержит концептуальные представления об основных принципах построения БД и СУБД, принципах проектирования БД, а также анализ основных технологий реализации БД. Особое внимание уделяется представлению фундаментальных понятий и математических моделей, лежащих в основе реляционных БД и СУБД.

Предназначено для студентов, обучающихся по специальностям 010501.65 Прикладная математика и информатика, 090102.65 Компьютерная безопасность (дисциплина «Базы данных и экспертные системы, СУБД», блок ОПД), очной формы обучения.

УДК 004.65
ББК З 973.233–018.2

Учебное издание

Власова Ольга Владимировна

Системы управления базами данных

Лабораторный практикум

Редактор, корректор И. В. Бунакова
Верстка И. Н. Иванова
Подписано в печать 29.04.10. Формат 60×84 $^{1}/_{16}$. Бум. офсетная.
Гарнитура «Times NewRoman». Усл. печ. л. 4,42. Уч.-изд. л. 3,09.
Тираж 100 экз. Заказ

Оригинал-макет подготовлен в редакционно-издательском отделе Ярославского государственного университета им. П. Г. Демидова.
Отпечатано на ризографе.
Ярославский государственный университет им. П. Г. Демидова.
150000, Ярославль, ул. Советская, 14.

Предисловие

Последние десятилетия в области программирования характеризуются резким ростом количества создаваемых информационных систем организационного управления. Практически в каждой организации функционирует (или создается) такая система (или её элементы). Важнейшей структурной частью информационных систем являются БД, создаваемые и функционирующие на основе использования специализированных программных систем – СУБД. Все это обусловливает большую потребность в квалифицированных кадрах, способных как создавать информационные системы на основе СУБД, так и обслуживать соответствующие информационные системы и БД.

Цель данного учебного пособия состоит в формировании концептуальных представлений об основных принципах построения БД, СУБД; о математических моделях, описывающих БД; о принципах проектирования БД.

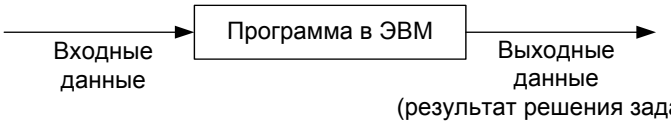
Предварительные знания

Курс «Базы данных» опирается на материалы следующих курсов: «Основы построения ЭВМ»; «ЭВМ и программирование»; «Дискретная математика».

Теоретические сведения

1.1. Развитие основных понятий представления данных

Любой вычислительный процесс представляет собой отображение (по определенному алгоритму) некоторых входных данных в выходные.



Соотношение сложности представления обрабатываемых данных и алгоритма вычислений определяет два класса задач:

- вычислительные задачи – достаточно простое представление данных и сложный, многооперационный процесс вычислений;
- задачи обработки данных (невычислительные задачи) – простой алгоритм обработки данных и сложное представление обрабатываемых данных.

На начальной стадии обучения программированию основное внимание уделяется разработке алгоритма решения задачи. Однако часто оказывается, что возможность (или невозможность) решения конкретной задачи зависит не только от выбранного алгоритма, но и от того, какие понятия используются для представления обрабатываемых данных.

Рассмотрим простейший пример нахождения корней квадратного уравнения:

$$A * X^2 + B * X + C = 0,$$

где *X, A, B* и *C* – числа, которые являются здесь элементарными единицами данных (элементами данных).

При программировании алгоритма решения этой задачи используется простейший вид данных – простая переменная. Заметим, что каждая простая переменная характеризуется определенным типом значений, который должен выбираться при программировании. Даже в этом простейшем случае необходимо правильно выбрать тип переменной, причем от этого выбора

может зависеть возможность или невозможность решения конкретной задачи.

Рассмотрим другой пример:

$$S = a_1 + a_2 + ... + a_N .$$

Решение этой задачи в общем случае невозможно получить, используя только простые переменные. Здесь обрабатывается не отдельное число, а последовательность чисел. В этом случае при программировании используется такой вид данных, как массив – совокупность элементов, с каждым из которых связан упорядоченный набор целых чисел, называемых индексами. Все элементы должны иметь одинаковый тип значений, который и будет типом массива. В этом случае числа a_1, a_2,…, a_N представляются в программе массивом $A[1]$, $A[2]$,..., $A[N]$.

Приведенные примеры показывают, что изменение вида задач обусловливает необходимость использования других типов данных.

Ранние языки программирования (ФОРТРАН, АЛГОЛ-60) были предназначены для решения научно-технических вычислительных задач. В этих языках использовались только вышеуказанные типы данных (простые переменные и массивы), что было вполне достаточно.

Начиная с конца 1960-х годов компьютеры начинают интенсивно использоваться для решения так называемых невычислительных задач, связанных с обработкой различного рода документов. Рассмотрим появление новых типов данных на примере упрощенных задач обработки данных.

Задача 1. Оплата услуг телефонной компании.

Рассматриваем задачу при двух упрощающих предположениях:

- клиент вносит абонентскую плату за K минут разговора;
- клиенту начисляется оплата в зависимости от количества потраченных минут.

Необходимые для решения этой задачи сведения о клиенте представлены в следующей карточке НАЧИСЛЕНИЕ:

Фамилия, имя, отчество	Абонентская плата	Количество минут, входящих в абонентскую плату	Количество потраченных минут в месяце	Начисленная сумма
FIO	*A*	*K*	*K*o	*S*

Для каждого клиента начисленная сумма за определенный месяц рассчитывается по следующей формуле:

$$S = \begin{cases} A, \ K_o \le K \\ A + (K_o - K) * C, \ K_o > K, \end{cases}$$

где C – стоимость одной минут.

Для каждого клиента соответствующие данные имеют конкретное значение, например:

Иванов Иван Иванович	180	30	24	180

Эти значения имеют смысл только во взаимосвязи друг с другом. Отдельно выбранное число 180 теряет свой содержательный смысл, поэтому использовать такой тип данных, как простая переменная, здесь нельзя. В то же время набор соответствующих значений, характеризующих конкретного сотрудника, имеет разные типы (символьный и числовой), т. е. использовать для его представления такой тип данных, как массив, также нельзя. Таким образом, понятий «простая переменная» и «массив» недостаточно, чтобы представить соответствующую карточку.

Для описания данных в предметной области невычислительных задач вводится ряд новых понятий.

Элемент данных (поле) – наименьшая единица поименованных данных.

Для данного примера элементами данных являются *FIO*, *A*, *K*, *K*o, *S*.

Для описания карточки клиента используется понятие «Логическая запись».

Логическая запись – поименованная совокупность элементов данных (полей).

Экземпляр логической записи – текущее значение элементов записи.

Для представления всего набора карточек клиентов используется понятие «Логический файл»ю

Логический файл – поименованная совокупность всех экземпляров записей заданного типа.

Пример логического файла НАЧИСЛЕНИЕ:

Иванов Иван Иванович	180	30	24	180
Петров Петр Петрович	220	25	20	220
Сидоров Сидор Сидорович	250	20	24	300

Таким образом, с помощью введенных понятий можно описывать соответствующие данные. Для отображения этих понятий в современных языках программирования, предназначенных как для вычислительных задач, так и для задач обработки данных, введены новые типы данных.

В алгоритмическом языке Паскаль вводится такой тип данных, как запись (RECORD) – сложная переменная с несколькими компонентами, которые могут иметь разные типы. Кроме того, доступ к компонентам записи (полям) осуществляется не по индексу, а по имени. При программировании задачи 1 на языке Паскаль логическая запись НАЧИСЛЕНИЕ представляется типом данных RECORD, набор экземпляров логических записей сотрудников (логический файл) – «физическим» файлом, формируемым средствами языка Паскаль и операционной системы.

```
Salary = RECORD
        FIO:    string;
        A:      real;
        K:      integer;
        Ko:     integer;
        S:      real;
END;
```

Отметим важную специфику таких невычислительных задач. Для этих задач характерны большие объемы данных (большое количество клиентов, большое количество производимых звонков и т. п.). Указанные данные, как правило, используются для решения задачи многократно (оплата начисляется постоянно каждый месяц), поэтому данные должны достаточно долго храниться в памяти ЭВМ. Для длительного хранения всегда используется внешняя память.

В связи с этим решение задачи 1 состоит из двух этапов:

1. Ввод исходных данных и занесение их во внешнюю память.

2. Чтение исходных данных из внешней памяти, расчет начисленных сумм и вывод на печать.

Необходимые данные хранятся в файле, предназначенном только для решения этой задачи. Отметим, что в этом случае описание данных включено в прикладную программу. При изменении формата записей файла необходимо изменение прикладной программы. Таким образом, программная система, решающая поставленную задачу, определяет свои собственные данные и управляет ими. Такие программные системы называются файловыми системами.

Задача 2. Учет состава клиентов.

Здесь обрабатываются сведения о клиенте, представленные в карточке КЛИЕНТ:

Фамилия, имя, отчество	Год рождения	Телефон	Место жительства
FIO	*G*	*T*	*M*

Решение задачи состоит из следующих этапов:

1. Ввод исходных данных и занесение их во внешнюю память.

2. Чтение исходных данных из внешней памяти с целью удаления, корректировки или добавления записи.

В рассматриваемом случае задача 2 решается независимо от задачи 1.

Задача 3. Учет экономии.

Обрабатываются сведения, представленные записями ЭКОНОМИЯ:

Фамилия, имя, отчество *FIO*	Количество непотраченных минут *Км*	Переплаченная сумма *SN*

$$SN = K_{\text{М}} * C .$$

Рассмотрим типичный случай, когда все три вышеуказанные программные системы функционируют в одной организации. Отметим следующие принципиальные эксплуатационные недостатки:

1. Информация дублируется. В трех файлах присутствуют поля *FIO*, что приводит к существенному перерасходу памяти. При внесении изменений (например, изменении фамилии) приходится вносить одно и то же значение несколько раз в разные файлы, что приводит к увеличению затрат машинного времени. Существует потенциальная возможность противоречивости данных (в один файл изменения внесены, в другой – нет).

2. Устранить перечисленные недостатки можно, объединив соответствующие записи и создав единую информационную базу для всех вышеназванных задач. На первый взгляд наиболее естественно объединить все записи в одну, убрав дублирующие поля. Получаем возможный вариант объединения:

FIO	*G*	*M*	*T*	*A*	*K*	K_O	K_M	*S*	*SN*

3. Дублирование информации полностью убрано. Расход памяти минимален. Недостатки устранены. Рассмотрим, как в этом случае изменится время решения задач 1 –3. Время решения задачи прямо пропорционально объему считываемых из внешней памяти данных.

Для нашего примера время решения задач в зависимости от выбранной длины полей может изменяться в 2–3 раза, что совершенно недопустимо.

Рассмотрим другой вариант построения единой информационной базы. Объединим записи задач 1 и 3, запись задачи 2 оставим отдельно. Получим два типа записей:

FIO	*G*	*M*	*T*

FIO	*A*	*K*	K_O	K_M	*S*	*SN*

В этом случае дублирование остается (*FIO*). Время решения задачи 1 и 3 в этом случае незначительно возрастет по сравнению с вариантом отдельных файлов, время решения задачи 2 такое же, как и в начальном варианте отдельного файла. Такое объединение позволяет значительно уменьшить влияние недостатков и в то же время несущественно увеличивает время решения всех задач. Все три задачи можно решать, используя общую информационную базу из двух типов записей. Отметим, что два приведенных типа записей связаны друг с другом по полю *FIO* (находятся в некотором отношении). Приведенные варианты интеграции не исчерпывают все возможные способы интеграции данных для приведенных задач, вопрос выбора наилучшего варианта рассмотрим в п. 1.2.1 и 1.5.

Для решения вышеуказанных задач используется некоторый новый вид данных, формируемый на основе интеграции записей.

Для описания этого вида данных вводится новое понятие **«База данных»**.

База данных (БД) – совокупность экземпляров различных типов записей и отношений между записями и элементами.

БД можно определить как совокупность взаимосвязанных хранящихся вместе данных при наличии такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений.

Таким образом, появление понятия **«Базы данных»** обусловлено возникновением нового класса невычислительных задач, при решении которых используются общие данные. В качестве основного критерия оптимальности функционирования БД, как правило, используются временные характеристики реализации запросов пользователей прикладными программами.

В прикладной программе, использующей при решении задачи один или несколько отдельных файлов, за сохранность и до-

стоверность данных отвечал программист, работающий с этой задачей. Использование БД предполагает работу с ней нескольких прикладных программ, решающих задачи разных пользователей.

Естественно, что за сохранность и достоверность интегрированных данных программист, решающий одну из прикладных задач, отвечать уже не может. Кроме того, расширение круга решаемых с использованием БД задач может приводить к появлению новых типов записей и отношений между ними. Такое изменение структуры БД не должно вести к изменению множества ранее разработанных и успешно функционирующих прикладных программных систем, работающих с БД. С другой стороны, возможное изменение любой из прикладных программ, в свою очередь, не должно приводить к изменению структуры данных. Все вышесказанное обусловливает необходимость отделения данных от прикладных программ.

Роль интерфейса между прикладными программами и БД, обеспечивающего их независимость, играет программный комплекс – система управления базами данных (СУБД) (рис.1.1).

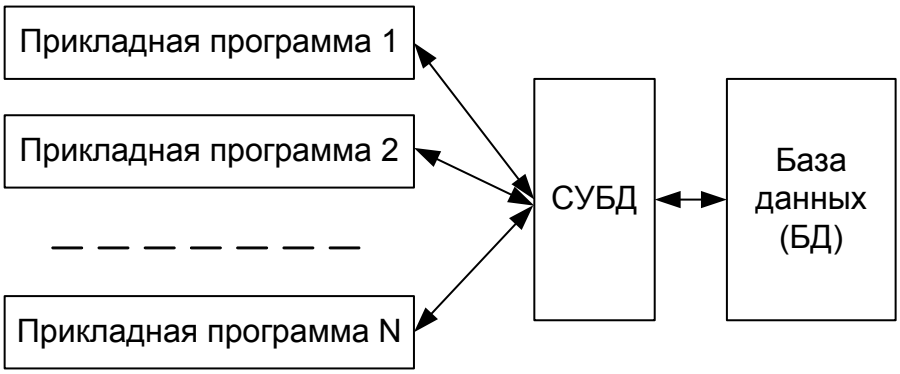


Рис. 1.1. Обеспечение независимости прикладных программ и БД

СУБД – программный комплекс поддержки интегрированной совокупности данных, предназначенный для создания, ведения и совместного использования БД многими пользователями (прикладными программами).

Перечислим основные **функции СУБД**.

1. Определение структуры создаваемой БД, ее инициализация и проведение начальной загрузки.

2. Предоставление пользователям возможности манипулирования данными (выборка необходимых данных, выполнение вычислений, разработка интерфейса ввода/вывода, визуализация).

3. Обеспечение независимости прикладных программ и данных (логической и физической независимости).

4. Защита логической целостности БД.

Основной целью реализации этой функции является повышение достоверности данных в БД. Достоверность данных может быть нарушена при их вводе в БД или при неправомерных действиях процедур обработки данных, получающих и заносящих в БД неправильные данные. Для повышения достоверности данных в системе объявляются так называемые ограничения целостности, которые в определенных случаях «отлавливают» неверные данные.

5. Защита физической целостности.

При работе ЭВМ возможны сбои в работе (например, из-за отключения электропитания), повреждение машинных носителей данных. При этом могут быть нарушены связи между данными, что приводит к невозможности дальнейшей работы. Развитые СУБД имеют средства восстановления базы данных.

6. Управление полномочиями пользователей на доступ к БД.

7. Синхронизация работы нескольких пользователей.

8. Управление ресурсами среды хранения.

БД располагается во внешней памяти ЭВМ. При работе в БД заносятся новые данные (занимается память) и удаляются данные (освобождается память). СУБД выделяет ресурсы памяти для новых данных, перераспределяет освободившуюся память, организует ведение очереди запросов к внешней памяти и т. п.

9. Поддержка деятельности системного персонала.

При эксплуатации БД может возникать необходимость изменения параметров СУБД, выбора новых методов доступа, изменения (в определенных пределах) структуры хранимых данных, а также выполнения ряда других общесистемных действий. СУБД предоставляет возможность выполнения этих и других действий для поддержки деятельности БД обслуживающему БД системному персоналу, называемому администратором БД.

1.2. Проектирование БД

Проектирование данных (БД) представляет собой процесс последовательного отображения исследуемых явлений реального мира в виде данных в памяти ЭВМ (рис. 1.2).

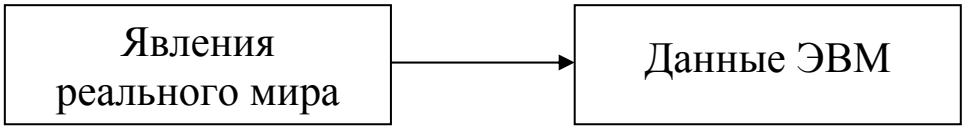


Рис. 1.2. Общая схема проектирования

Конкретные явления реального мира, представляющие интерес для проводимого исследования, будем называть предметной областью (ПО).

Создание БД предполагает интеграцию данных, предназначенных для решения нескольких прикладных задач разных пользователей. Соответственно, при интеграции данных должны учитываться требования к данным каждого пользователя, основанные на его представлении о данных и связях между ними. Далее эти требования должны обобщаться в единое представление, которое и будет служить основой для построения единой БД (рис. 1.3).

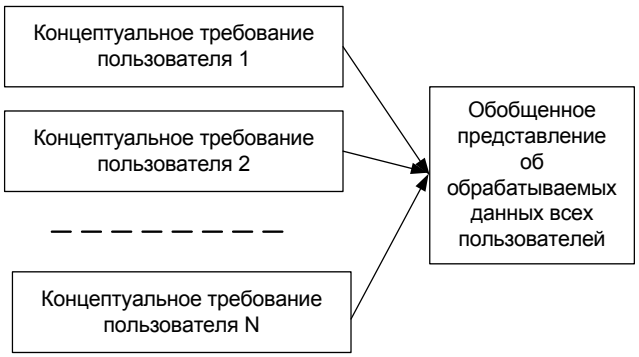


Рис. 1.3. Обобщение представления пользователей о данных

Обобщение представлений всех пользователей о данных называется концептуальной моделью (схемой) БД. Концептуальная модель представляет информационное описание предметной области с учетом логических взаимосвязей, поэтому её еще называют инфологической (информационно-логической) моделью. В модели отсутствуют какие-либо понятия, свя-

занные с ЭВМ, памятью ЭВМ, способами размещения данных в памяти ЭВМ, и, по сути, это модель только предметной области.

Как уже отмечалось, для создания базы данных и работы с ней используется СУБД. ***Каждая конкретная СУБД поддерживает определенный вид данных (форматов записей и отношений), называемый моделью данных СУБД.***

Следующий этап разработки БД предполагает выбор представления концептуальной модели с помощью модели данных конкретной СУБД. ***Полученное таким образом представление концептуальной модели называется логической моделью БД. Или другими словами, логическая модель – это концептуальная схема, специфицированная в языке конкретной СУБД.*** Логическая модель представляет данные и элементы данных вне зависимости от их содержания и среды хранения. Далее разработчик системы средствами СУБД отображает полученную логическую модель БД в память ЭВМ и определяет методы доступа. ***Полученное представление данных в памяти ЭВМ называется внутренним представлением, или структурой хранения (физической модели).*** Прикладные программы работают с логической моделью, причем каждому пользователю представляется подмножество этой логической модели (подсхема), отражающее его представление о предметной области. Каждая прикладная программа «видит» и обрабатывает только те данные, которые необходимы именно ей.

Соответствующее «видение» данных прикладными программами (пользователями) представляет собой внешние представления. Взаимосвязь вышеуказанных моделей изображена на рис. 1.4.

На данной схеме выделены три различных уровня описания данных (внешний, концептуальный, внутренний). ***Эти уровни формируют так называемую трехуровневую архитектуру*** ANSI/SPARC, предложенную в 1975 г. Комитетом планирования стандартов и норм SPARC (Standards Planning and Requirements Committee) Национального института стандартизации США (American National Standards Institute – ANSI). Основная цель этой архитектуры состоит в отделении пользовательского представления о данных в БД от их физического представления. Использование

таких представлений о данных позволяет обеспечить выполнение основного требования к БД – независимости программ и данных. При изменении прикладных программ может измениться соответствующее внешнее представление, но логическая модель данных не изменяется и, соответственно, это не отразится на других прикладных программах. При изменении внутреннего представления (структур хранения) логическая модель не изменяется, соответственно, не изменяются прикладные программы.

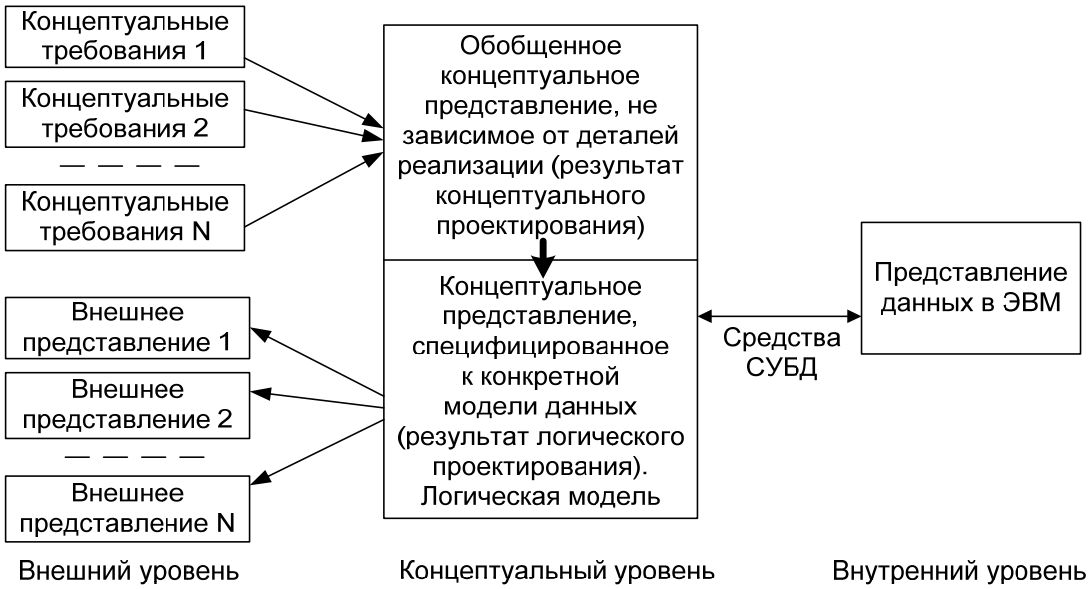


Рис. 1.4. Различные представления о данных в БД

Использование соответствующих представлений также позволяет четко разграничить полномочия различных лиц, работающих с базой данных.

Соответствующие представления позволяют описать «видение» БД разными лицами, работающими с ней:

- внешнее представление – представление специалиста предметной области (пользователя);
- внешнее представление и логическая модель – представление прикладного программиста, разрабатывающего конкретное приложение для пользователя;
- логическая модель и внутреннее представление – представление системного программиста, администрирующего БД.

Более детально описанный многоэтапный процесс приведен на рис. 1.5.

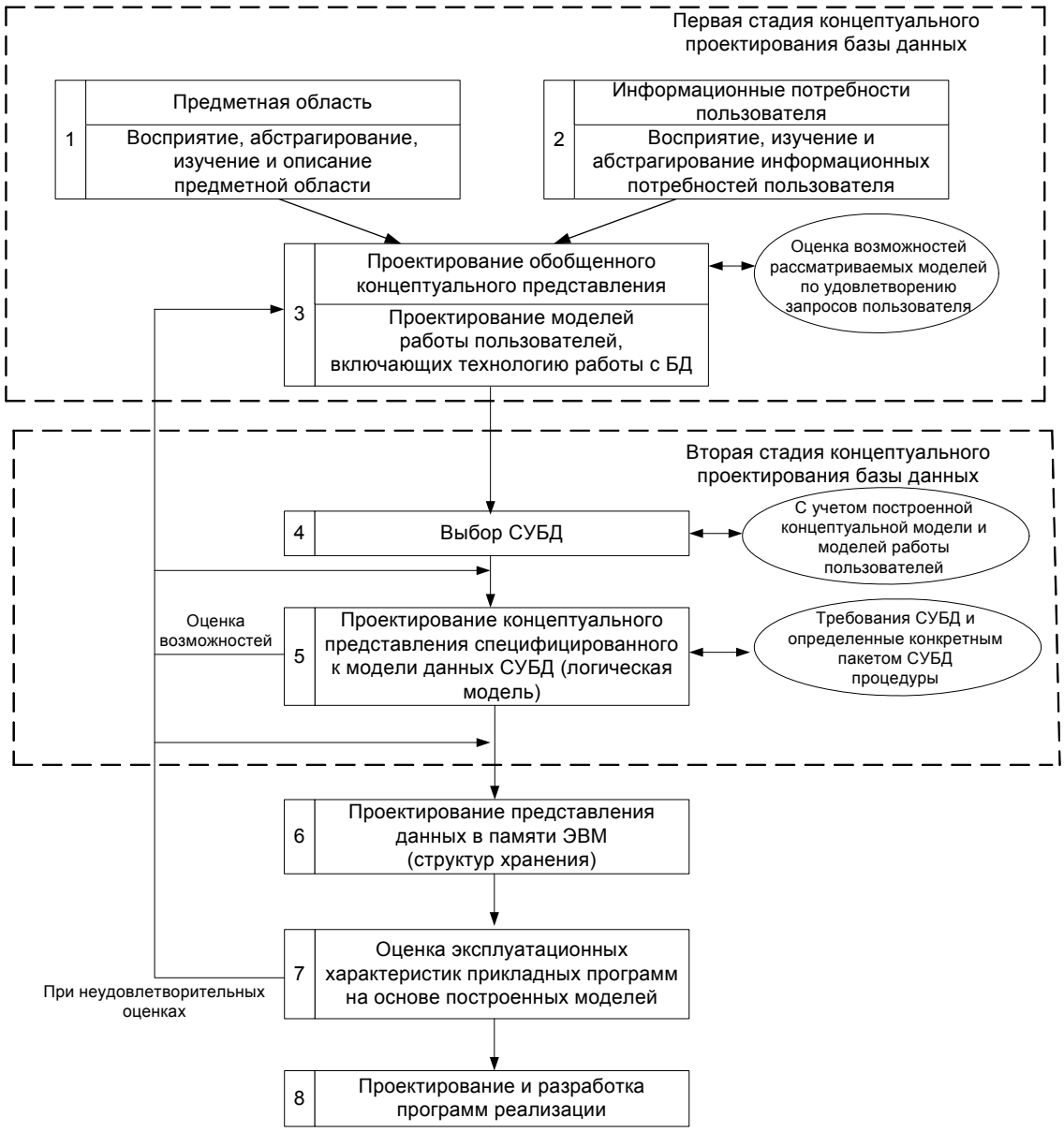


Рис. 1.5. Этапы проектирования базы данных

1.2.1. Первая стадия концептуального проектирования базы данных. ER-диаграмма

Рассмотрим основные подходы к созданию инфологической модели предметной области.

Функциональный подход к проектированию БД

Этот метод реализует принцип «от задач» и применяется тогда, когда известны функции некоторой группы лиц и/или

комплекса задач, для обслуживания информационных потребностей которых создаётся рассматриваемая БД.

Предметный подход к проектированию БД

Предметный подход к проектированию БД применяется в тех случаях, когда у разработчиков есть чёткое представление о самой ПО и о том, какую именно информацию они хотели бы хранить в БД, а структура запросов не определена или определена не полностью. Тогда основное внимание уделяется исследованию ПО и наиболее адекватному её отображению в БД с учётом самого широкого спектра информационных запросов к ней.

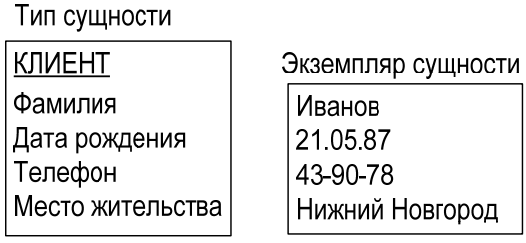
Проектирование с использованием метода «Сущность-Связь»

Метод «Сущность-Связь» (Entity-Relation, ER-method) является комбинацией двух предыдущих и обладает достоинствами обоих. Этап инфологического проектирования начинается с моделирования ПО. Проектировщик разбивает её на ряд локальных областей, каждая из которых (в идеале) включает в себя информацию, достаточную для обеспечения запросов отдельной группы будущих пользователей или решения отдельной задачи (подзадачи). Каждое локальное представление моделируется отдельно, затем они объединяются.

Выбор локального представления зависит от масштабов ПО. Обычно она разбивается на локальные области таким образом, чтобы каждая из них соответствовала отдельному внешнему приложению и содержала 6–7 сущностей.

Сущность – это любой различимый объект ПО, о котором в системе будет накапливаться информация. Сущности бывают как физически существующие (например, *КЛИЕНТ* или *АВТОМОБИЛЬ*), так и абстрактные (например, *ЭКЗАМЕН* или *ДИАГНОЗ*).

Для сущностей различают тип сущности и экземпляр. Тип характеризуется именем и списком свойств, а экземпляр – конкретными значениями свойств.



Типы сущностей можно классифицировать как сильные и слабые. Сильные сущности существуют сами по себе, а существование слабых сущностей зависит от существования сильных. Например, КЛИЕНТ – сильная сущность, а ЗВОНКИ этого клиента – слабая, которая зависит от наличия соответствующего клиента. Слабые сущности называют подчинёнными (дочерними), а сильные – базовыми (основными, родительскими).

Информация о сущности представляется совокупностью атрибутов.

Атрибут – поименованное свойство (характеристика) сущности.

Различают:

1. *Идентифицирующие и описательные атрибуты*. Идентифицирующие атрибуты имеют уникальное значение для сущностей данного типа и являются потенциальными ключами. Они позволяют однозначно распознавать экземпляры сущности. Из потенциальных ключей выбирается один первичный ключ (ПК). В качестве ПК обычно выбирается потенциальный ключ, по которому чаще происходит обращение к экземплярам записи. Кроме того, ПК должен включать в свой состав минимально необходимое для идентификации количество атрибутов. Остальные атрибуты называются описательными и заключают в себе интересующие свойства сущности.

2. *Составные и простые атрибуты*. Простой атрибут состоит из одного компонента, его значение неделимо. Составной атрибут является комбинацией нескольких компонентов, возможно, принадлежащих разным типам данных (например, ФИО или адрес). Решение о том, использовать составной атрибут или разбивать его на компоненты, зависит от характера его обработки и формата пользовательского представления этого атрибута.

3. *Однозначные и многозначные атрибуты* (могут иметь соответственно одно или много значений для каждого экземпляра сущности).

4. *Основные и производные атрибуты*. Значение основного атрибута не зависит от других атрибутов. Значение производного атрибута вычисляется на основе значений других атрибутов

(например, возраст студента вычисляется на основе даты его рождения и текущей даты).

Таким образом, атрибут представляет информационное описание количественных или качественных свойств сущности, описывает состояние сущности, позволяет идентифицировать сущность.

Спецификация атрибута состоит из его названия, указания типа данных и описания ограничений целостности – множества значений (или домена), которые может принимать данный атрибут.

Соответствующие взаимоотношения сущностей выражаются связями. Каждая связь характеризуется именем, степенью, типом и обязательностью.

Различают факультативные и обязательные связи.

Связь является *обязательной,* если в ней должен участвовать каждый экземпляр сущности, *факультативной* – если не каждый экземпляр сущности должен участвовать в данной связи. При этом связь может быть *обязательной,* с одной стороны, и *факультативной* – с другой.

Связь может затрагивать несколько типов сущностей. Число типов сущностей, участвующих в связи, называется степенью связи n = 2, 3, … Так, например, тип сущности СТУДЕНТ связан с типом сущности ФАКУЛЬТЕТ связью «учится на факультете». Степень этой связи равна двум. При n=2 связь называется бинарной. Заметим, что связь нужно рассматривать как двустороннюю: «студент учится на факультете» (обязательная) и «на факультете учатся студенты» (факультативная). Примером тернарной связи является связь типа экзамен между сущностями ДИСЦИПЛИНА, СТУДЕНТ, ПРЕПОДАВАТЕЛЬ. Из последнего примера видно, что связь также может иметь атрибуты (в данном случае это «Дата проведения» и «Оценка»).

Рассмотрим классификацию бинарных связей. В зависимости от того, сколько экземпляров сущности одного типа связаны со сколькими экземплярами сущности другого типа, различают следующие **типы связей:**

- Связь *1:1*. Каждый экземпляр сущности А связан с не более чем одним экземпляром сущности В, и наоборот. Приме-

ром является связь между типами сущностей СПЕЦИАЛЬНОСТЬ и УЧЕБНЫЙ ПЛАН ПО СПЕЦИАЛЬНОСТИ (каждой специальности соответствует свой учебный план по специальности).

- Связь *1:M*. Каждый экземпляр сущности А связан со многими экземплярами сущности В, но каждый экземпляр сущности В связан с не более чем одним экземпляром сущности А. Примером является связь между типами сущностей СПЕЦИАЛЬНОСТЬ и СТУДЕНТ (на одной специальности учатся много студентов, а каждый студент учится на одной специальности).
- Связь *M*:*N*. Несколько экземпляров сущности А связаны с несколькими экземплярами сущности В, и наоборот. Примером является связь между типами сущностей ФАКУЛЬТЕТ и СПЕЦИАЛЬНОСТЬ (на факультете может быть несколько специальностей и одна и та же специальность может быть на нескольких факультетах).

Числа, описывающие типы бинарных связей (*1:1, 1:M, M:N*), обозначают максимальное количество сущностей на каждой стороне связи. Эти числа называются максимальными кардинальными числами, а соответствующая пара чисел – максимальной кардинальностью.

ER-диаграмма, содержащая различные типы связей, приведена на рис. 1.6.

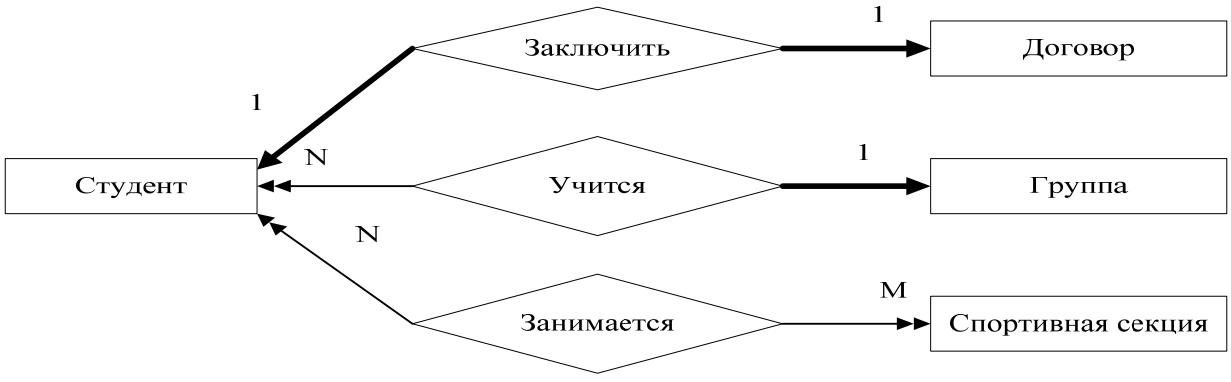


Рис. 1.6. ER-диаграмма с примерами типов множественных связей

Обратите внимание, что обязательные связи на рис. 1.6 выделены жирной линией.

Пример ER-диаграммы с указанием сущностей, их атрибутов и связей приведен на рис. 1.7.

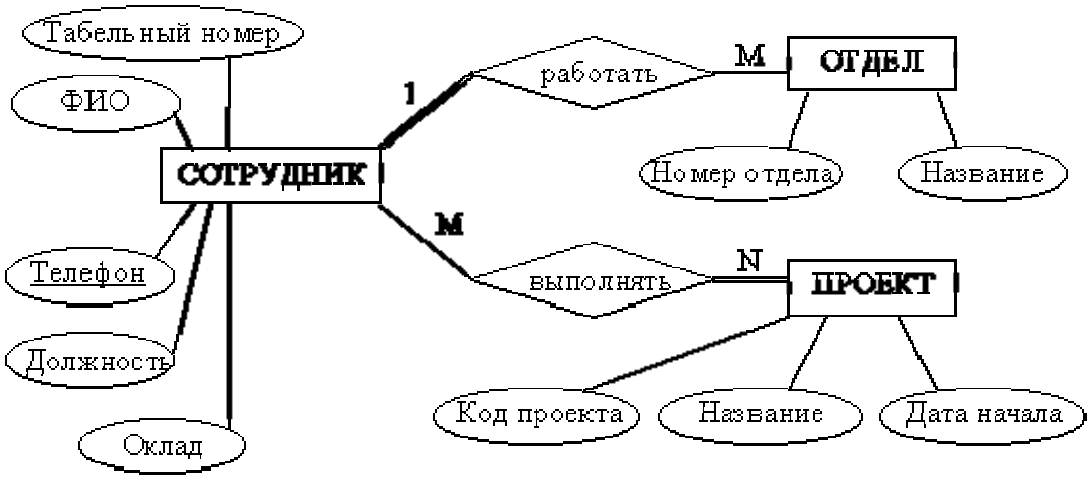


Рис. 1.7. Пример ER-диаграммы с однозначными
и многозначными атрибутами

Прежде всего необходимо отметить, что построенная модель должна удовлетворять ряду требований:

- адекватно отражать представление пользователя о данных;
- давать возможность ответа на возможные запросы пользователя, причем делать это с минимальными затратами по количеству просматриваемых сущностей;
- представлять данные с минимальным дублированием.

Процесс построения модели, удовлетворяющей указанным требованиям, является творческим, и формализовать его, как правило, невозможно.

1.2.2. Объединение локальных представлений

После того как созданы локальные представления, выполняется их объединение. При небольшом количестве локальных областей (не более пяти) они объединяются за один шаг. В противном случае обычно выполняют бинарное объединение в несколько этапов.

На этапе объединения необходимо выявить и устранить все противоречия. Например, одинаковые названия семантически различных объектов или связей или несогласованные ограничения целостности на одни и те же атрибуты в разных приложениях. Устранение противоречий вызывает необходимость воз-

врата к этапу моделирования локальных представлений с целью внесения в них соответствующих изменений.

Объединение локальных моделей производится следующими путями:

- слияние идентичных элементов;
- установление связей между наборами сущностей разных моделей;
- введение новых агрегированных элементов для представления связей между элементами разных моделей;
- обобщение различных подобных типов сущностей, позволяющее трактовать эти сущности как одну обобщенную сущность.

Рассмотрим каждый из этих путей.

Слияние идентичных элементов

Два или более элементов модели идентичны, если они имеют одинаковое смысловое значение.

Объединение моделей с идентичными элементами осуществляется путем «слияния» этих элементов в один. Два набора сущностей СПЕЦИАЛЬНОСТЬ в модели 1 и 2 имеют одинаковое смысловое значение (рис. 1.8) и могут быть заменены одним набором сущностей (рис. 1.9).

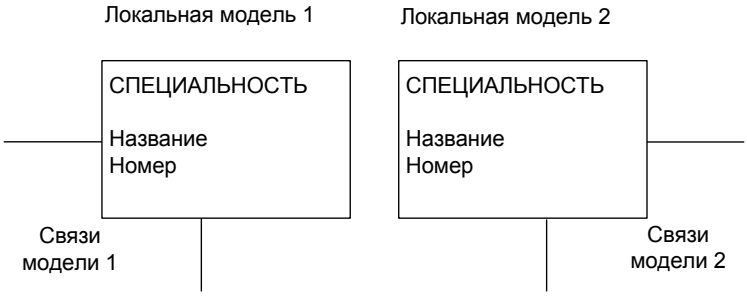


Рис. 1.8. Модели с идентичным элементом

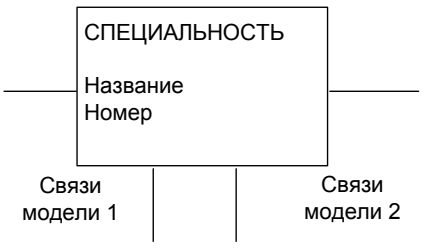


Рис. 1.9. Объединенная модель

Установление связей между наборами сущностей разных моделей

При рассмотрении наборов сущностей объединяемых моделей необходимо выявление связей между ними, т. к. именно эти связи и определяют в конечном итоге интегрированную базу данных.

Введение агрегированных элементов

При объединении моделей связь между элементами разных моделей может рассматриваться как новый элемент.

Рассмотрим в качестве примера моделирование информационного представления сдачи студентом экзаменов. Можно выделить ряд локальных представлений (рис. 1.10).

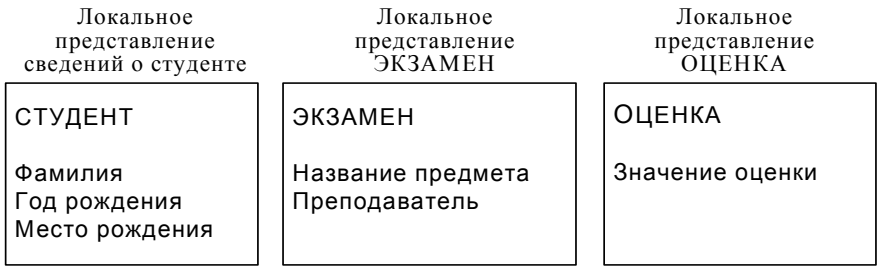


Рис. 1.10. Локальные представления

Объединяя локальные представления, устанавливаем новые связи (рис. 1.11).

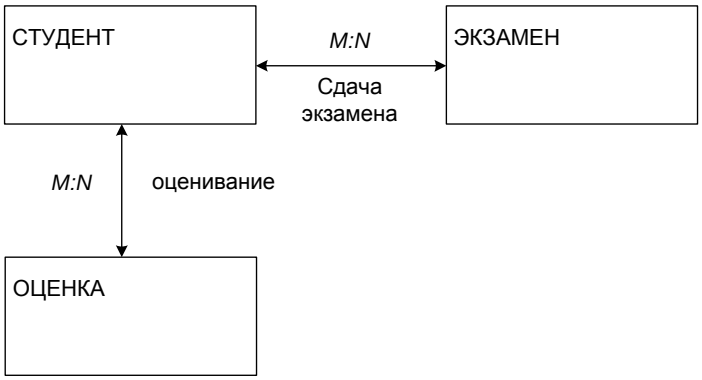


Рис. 1.11. Объединение локальных представлений

Как уже отмечалось, одним из показателей «зрелости» модели является возможность ответа на запросы пользователей, и установление связей преследует именно эту цель. Нетрудно видеть, что какие бы связи в рассматриваемой модели ни вводились, невозможно ответить на запрос «какую оценку получил

студент А по дисциплине В». В таком случае необходимо использовать принцип агрегации – необходимую связь между элементами модели ввести как некоторый новый элемент. В данном примере можно определить этот новый агрегированный элемент как ЭКЗАМЕН СТУДЕНТА (рис. 1.12).

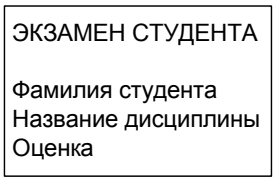


Рис. 1.12. Агрегированный элемент

Далее процесс объединения локальных моделей продолжается обычным образом.

Обобщение подобных типов сущностей

Рассмотрим локальные модели разных факультетов, например: модель факультета математики (М), модель факультета ИВТ и так далее. В локальную модель факультета М входят сущности СПЕЦИАЛЬНОСТИ ФАКУЛЬТЕТА М и СТУДЕНТЫ ФАКУЛЬТЕТА М, в локальную модель факультета входят, соответственно ИВТ, сущности СПЕЦИАЛЬНОСТИ ФАКУЛЬТЕТА ИВТ и СТУДЕНТЫ ФАКУЛЬТЕТА ИВТ (рис. 1.13).

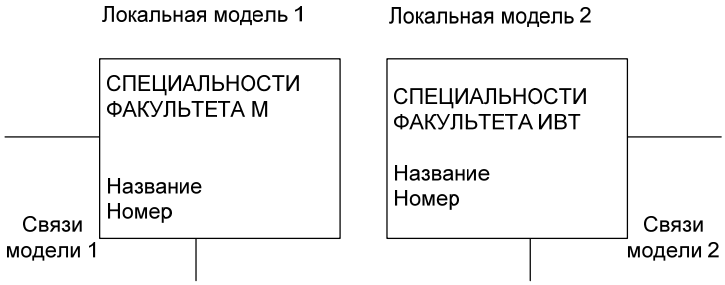


Рис. 1.13. Модели с подобным элементом

Два набора сущностей СПЕЦИАЛЬНОСТИ ФАКУЛЬТЕТА М и СПЕЦИАЛЬНОСТИ ФАКУЛЬТЕТА ИВТ в моделях 1 и 2 имеют одинаковое смысловое значение и могут быть заменены одним набором сущностей с добавлением нового атрибута – название факультета (рис. 1.14).

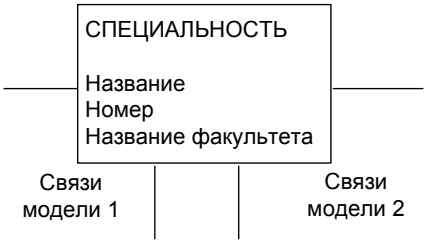


Рис. 1.14. Пример обобщенной сущности

Отметим, что в данном случае подобным образом можно слить и все остальные сущности локальных моделей факультетов, так как сущности СТУДЕНТЫ ФАКУЛЬТЕТА ИВТ и СТУДЕНТЫ М также имеют одинаковое смысловое значение и их также можно объединить. Однако в общем случае каждая локальная модель может содержать сущности и связи, которых нет в других локальных моделях.

В процессе объединения локальных представлений, как и при локальном моделировании, производится редактирование наименований (т. к. здесь появляются новые наименования). Полученное в результате объединения локальных представлений обобщенное представление и является концептуальной моделью.

1.2.3. Ограничения целостности

Под целостностью базы данных понимается то, что в ней содержится полная, непротиворечивая и адекватно отражающая предметную область информация.

Огромный объем данных, вводимых в БД (причем разные данные могут вводиться разными пользователями), обусловливает большое число ошибок ввода (занесения). Человек, работая с определенными данными, неявно использует для контроля имеющиеся у него представления об этих данных. Например, сотрудник отдела кадров, увидев в карточке работника год рождения 1693, сразу заметит эту ошибку и предположит, что просто переставлены две цифры и реальный год рождения 1963. То есть в представлениях сотрудника заключены некоторые логические ограничения на данные. Очевидно, что для контроля правильности вводимых данных при работе с БД целесообразно сформировать и использовать ограничения.

Соответствующие ограничения обычно разделяют на 3 группы: внешние, специально конструируемые и внутренние. К предметной области относятся первые две группы, которые мы кратко охарактеризуем в этом подразделе. Внутренние ограничения относятся уже к модели данных и будут рассматриваться в разделе, посвященном модели данных.

Внешние ограничения

Эти ограничения связаны с адекватностью отражения предметной области. Например, сотрудник организации не может быть моложе 17 и старше 90 лет. Соответствующее ограничение на год рождения (GR) можно записать следующим образом:

Текущий год – 17 > GR > Текущий год – 90.

Одним из способов задания таких ограничений является перечисление конечного множества допустимых значений какого-либо атрибута (так называемый перечислимый тип данных). Например, должность преподавателя в вузе может принимать одно из следующих значений: профессор, доцент, старший преподаватель, преподаватель, ассистент. Вводимое значение должности для конкретного экземпляра, не совпадающее с одним из перечисленных значений, является ошибкой.

Ограничения, описанные с помощью специальных конструкций

Например, в БД вуза вводятся данные о числе студентов и преподавателей. По нормативным документам задано конкретное значение отношения числа студентов к числу преподавателей. Проверку этого отношения можно использовать для контроля достоверности данных. Такие конструкции строятся исходя из специфики данных рассматриваемой предметной области. Можно, например, построить много конструкций следующего вида: сумма значений по заданному атрибуту по всем экземплярам сущностей должна совпадать со значением определенного атрибута в экземпляре другой сущности.

Таким образом, на стадии ER-моделирования для повышения достоверности данных необходимо сформулировать соответствующие ограничения на данные. В идеальном случае каждое зна-

чение атрибута должно каким-то образом контролироваться. Использование этих ограничений позволяет существенно повысить достоверность данных в БД.

1.2.4. Представление концептуальной модели средствами модели данных СУБД

Общие представления о моделях данных СУБД

В соответствии с основными этапами проектирования БД после построения концептуальной модели выбирается СУБД, с помощью которой будет организована БД и работа с ней. Каждая СУБД поддерживает определенные виды и типы данных, а также средства представления связей между данными, составляющими модель данных СУБД. ***Этот этап часто называют логическим проектированием БД. Полученная при этом модель часто также называется концептуальной моделью или схемой (но специфицированной к понятиям модели данных СУБД). В некоторых источниках полученную модель называют логической структурой данных, или моделью данных БД.***

Можно по-разному характеризовать понятие **модели данных СУБД**. С одной стороны, ***модель данных СУБД – это способ структурирования данных, которые рассматриваются как некоторая абстракция в отрыве от предметной области.*** С другой стороны, ***модель данных СУБД – это инструмент представления концептуальной модели предметной области и динамики ее изменения в виде БД.***

Учитывая обе вышеуказанные стороны, определим основные структуры моделей данных СУБД, используемые для представления концептуальной модели предметной области (сущностей, атрибутов, связей).

Элемент данных (поле) – наименьшая поименованная единица данных. Используется для представления значения атрибута.

С элементом данных неразрывно связано понятие «тип данных», который может принимать соответствующее поле. В разных СУБД могут использоваться разные типы данных, наиболее распространенными из которых (используемые во многих СУБД)

являются следующие: числовой (numeric), символьный (char), дата (date) и т. д.

Запись – поименованная совокупность полей. Используется для представления совокупности атрибутов сущности (записи о сущности).

Экземпляр записи – запись с конкретными значениями полей.

Первичный ключ – минимальный набор полей записи, однозначно идентифицирующих экземпляр записи файла.

Файл – поименованная совокупность экземпляров записей одного типа. Используется для представления однородного набора сущностей.

Набор файлов – поименованная совокупность файлов, обрабатываемых в системе. Используется для представления нескольких наборов сущностей.

Введем понятие «группа», обобщающее понятия «файл» и «запись».

Группа – это поименованная совокупность элементов данных или элементов данных и других групп.

Важнейшим понятием концептуальной модели является понятие связи между сущностями (наборами сущностей). В моделях данных СУБД соответствующее понятие отражается понятием «групповое отношение».

Групповое отношение – поименованное бинарное отношение, заданное на двух множествах экземпляров рассматриваемых групп. В групповом отношении один член группы назначается владельцем отношения, другой – членом.

База данных – поименованная совокупность экземпляров групп и групповых отношений.

Для представления группового отношения используется две формы:

а) теоретико-графовая. Группы изображаются вершинами графа, связи между группами – дугами, направленными от группы-владельца к группе-члену с указанием имени отношения и коэффициента.

По типу графов различают:

- *иерархическую модель (**граф без циклов – дерево**);*

- *сетевую модель (**ориентированный граф общего вида**);*

б) теоретико-множественная. Связь между группами изображается таблицей, столбцы которой представляют ключи соответствующих групп. Для формального описания таблицы используется математическое (теоретико-множественное) понятие отношения. Соответствующая модель данных называется реляционной моделью.

Модель данных СУБД описывается следующим образом:

- определены возможные типы и характеристики логических структур данных (полей, записей, файлов);
- заданы правила составления структур более общего типа из структур более простых типов (например, записей из полей, файлов из записей и т. д.);
- определен способ представления связей (отношений) между файлами и записями с помощью дополнительных полей;
- определены возможные действия над структурами и правила их выполнения, включающие:
  - основные элементарные операции над данными (поиск записи с заданным значением ключа, чтение нужной записи, добавление записи, корректировка, удаление);
  - обобщенные операции (процедуры) (процедуры копирования БД, восстановления БД, процедуры, вычисляющие значения определенных атрибутов в БД по значениям других атрибутов, и т. п.);
  - средства контроля относительно простых условий корректности операций добавления, обновления или удаления данных (ограничения), реализуемые автоматически запускаемыми при выполнении вышеуказанных операций специальными процедурами (триггерами);
  - средства контроля сколь угодно сложных условий корректности выполнения определенных действий (правила);
  - специальный класс процедур (триггеры).

1.3. Реляционная модель данных

Теоретической основой этой модели стала теория отношений, основу которой заложили два логика – американец Чарльз

Содерс Пирс (1839–1914) и немец Эрнст Шредер (1841–1902). В руководствах по теории отношений было показано, что множество отношений замкнуто относительно некоторых специальных операций, то есть образует вместе с этими операциями абстрактную алгебру. Это важнейшее свойство отношений было использовано в реляционной модели для разработки языка манипулирования данными, связанного с исходной алгеброй. Американский математик Э. Ф. Кодд в 1970 году впервые сформулировал основные понятия и ограничения реляционной модели, ограничив набор операций в ней семью основными и одной дополнительной операцией.

Основной структурой данных в модели является отношение, именно поэтому модель получила название ***реляционной*** (от английского *relation* – отношение).

N-арным отношением R называют подмножество декартова произведения $D_1 \times D_2 \times ... \times D_n$ множеств $D_1, D_2, \dots, D_n$ ***(n > 1),*** необязательно различных. Исходные множества $D_1, D_2, \dots, D_n$ называют в модели ***доменами***.

$R \subseteq D_1 \times D_2 \times ... \times D_n$, где $D_1 \times D_2 \times ... \times D_n$ – полное декартово произведение.

Например, имеем три домена: D_1 содержит три фамилии, D_2 – набор из двух учебных дисциплин и D_3 – набор из трех оценок. Допустим, содержимое доменов следующее:

- D_1 = {Иванов, Крылов, Степанов};
- D_2 = {Теория автоматов, Базы данных};
- D_3 = {3, 4, 5}

Тогда полное декартово произведение содержит набор из 18 троек, где первый элемент – это одна из фамилий, второй – это название одной из учебных дисциплин, а третий – одна из оценок.

<Иванов, Теория автоматов,3>; <Иванов, Теория автоматов,4>; <Иванов, Теория автоматов,5>; <Крылов, Теория автоматов,3>; <Крылов, Теория автоматов,4>; <Крылов, Теория автоматов,5>; <Степанов, Теория автоматов,3>; <Степанов, Теория автоматов,4>; <Степанов, Теория автоматов,5>; <Иванов, Базы данных,3>; <Иванов, Базы данных,4>; <Иванов, Базы данных,5>; <Крылов, Базы данных,3>; <Крылов, Базы данных,4>; <Крылов,

Базы данных,5>; <Степанов, Базы данных,3>; <Степанов, Базы данных,4>; <Степанов, Базы данных,5>;

Отношение R моделирует реальную ситуацию, и оно может содержать, допустим, только 5 строк, которые соответствуют результатам сессии (Крылов экзамен по «Базам данных» еще не сдавал):

<Иванов, Теория автоматов,4>; <Крылов, Теория автоматов,5>;
<Степанов, Теория автоматов,5>; <Иванов, Базы данных,3>;
<Степанов, Базы данных,4>;

Отношение имеет простую графическую интерпретацию, оно может быть представлено в виде таблицы, столбцы которой соответствуют вхождениям доменов в отношение, а строки – наборам из *n* значений, взятых из исходных доменов, которые расположены в строго определенном порядке в соответствии с заголовком. Такие наборы из *n* значений часто называют n-ками.

R		
Фамилия	**Дисциплина**	**Оценка**
Иванов	Теория автоматов	4
Иванов	Базы данных	3
Крылов	Теория автоматов	5
Степанов	Теория автоматов	5
Степанов	Базы данных	4

Вхождение домена в отношение принято называть *атрибутом*. Строки отношения называются *кортежами*.

Количество атрибутов в отношении называется степенью, или рангом, отношения.

Следует заметить, что в отношении не может быть одинаковых кортежей, это следует из математической модели: отношение – это подмножество декартова произведения, а в декартовом произведении все n-ки различны.

Любое отношение является динамической моделью некоторого реального объекта внешнего мира. Поэтому вводится понятие экземпляра отношения, которое отражает состояние данного объекта в текущий момент времени, и понятие схемы отношения, которая определяет структуру отношения.

Схемой отношения *R* называется перечень имен атрибутов данного отношения с указанием домена, к которому они относятся:

$$S_R = (A_1, A_2, \dots, A_n), \text{где } A_i \in D_i.$$

Если атрибуты принимают значения из одного и того же домена, то они называются **θ – *сравнимыми***, где ***θ*** – множество допустимых операций сравнения, заданных для данного домена. Например, если домен содержит числовые данные, то для него допустимы все операции сравнения, тогда θ = {=, <>,>=,<=,<,>}. Однако и для доменов, содержащих символьные данные, могут быть заданы не только операции сравнения по равенству и неравенству значений. Если для данного домена задано лексикографическое упорядочение, то он имеет также полный спектр операций сравнения.

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

$S_{R1} = (A_1, A_2, ..., A_n)$ – схема отношения R_1.

$S_{R2} = (B_{i1}, B_{i2}, ..., B_{im})$ – схема отношения R_2 после упорядочения имен атрибутов.

Тогда

$$S_{R1} \sim S_{R2} \Leftrightarrow \begin{cases} 1.\ n=m, \\ 2.\ A_j, B_{ij} \in D_j \end{cases}$$

Ключом отношения R называется минимальное подмножество $K = \{A_{i1}, A_{i2}, \dots, A_{im}\} \subseteq \{A_1, A_2, \dots, A_n\}$, где $\{i1, i2, \dots, im\} \subseteq \{1, 2, \dots, n\}$, такое, что любые два различных кортежа $t_1, t_2 \in R$ $(t_1 \neq t_2)$ не совпадают по значениям множества $K = \{A_{i1}, A_{i2}, \dots, A_{im}\}$.

Возможны случаи, когда отношение ***R*** имеет несколько ключей. Такие ключи называются потенциальными (возможными). ***Выбранный из них ключ для идентификации кортежей называется первичным ключом***. Таким образом, достаточно знать значение кортежа на множестве *K*, чтобы однозначно его идентифицировать. Ключ используется для представления связей

между отношениями. С этой целью первичный ключ одного отношения включается в структуру (набор атрибутов) связанного с ним отношения. Для второго отношения соответствующий ключ называется ***внешним ключом***.

Совокупность схем отношений, используемых для представления концептуальной модели, называется схемой реляционной базы данных, а текущие значения соответствующих отношений – реляционной базой данных.

Как уже говорилось ранее, реляционная модель представляет БД в виде множества взаимосвязанных отношений. В этой модели поддерживаются иерархические связи между отношениями. В каждой связи одно отношение выступает как основное, а другое – в роли подчиненного. Это означает, что один кортеж основного отношения может быть связан с несколькими кортежами подчиненного отношения. Для поддержки этих связей оба отношения должны содержать наборы атрибутов, по которым они связаны. В основном отношении это первичный ключ отношения (PRIMARY KEY), который однозначно определяет кортеж основного отношения. В подчиненном отношении для моделирования связи должен присутствовать набор атрибутов, соответствующий первичному ключу основного отношения. Однако здесь этот набор атрибутов уже является внешним ключом (FOREIGN KEY), то есть он определяет множество кортежей подчиненного отношения, которые связаны с единственным кортежем основного отношения.

Рассмотрим правила преобразования ER-модели в реляционную.

1. Каждой сущности ставится в соответствие отношение реляционной модели данных.

2. Каждый атрибут сущности становится атрибутом соответствующего отношения. Для каждого атрибута задается конкретный допустимый в СУБД тип данных и обязательность или необязательность данного атрибута (то есть допустимость или недопустимость NULL значений для него).

3. Первичный ключ сущности становится PRIMARY KEY соответствующего отношения. Атрибуты, входящие в первичный

ключ отношения, автоматически получают свойство обязательности (NOT NULL).

4. В каждое отношение, соответствующее подчиненной сущности, добавляется набор атрибутов основной сущности, являющейся первичным ключом основной сущности. В отношении, соответствующем подчиненной сущности, этот набор атрибутов становится внешним ключом (FOREING KEY).

5. Для моделирования необязательного типа связи на физическом уровне у атрибутов, соответствующих внешнему ключу, устанавливается свойство допустимости неопределенных значений (признак NULL). При обязательном типе связи атрибуты получают свойство отсутствия неопределенных значений (признак NOT NULL)

6. Групповое отношение может представляться двумя способами. При первом способе в таблицы, соответствующие группам – членам отношения, добавляются столбцы ключевых полей (атрибутов) другого члена отношения (связь описывается через ключевые атрибуты). При втором способе групповое отношение определяется как дополнительная группа (дополнительная таблица). Столбцами этой дополнительной таблицы являются ключи групп – членов отношения. Таким образом, при любом способе соответствующая модель данных представляет собой совокупность структур таблиц.

В качестве основного недостатка реляционной модели можно указать дублирование информации при представлении связей.

Необходимо отметить, что большинство СУБД для персональных ЭВМ поддерживают именно реляционную модель данных. В качестве примеров таких наиболее распространенных СУБД можно указать все dBase-подобные системы: DB2, Paradox, Access, FoxPro, Oracle, MS SQL Server.

Отметим следующие свойства отношения:

1. Отношение имеет имя, которое отличается от имен всех других отношений.

2. Каждое значение элементов кортежей представляется простым (атомарным) типом данных.

3. Каждый атрибут имеет уникальное имя.

4. Значения всех атрибутов являются атомарными (неделимыми). Это следует из определения домена как множества значений простого типа данных, т. е. среди значений домена не могут содержаться множества.

5. Порядок рассмотрения атрибутов в схеме отношения (отношении) не имеет значения, т. к. для ссылки на значение атрибута в кортеже отношения всегда используется имя атрибута.

6. Порядок рассмотрения кортежей в отношении не имеет значения, т. к. отношение представляет собой множество кортежей, а элементы множества, по определению теории множеств, неупорядочены.

1.4. Манипулирование данными в реляционной модели

Для манипулирования данными в реляционной модели используются два формальных аппарата:

– реляционная алгебра, основанная на теории множеств;

– реляционное исчисление, базирующееся на исчислении предикатов первого порядка.

Механизмы реляционной алгебры и реляционного исчисления эквивалентны, т. е. для любого допустимого выражения реляционной алгебры можно построить эквивалентную формулу реляционного исчисления и наоборот. Отличаются два этих формальных аппарата уровнем процедурности. Выражения реляционной алгебры строятся на основе алгебраических операций (высокого уровня), и подобно тому, как интерпретируются арифметические и логические выражения, выражение реляционной алгебры также имеет процедурную интерпретацию. Другими словами, запрос, представленный на языке реляционной алгебры, может быть реализован как последовательность элементарных алгебраических операций с учетом их старшинства и возможного наличия скобок.

Для формулы реляционного исчисления однозначная интерпретация (соответствующая однозначная последовательность действий), вообще говоря, отсутствует. Формула только устанавливает условия, которым должны удовлетворять кортежи резуль-

тирующего отношения. Поэтому языки реляционного исчисления являются декларативными.

Операции, реализуемые с помощью указанных аппаратов, обладают важным свойством: они замкнуты на множестве отношений. Это означает, что выражения реляционной алгебры и формулы реляционного исчисления определяются над отношениями реляционных БД и результатом вычисления также являются отношения. В результате любое выражение или формула могут интерпретироваться как отношение, что позволяет использовать их в других выражениях или формулах.

И алгебра, и исчисление обладают большой выразительной мощностью, очень сложные запросы к базе данных могут быть выражены с помощью одного выражения реляционной алгебры или одной формулы реляционного исчисления. Именно по этой причине такие механизмы включены в реляционную модель данных. ***Конкретный язык манипулирования реляционными БД называется реляционно полным, если любой запрос, выражаемый с помощью одной операции реляционной алгебры или одной формулы реляционного исчисления, может быть выражен с помощью одного оператора этого языка.***

Заметим, что крайне редко алгебра или исчисление принимаются в качестве полной основы какого-либо языка БД. Обычно (как, например, в случае языка SQL) язык основывается на некоторой смеси алгебраических и логических конструкций. Тем не менее знание алгебраических и логических основ языков баз данных часто бывает полезно на практике.

1.4.1. Операции реляционной алгебры

Операции реляционной алгебры определены на множестве отношений и являются замкнутыми относительно этого множества (образуют алгебру).

Набор основных алгебраических операций состоит из восьми операций, которые делятся на два класса – теоретико-множественные операции и специальные реляционные, дополненные некоторыми специальными операциями, специфичными для баз данных.

В состав теоретико-множественных операций входят традиционные операции над множествами:

- объединение;
- пересечение;
- разность;
- декартово произведение.

Специальные реляционные операции включают:

- выборку;
- проекцию;
- естественное соединение;
- деление.

Операции объединения, пересечения и разности применяются к отношениям, совместимым по типу, или, другими словами, к отношениям с эквивалентными схемами.

Отношение R

ID_NUM	NAME	CITY	AGE
1809	Иванов	Москва	45
1996	Петров	Нижний Новгород	39
1777	Сидоров	Рязань	21

Отношение S

ID_NUM	NAME	CITY	AGE
1809	Иванов	Москва	45
1896	Галкин	Иваново	40

Объединением двух совместимых по типу отношений R и S ($R \cup S$) называется отношение с тем же заголовком, как в отношениях R и S, и с телом, состоящим из множества кортежей t, принадлежащих R или S или обоим отношениям.

$R \cup S$

ID_NUM	NAME	CITY	AGE
1809	Иванов	Москва	45
1996	Петров	Нижний Новгород	39
1777	Сидоров	Рязань	21
1896	Галкин	Иваново	40

При выполнении операции объединения двух отношений создается отношение, включающее кортежи, входящие хотя бы в одно из отношений-операндов. Обратите внимание, что повторяющиеся кортежи удаляются по определению отношения.

Пересечением двух совместимых по типу отношений R и S ($R \cap S$) называется отношение с тем же заголовком, как в отно-

шениях R и S, и с телом, состоящим из множества кортежей t, принадлежащих одновременно обоим отношениям R и S**.**

$R \cap S$

ID_NUM	NAME	CITY	AGE
1809	Иванов	Москва	45

Разностью двух совместимых по типу отношений R и S (R – S) называется отношение с тем же заголовком, как в отношениях R и S, и с телом, состоящим из множества кортежей t, принадлежащих отношению R и не принадлежащих отношению S**.**

R – S

ID_NUM	NAME	CITY	AGE
1996	Петров	Нижний Новгород	39
1777	Сидоров	Рязань	21

Декартово произведение двух отношений R и S ($R \times S$) определяется как отношение с заголовком, представляющим собой сцепление двух заголовков исходных отношений R и S, и телом, состоящим из множества кортежей t, таких, что первым является любой кортеж отношения R, а вторым – любой кортеж, принадлежащий отношению S.

Пусть отношение R содержит имена всех текущих поставщиков, а отношение B – номера всех текущих деталей. Тогда $R \times S$ – это все текущие пары поставщик – деталь и деталь – поставщик.

Отношение R

S1
S2
S3

Отношение S

P1
P2
P3
P4

$R \times S$

S1	P1
S1	P2
S1	P3
S1	P4

S2	P1
S2	P2
S2	P3
S2	P4

S3	P1
S3	P2
S3	P3
S3	P4

На практике явное использование операции декартово произведение требуется только для очень сложных запросов. Эта опе-

рация включена в реляционную алгебру по концептуальным соображениям: (декартово произведение требуется как промежуточный шаг при определении операции θ-соединения, которая используется довольно часто).

Выборка – это сокращенное название θ-выборки, где θ означает любой скалярный оператор сравнения ($=, \neq, \leq, \geq$).

θ-выборкой из отношения R по атрибутам X и Y ($\sigma_{X\theta Y}(R)$) называется отношение, имеющее тот же заголовок, что и отношение R, и тело, содержащее множества кортежей t отношения R, для которых проверка условия X θ Y дает значение истина. Атрибуты X и Y должны быть определены на одном и том же домене, а оператор должен иметь смысл для этого домена.

Операция выборка (или операция ограничение отношения) создает новое отношение, содержащее только те строки отношения-операнда, которые удовлетворяют некоторому условию ограничения.

Отношение R

ID_NUM	NAME	CITY	AGE
1809	Иванов	Москва	45
1996	Петров	Нижний Новгород	39
1777	Сидоров	Рязань	21
1896	Галкин	Москва	30

$\sigma_{\text{CITY} = \text{'Москва'}}(R)$

ID_NUM	NAME	CITY	AGE
1809	Иванов	Москва	45
1896	Галкин	Москва	30

$\sigma_{CITY = 'Москва' and\ AGE < 40}(R)$

ID_NUM	NAME	CITY	AGE
1896	Галкин	Москва	30

Проекцией отношения R по атрибутам X, Y,…,Z ($\pi_{[X, Y,...Z]}(R)$), где каждый из атрибутов принадлежит отношению R, называется отношение с заголовком {X, Y,…,Z} и с телом, содержащим множество всех кортежей вида <X:x, Y:y, ..., Z:z> таких, что в отношении R имеется кортеж, атрибут X которого имеет значение x, атрибут Y имеет значение y, ..., атрибут Z имеет значение z. Тем самым при выполнении операции проек-

ции получается «вертикальное» подмножество данного отношения, то есть подмножество, получаемое исключением всех атрибутов, отношения-операнда с естественным уничтожением потенциально возникающих кортежей-дубликатов.

$\pi_{[NAME, CITY]}(R)$

NAME	CITY
Иванов	Москва
Петров	Нижний Новгород
Сидоров	Рязань
Галкин	Москва

$\pi_{[CITY]}(R)$

CITY
Москва
Нижний Новгород
Рязань

Соединение отношений. Эта операция создает новое отношение, каждый кортеж которого является результатом сцепления кортежей операндов (исходных отношений). Соединение имеет две разновидности: естественное соединение и соединение по условию (θ-соединение).

Пусть X={X1, X2, …, Xm}, Y={Y1, Y2, …, Yn}, Z={Z1, Z2, …, Zk}.

Естественным соединением отношений R(X,Y) и S(Y,Z) (R ⋈ S) называется отношение с заголовком {X, Y, Z} и с телом, содержащим множество всех кортежей вида <X:x, Y:y, Z:z>, таких, для которых в отношении R значение атрибута X равно x, а значение атрибута Y равно y, и в отношении S значение атрибута Y равно y, а атрибута Z равно z. При естественном соединении производится сцепление строк операндов соединения по общим атрибутам при условии равенства общих атрибутов.

Замечание 1. Соединения не всегда выполняются по внешнему ключу и соответствующему первичному ключу, хотя такие соединения очень распространены и являются важным частным случаем.

Замечание 2. Если отношения R и S не имеют общих атрибутов, то выражение $R \bowtie S$ В эквивалентно R ×S.

Отношение R (поставщики)

ID_NUM	NAME	CITY	STATUS
1809	Иванов	Москва	20
1996	Петров	Нижний Новгород	15
1777	Сидоров	Рязань	10

Отношение S (детали)

IP_NUM	NAMEN	CITY	WEIGHT
P123	Болт	Москва	12
P896	Гайка	Нижний Новгород	14
P432	Шарнир	Москва	15

$(R \bowtie S)$

ID_NUM	NAME	STATUS	CITY	IP_NUM	NAMEN	CITY	WEIGHT
1809	Иванов	20	Москва	P123	Болт	Москва	12
1809	Иванов	20	Москва	P432	Шарнир	Москва	15
1996	Петров	15	Нижний Новгород	P896	Гайка	Нижний Новгород	14

θ-соединение. Пусть отношения R и S не имеют общих имен атрибутов, и θ определяется так же, как в операции выборки. θ-соединением отношения R по атрибуту X с отношением R по атрибуту Y называется результат вычисления выражения $\sigma_{X\,\theta\,Y}(R \times S)$.

θ-соединение – это отношение с тем же заголовком, что и при декартовом произведении отношений R и S, и с телом, содержащим множество кортежей $t \in R \times S$, таких, что вычисление условия X θ Y дает значение истина для данного кортежа. Атрибуты X и Y должны быть определены на одном и том же домене, а оператор должен иметь смысл для этого домена.

Отношение R (поставщики)

ID_NUM	NAME	CITY	STATUS
1809	Иванов	Москва	20
1996	Петров	Нижний Новгород	15
1777	Сидоров	Рязань	10

Отношение S (поставки)

ID_NUM	IP_NUM	QTY
1809	P123	100
1809	P896	200
1777	P432	150
1996	P432	150
1996	P123	250

$\sigma_{\text{QTY}<200 \text{ and } \text{R.ID\_NUM}=\text{S.ID\_NUM}}(R \times S)$

R.ID_NUM	NAME	CITY	STATUS	S.ID_NUM	IP_NUM	QTY
1809	Иванов	Москва	20	1809	P123	100
1996	Петров	Нижний Новгород	15	1996	P432	150
1777	Сидоров	Рязань	10	1777	P432	150

Операция деления

У операции реляционного деления два операнда – бинарное и унарное отношения.

Пусть X={X1, X2, …, Xm}, Y={Y1, Y2, …, Yn}.

Делением отношений R (X,Y) на S(Y) (R/S) называется отношение с заголовком {X} и телом, содержащим множество

всех кортежей {X:x}, таких, что существует кортеж {X:x, Y:y}, который принадлежит отношению R для всех кортежей {Y:y}, принадлежащих отношению S.

Деление отношений создает новое отношение, содержащее атрибуты первого отношения, отсутствующие во втором отношении и кортежи, которые при сцеплении с кортежами второго отношения будут принадлежать первому отношению. Для выполнения этой операции второе отношение должно содержать лишь атрибуты, совпадающие с атрибутами первого.

Отношение A

S#	P#
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P3
S3	P2
S3	P3

Отношение B

P#
P1

A/B
S1
S2

Отношение B1

P#
P2
P3

A/B1
S1
S3

Отношение B2

P#
P1
P2
P3

A/B2
S1

Замечание. Операция деления полезна тогда, когда запрос содержит слово «все».

Кроме рассмотренных выше операций в состав алгебры включаются:

- операция присваивания, позволяющая сохранить в БД результаты вычисления алгебраических выражений (A:= B),

- операция переименования атрибутов, дающая возможность корректно сформировать заголовок (схему) результирующего отношения.

Пример использования реляционных операторов

Информация о поставщиках, деталях и поставках содержится в трех отношениях.

Отношение Поставщики (P) содержит номер поставщика, его имя, город и статус.

P (P#, PNAME, CITY, STATUS).

Отношение Детали (D) содержит информацию о коде детали, наименовании, весе, цвете и месте хранения.

D (D#, DNAME, WEIGHT, COLOR, CITY).

Отношение Поставка (PD) содержит сведения о номере поставщика, коде детали и количестве.

PD (P#, D#, QTY).

Необходимо получить имена поставщиков, которые не поставляют деталь с кодом D2.

Рассмотрим пошаговое решение этого запроса:

1. Найдем коды поставщиков детали с кодом D2. Для этого из отношения PD выберем кортежи, в которых код детали равен D2. Получим отношение R1 с той же самой структурой, что и исходное отношение PD:

$R1 := \sigma_{D\# = 'D2'}(PD)$.

Возьмем проекцию отношения R1 по атрибуту P#. Получим отношение R2 с одним атрибутом:

$R2 := \pi_{[P\#]}(R1)$.

2. Найдем поставщиков, не выпускающих детали с кодом D2. Для этого возьмем проекцию отношения P по атрибуту P#. Получим отношение R3 с одним атрибутом:

$R3 := \pi_{[P\#]}(P)$.

Разность отношений R3 и R2 даст номера тех поставщиков, которые не поставляют деталь с кодом D2.

R4 := R3 – R2.

3. Операция естественного соединения отношений R4 и P по атрибуту P# позволяет сформировать отношение R5 с такой же структурой, что и отношение P, но кортежи этого отношения будут содержать информацию лишь о тех поставщиках, которые не поставляют деталь D2:

$R5 := (R4 \bowtie P)$.

Выполним проекцию отношения R5 по атрибуту PNAME. Получим искомое отношение, содержащее имена поставщиков:

$R6 := \pi_{[PNAME]}(R5)$.

Выразим данный запрос в виде одной формулы:

$$\pi_{[PNAME]}\left((\pi_{[P\#]}(P) - \pi_{[P\#]}(\sigma_{D\#='D2'}(PD))) \bowtie P\right).$$

1.5. Процесс нормализации отношений

При представлении концептуальной схемы в виде реляционной модели возможны различные варианты выбора схем отношений. От правильного выбора схем отношений, представляющих концептуальную схему, в значительной степени будет зависеть эффективность функционирования БД.

Рассмотрим для примера конкретную схему отношений и проанализируем её недостатки. Предположим, что данные о студентах, факультетах, специальностях, включены в таблицу со следующей схемой отношения:

СТУДЕНТ
(Код студента, Фамилия, Название факультета,
Название специальности).

Эта схема отношений обусловливает следующие недостатки соответствующей БД:

- Дублирование информации (избыточность). У студентов, обучающихся на одном факультете, будет повторяться название факультета. Для разных факультетов будут повторяться специальности.
- Потенциальная противоречивость (аномалии обновления). Если, например, изменится название специальности, то изменяя её в одном кортеже (у одного студента), необходимо изменять и во всех других кортежах, где она присутствует.
- Потенциальная возможность потери сведений (аномалии удаления). При удалении информации обо всех студентах, поступающих на определенную специальность, мы теряем все сведения об этой специальности.
- Потенциальная возможность невключения информации в БД (аномалии включения). В БД будут отсутствовать сведения о специальности, если на ней нет обучающихся студентов.

В теории реляционных БД существует формальный метод построения реляционной модели БД, в которой отсутствует избы-

точность и аномалии обновления, удаления и включения. Это метод называется нормализация.

Понятие функциональной зависимости является фундаментальным в теории нормализации реляционных БД. Функциональные зависимости определяют устойчивые отношения между объектами и их свойствами в рассматриваемой предметной области. Именно поэтому процесс поддержки функциональных зависимостей, характерных для данной предметной области, является базовым для процесса проектирования.

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

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

В теории реляционных БД обычно выделяется следующая последовательность нормальных форм:

- первая нормальная форма (1NF);
- вторая нормальная форма (2NF);
- третья нормальная форма (3NF);
- нормальная форма Бойса—Кодда (BCNF);
- четвертая нормальная форма (4NF);
- пятая нормальная форма, пли форма проекции-соединения (5NF или PJNF).

Основные свойства нормальных форм:

- каждая следующая нормальная форма в некотором смысле улучшает свойства предыдущей;
- при переходе к следующей нормальной форме свойства предыдущих нормальных форм сохраняются.

В основе классического процесса проектирования лежит последовательность переходов от предыдущей нормальной формы к последующей. Однако в процессе декомпозиции мы сталкиваемся с проблемой ***обратимости,*** то есть возможности восстановления исходной схемы. Таким образом, декомпозиция должна

сохранять эквивалентность схем БД при замене одной схемы на другую.

Приведем ряд основных определений.

Эквивалентные схемы.

Схемы БД называются эквивалентными, если содержание исходной БД может быть получено путем естественного соединения отношений, входящих в результирующую схему, и при этом не появляется новых кортежей в исходной БД.

Функциональная зависимость.

В отношении R набор атрибутов Y функционально зависит от набора атрибутов X или набор атрибутов X функционально определяет набор атрибутов Y (обозначается R.X→R.Y) тогда и только тогда, когда каждому значению X соответствует в точности одно значение Y.

Полная функциональная зависимость.

Функциональная зависимость называется полной, если набор атрибутов Y функционально зависит от X и не зависит функционально от любого подмножества X.

Транзитивная зависимость.

Функциональная зависимость R.X→ R.Y называется транзитивной, если существует набор атрибутов Z такой, что: Z не является подмножеством X; Z не включает в себя Y; существует функциональная зависимость R.X → R.Z; существует функциональная зависимость R.Z → R.Y и не существует функциональной зависимости R.Z → R.X.

Замечание. Транзитивная зависимость появляется, когда неключевой атрибут имеет свои собственные свойства.

Возможный ключ.

Возможный ключ – это набор атрибутов, однозначно определяющий кортеж отношения, и при этом при удалении любого атрибута из этого набора его свойство однозначной идентификации кортежа теряется. Среди всех возможных ключей отношения обычно выбирают одни, который считается главным и который называют первичным ключом отношения.

Неключевой атрибут.

Неключевой атрибут – атрибут, не входящий в состав ни одного возможного ключа.

Взаимно-независимые атрибуты.

Взаимно-независимые атрибуты – атрибуты, которые не зависят функционально один от другого.

Детерминант отношения.

Если в отношении существует несколько функциональных зависимостей, то каждый атрибут или набор атрибутов, от которого зависит другой атрибут, называется детерминантом отношения.

Для функциональных зависимостей как фундаментальной основы проекта БД были проведены исследования, позволяющие избежать избыточного их представления. Ряд зависимостей может быть выведен из других путем применения правил, названных аксиомами Армстронга, по имени исследователя, впервые сформулировавшего их.

Это три основных аксиомы:

Рефлективность: если X является подмножеством Y, то Y→X.

Пополнение: если X→Y , то XC→YC.

Транзитивность: если X→Y и Y→Z, то X→Z.

Доказано, что данные правила являются полными и исчерпывающими, то есть, применяя их, из заданного множества функциональных зависимостей можно вывести все возможные функциональные зависимости.

Существует несколько других правил вывода, которые следуют из аксиом Армстронга.

Правило самоопределения. $X \rightarrow X$.

Правило объединения. Если $X \rightarrow Y$ и $X \rightarrow Z$, то $X \rightarrow Y \cup Z$.

Правило псевдотранзитивности. Если $X \rightarrow Y$ и $W \cup Y \rightarrow Z$, то $X \cup W \rightarrow Z$.

Правило композиции. Если $X \rightarrow Y$ и $Z \rightarrow W$, то $X \cup W \rightarrow Y \cup W$.

Правило декомпозиции. Если $X \rightarrow Y$ и Z входит в Y, то $X \rightarrow Z$.

Множество всех возможных функциональных зависимостей, выводимое из заданного набора исходных функциональных зависимостей, называется его замыканием.

При выполнении эквивалентных преобразований сохраняется множество исходных фундаментальных функциональных зависимостей между атрибутами отношений.

Функциональные зависимости определяют не текущее состояние БД, а все возможные ее состояния, то есть они отражают связи между атрибутами, присущие реальному объекту, который моделируется с помощью БД. Поэтому определить функциональные зависимости по текущему состоянию БД можно только в том случае, если экземпляр БД содержит абсолютно полную информацию (то есть никаких добавлений и модификации БД не предполагается). В реальной жизни это требование невыполнимо, поэтому набор функциональных зависимостей задает разработчик, системный аналитик, исходя из глубокого системного анализа ПО.

Декомпозиция схемы отношения.

Декомпозицией схемы отношения R (A_1, A_2, …A_n) называется замена ее совокупностью подмножеств R, таких, что их естественное соединение дает R. При этом допускается, чтобы подмножества были пересекающимися.

Алгоритм декомпозиции основан на следующей теореме.

Теорема о декомпозиции. Пусть R(A, B, C) – отношение, A, B, C – атрибуты.

Если R удовлетворяет зависимости A→B, то R равно соединению его проекций A, B и A, C

$$R(A, B, C) = \pi_{[A,B]}(R) \bowtie \pi_{[A,C]}(R).$$

При нормализации необходимо выбирать такие декомпозиции, которые обладают свойством соединения без потерь.

Вторым важнейшим желательным свойством декомпозиции является свойство сохранения функциональных зависимостей.

Первая нормальная форма (1NF).

Отношение находится в первой нормальной форме (1NF), если все атрибуты отношения принимают простые значения (атомарные или неделимые), не являющиеся множеством или кортежем из элементарных составляющих.

Рассмотрим следующий пример.

Таблица представляет сущность ЭКЗАМЕНАЦИОННАЯ ВЕДОМОСТЬ

Код студента	Фамилия	**Код экзамена**	Предмет и дата	Оценка
1	Сергеев	1	Математика 5.06.10	4
2	Иванов	1	Математика 5.06.10	5
1	Сергеев	2	Физика 9.06.10	5
2	Иванов	2	Физика 9.06.10	5

Данная таблица не находится в первой нормальной форме (ненормализованная), так как на пересечении строки и четвертого столбца располагается не одно значение, а несколько.

Перейдем к первой нормальной форме. Для этого разнесем значения предмета и даты в разные столбцы и запишем для каждой строчки информацию по экзамену.

Код студента	Фамилия	**Код экзамена**	Предмет	Дата	Оценка
1	Сергеев	1	Математика	5.06.10	4
2	Иванов	1	Математика	5.06.10	5
1	Сергеев	2	Физика	9.06.10	5
2	Иванов	2	Физика	9.06.10	5

Теперь на пересечении любой сроки и любого столбца находится одно значение и, следовательно, данная таблица находится в первой нормальной форме.

Вторая нормальная форма (2NF).

Отношение находится во второй нормальной форме (2NF), если оно находится в 1NF, и каждый неключевой атрибут функционально полно зависит от всего первичного ключа (не зависит от части ключа).

Для перевода отношения во 2NF необходимо, используя операцию проекции, разложить его на несколько отношений следующим образом:

1) построить проекцию без атрибутов, находящихся в неполной функциональной зависимости от первичного ключа;

2) построить проекции на части составного ключа и атрибуты, зависящие от этих частей.

В отношении ЭКЗАМЕНАЦИОННАЯ ВЕДОМОСТЬ первичный ключ – Код студента, Код экзамена.

Функциональные зависимости:

Полная функциональная зависимость

Код студента, Код экзамена→ Оценка

Неполная функциональная зависимость

Код студента→ Фамилия

Код экзамена→Предмет

Код экзамена→Дата

Перейдем ко второй нормальной форме. Для этого построим три проекции:

- для полной функциональной зависимости:

Код студента	**Код экзамена**	Оценка
1	1	4
2	1	5
1	2	5
2	2	5

- для неполной функциональной зависимости:

Код студента	Фамилия
1	Сергеев
2	Иванов

Код экзамена	Предмет	Дата
1	Математика	5.06.10
2	Физика	9.06.10

Замечание. При отсутствии атрибутов, зависящих от всего первичного ключа, проекция на первичный ключ обязательно должна быть в списке отношений, полученных при приведении ко 2NF.

Третья нормальная форма (3NF).

Отношение находится в 3NF, если оно находится в 2NF, и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Третья нормальная форма исключает избыточность и аномалии включения и удаления.

Рассмотрим следующий пример.

R (**№ студ. билета**, ФИО, Группа, Специальность, Факультет, Выпускающая кафедра).

Первичным ключом отношения является № студ. билета, однако рассмотрим остальные функциональные зависимости. Группа, в которой учится студент, однозначно определяет факультет, на котором он учится, а также специальность. Кроме того, выпускающая кафедра также однозначно определяет факультет, на котором обучаются студенты, выпускаемые по данной кафедре. Но если мы предположим, что одну специальность или группу могут выпускать несколько кафедр, то ни специальность, ни группа не определяют выпускающую кафедру. В этом случае у нас есть следующие функциональные зависимости:

№ студ. билета → ФИО
№ студ. билета. → Группа
№ студ. билета → Факультет
№ студ. билета → Специальность
№ студ. билета → Выпускающая кафедра
Группа → Факультет
Группа → Специальность
Выпускающая кафедра → Факультет

И эти зависимости образуют транзитивные группы. Для того чтобы избежать этого, мы можем предложить следующий набор отношений:

(**№ студ. билета**, ФИО, Группа, Выпускающая кафедра).
(**Группа**, Специальность, Факультет)
(**Выпускающая кафедра**, Факультет)
Первичные ключи отношений выделены.

Полученный набор отношений находится в третьей нормальной форме.

К сожалению, 3НФ не предотвращает все возможные аномалии.

Нормальная форма Бойса – Кодда (BCNF).

Отношение находится в нормальной форме Бойса – Кодда (BCNF) тогда и только тогда, когда каждый его детерминант является потенциальным ключом.

BCNF является более строгой версией 3NF. Иными словами, любое отношение, находящееся в BCNF, находится в 3NF. Обратное неверно.

Оказывается, что любая схема отношения может быть приведена в нормальную форму Бойса – Кодда таким образом, чтобы

декомпозиция обладала свойством соединения без потерь. Однако схема отношения может быть неприводимой в BCNF с сохранением зависимостей. В этом случае приходится довольствоваться третьей нормальной формой.

Рассмотрим отношение, моделирующее сдачу студентом текущих экзаменов. Предположим, что студент может сдавать экзамен по одной дисциплине несколько раз, если он получил неудовлетворительную оценку. Допустим, что во избежание возможных полных однофамильцев мы можем однозначно идентифицировать студента номером его студенческого билета, но, с другой стороны, у нас ведется электронный учет текущей успеваемости студентов, поэтому каждому студенту присваивается в период его обучения в вузе уникальный номер-идентификатор. Отношение, которое моделирует сдачу текущей сессии, имеет следующую структуру:

R1 (№ студ. билета, Идентификатор_студента, Дисциплина, Дата, Оценка)

Возможны два потенциальных ключа:

№ студ. билета, Дисциплина, Дата и

Идентификатор_студента, Дисциплина, Дата

Какие функциональные зависимости имеются?

№ студ. билета, Дисциплина, Дата → Оценка;

Идентификатор_студента, Дисциплина, Дата → Оценка;

№ студ. билета → Идентификатор_студента;

Идентификатор_студента → № студ. билета.

Рассмотрим две последние функциональные зависимости. Мы предварительно описали, что каждому студенту ставится в соответствие один номер студенческого билета и один Идентификатор_студента, поэтому по значению № студ. билета можно однозначно определить Идентификатор_студента (это третья зависимость) и обратно (это четвертая зависимость). Оценим это отношение. Оно находится в третьей нормальной форме, потому что неполных функциональных зависимостей непервичных атрибутов от атрибутов возможного ключа здесь не присутствует, и нет транзитивных зависимостей.

Но вот под нормальную форму Бойса–Кодда наше отношение не подходит, потому что есть два детерминанта № студ. би-

лета и Идентификатор_студента, которые не являются возможными ключами отношения. Для приведения отношения к нормальной форме Бойса–Кодда надо разделить отношение, например, на два со следующими схемами;

(**Идентификатор_студента, Дисциплина, Дата,** Оценка)

(**Идентификатор_студента**, № студ. билета)

или наоборот:

(№ студ. билета, Дисциплина, Дата, Оценка)

(№ студ. билета, Идентификатор_студента)

Эти схемы равнозначны с точки зрения теории нормализации, поэтому выбирать проектировщикам следует исходя из некоторых дополнительных рассуждений. Например, если учесть, что студенческие билеты могут теряться, то, как они будут восстанавливаться: если с тем же самым номером, то нет разницы, но если с новым номером, то тогда первая схема предпочтительней.

В большинстве случаев достижение третьей нормальной формы или даже формы Бойса – Кодда считается достаточным для реальных проектов БД, однако в теории нормализации существуют нормальные формы высших порядков, которые уже связаны не с функциональными зависимостями между атрибутами отношений, а отражают более тонкие вопросы семантики предметной области и связаны с другими видами зависимостей.

Четвертая нормальная форма (4NF).

Пусть дано отношение, которое моделирует предстоящую сдачу экзаменов на сессии.

R3 (**Группа, № студ. билета, Дисциплина**)

Каждой группе однозначно соответствует перечень дисциплин по учебному плану, и номер группы определяет список студентов, которые в этой группе учатся. Данное отношение находится в 3NF, однако и тут возможны аномалии вставки и удаления. Если мы будем работать с исходным отношением, то не сможем хранить информацию о новой группе и ее учебном плане – перечне дисциплин, которые должна пройти группа до тех пор, пока в нее не будут зачислены студенты. При изменении перечня дисциплин по учебному плану, например при добавлении новой дисциплины, внести эти изменения в отношение для

всех студентов, занимающихся в данной группе, весьма затруднительно. С другой стороны, если мы добавляем студента в уже существующую группу, то мы должны добавить множество кортежей, соответствующих перечню дисциплин для данной группы. Эти аномалии модификации отношения связаны с наличием многозначных зависимостей.

В отношении R(A, B,C) существует ***многозначная зависимость*** R.A→→R.B в том и только в том случае, если множество значений B, соответствующее паре значений A и C, зависит только от A и не зависит от C.

В теории реляционных БД доказывается, что в общем случае в отношении R (A, B, C) существует многозначная зависимость R.A →→ R.B в том и только в том случае, когда существует многозначная зависимость R.A →→ R.C.

В отношении R3 существуют следующие две многозначные зависимости:

Группа →→ Дисциплина и Группа →→ № студ. билета

Дальнейшая нормализация отношений, подобных R3, основывается на теореме Фейджина.

Теорема Фейджина

Отношение R (A, B,C) можно спроецировать без потерь в отношения R1 (A, B) и R2 (A, C) в том и только в том случае, когда существует многозначная зависимость R.A →→ R.B (R.A →→ R.C)

Под проецированием без потерь понимается такой способ декомпозиции отношения, при котором исходное отношение полностью и без избыточности восстанавливается путем естественного соединения полученных проекций.

Отношение R находится в четвертой нормальной форме (4NF) в том и только и том случае, если в случае существования многозначной зависимости R.A →→ R.B все остальные атрибуты R функционально зависят от A.

Для отношения R3 получим две проекции в соответствии с многозначными зависимостями.

R4 (Группа, Дисциплина) и R5 (Группа, № студ. билета)

Эти два отношения находятся в 4NF.

Пятая нормальная форма (5NF)

Последней нормальной формой является пятая нормальная форма 5NF, которая связана с анализом нового вида зависимостей, зависимостей «проекции соединения» (project-join зависимости, обозначаемые как PJ-зависимости). Этот вид зависимостей является в некотором роде обобщением многозначных зависимостей.

Отношение R (X, Y, …, Z) удовлетворяет зависимости соединения (X*, Y*, …, Z*) в том и только в том случае, когда R восстанавливается без потерь путем естественного соединения своих проекций X*, Y*, …, Z* .

Наличие PJ-зависимости делает его в некотором роде избыточным и затрудняет операции модификации.

Отношение R находится в пятой нормальной форме (нормальной форме проекции-соединения PJNF) в том и только том случае, когда любая зависимость соединения в R следует из существования возможного ключа в R.

Рассмотрим отношение R6:

R6 (**Преподаватель, Кафедра, Дисциплина**)

Предположим, что каждый преподаватель может работать на нескольких кафедрах и на каждой кафедре может вести несколько дисциплин. В этом случае ключом отношения является полный набор из трех атрибутов. В отношении отсутствуют многозначные зависимости, и поэтому отношение находится в 4NF.

Введем следующие обозначения наборов атрибутов:

ПК (Преподаватель, Кафедра)

ПД (Преподаватель, Дисциплина)

КД (Кафедра, Дисциплина)

Допустим, что отношение R6 удовлетворяет зависимости проекции соединения (ПК, ПД, КД). Тогда отношение R6 не находится в PJNF, потому что его единственным ключом является полный набор атрибутов, а наличие зависимости PJ связано с наборами атрибутов, которые составляют возможные ключи отношения R6. Для того чтобы привести это отношение к PJNF, его надо представить в виде трех отношений:

ПК (Преподаватель, Кафедра),

ПД (Преподаватель, Дисциплина) и КД (Кафедра, Дисциплина)

Пятая нормальная форма редко используется на практике. В большей степени она является теоретическим исследованием. Очень тяжело определить само наличие PJ-зависимостей, потому что утверждение о наличии такой зависимости делается для всех возможных состояний БД, а не только для текущего экземпляра отношения R6. Однако знание о возможном наличии подобных зависимостей, даже теоретическое, нам все же необходимо.

2. Примеры предметных областей для лабораторных работ

1. Автоматизация библиотеки.

Описание предметной области.

Задача – информационная поддержка деятельности научной библиотеки в вузе.

Информации о книгах. Каждая книга имеет шифр, список авторов, название, издательство, год издания, цена, число экземпляров, число свободных экземпляров.

БД должна обеспечивать:

- ведение автоматизированного учёта выдачи/приёма литературы;
- ведение очередей на литературу (по заказам);
- учёт рейтинга изданий (количество читателей и дата последней выдачи);
- поиск литературы по требуемым разделу, теме, автору, ключевому слову (с заданием интересующего периода);
- составление списков должников по годам.

Реализовать следующие функции системы

Функции библиотекаря.

1. Ввод и Обновление сведений о книгах.
2. Поиск книги по шифру, автору или названию и выдача ее.
3. Соответственно сдача, начисление штрафа за порчу и сдачу позднее указанного срока.
4. Удаление в конце года книг с «плохим» состоянием и соответственно уменьшение количества экземпляров.
5. Печать формуляров выбранной учебной группы, т. е. списка взятых книг.
6. Печать списка должников для каждой группы.
7. Ввод и обновление сведений о читателях по группам.

Функции читателя

1. Поиск информации о нужной книге по шифру, автору, названию, теме и по нескольким характеристикам.
2. Печать списка книг по нужной теме.
3. Получение своего формуляра.

2. Автоматизация поликлиники – выдача талонов.

Описание предметной области.

Задача – информационная поддержка деятельности регистратуры в поликлинике.

Информация.

Врачи. Для каждого врача указываются инициалы, домашний адрес, телефон, специальность, кабинет, график приема (день недели, время). Для терапевтов – номер участка.

Участки – Номер участка, Улица – дома, [Улица – дома]

Пример 1 ул. имени Иванова 1, 13, 15, 17
ул. Пушкина 24, 26
2
ул. Ленина 1, 2, 3, 4, 5, 6

Расписание врачей на текущую неделю формируется программно (вставляются все врачи на все дни недели). Общий интервал приема для всех врачей – 3 часа.

Талоны формируются автоматически на всю неделю по дням. Время приема из «расписания» делится на интервалы по 20 минут для каждого врача. Отсутствие фамилии больного означает, что талон не выдан. При выдаче талона вписывается фамилия больного.

Реализовать следующие функции системы

Функции регистратора

1. Ввод и редактирование данных о врачах и графике приема с учетом возможной болезни, отпуска врача.
2. Формирование таблиц Расписания и Талоны автоматически на всю рабочую неделю (запрос делается раз в неделю).
3. Выписать талон к указанному врачу на указанную дату на ближайшее свободное время и распечатать его.
4. Список больных, записавшихся на прием к каждому врачу.
5. Печать расписания врачей на неделю.
6. Печать списка участков с указанием участкового врача.

Функции посетителя

1. Просмотр Расписания.
2. Получение информации о наличии свободных талонов к специалисту (по участку к терапевту).

3. Автоматизация поликлиники – вызовы на дом.

Описание предметной области.

Задача – информационная поддержка деятельности регистратуры в поликлинике.

Информация.

Врачи, участки – смотри предыдушее задание.

Больные – фамилия, адрес, участок, история болезни, номер полиса, дата рождения.

Вызовы врача на дом – участок, улица, дом, квартира, фамилия больного, дата и время обращения больного, симптомы, диагноз, поставленный врачом после посещения.

Технология работы.

1. Больной звонит по телефону в поликлинику. Оператор вносит информацию, кроме диагноза.
2. Больной не обязан помнить свой участок. Участок вычисляется исходя из его адреса.
3. Для каждого участка (терапевта) формируется список людей, которых должен посетить врач.
4. После посещения больного врач звонит (приходит) в поликлинику и проставляет атрибут «диагноз» в вызовах.
5. После проставления диагноза вызов считается обслуженным.

Реализовать следующие функции системы

1. Ввод данных о вызове.
2. Автоматическое вычисление участка больного на основании адреса.
3. Для терапевтов – количество больных людей на его участке.
4. Для терапевтов – сколько вызовов на дом поступило за неделю.
5. Сколько больных на всех участках обратилось к врачу с одним диагнозом.
6. Для каждого врача распечатать список людей, которых нужно сегодня посетить.

4. Автоматизация работы кадрового агентства.

Описание предметной области.

Задача – информационная поддержка деятельности кадрового агентства.

Информация.

Список вакансий, полученных от предприятий – предприятие, должность, пол, образование (не ниже…), возраст (границы допустимого), условия работы, заработная плата.

Список безработных, имеющихся в агентстве, – фамилия, адрес, телефон, должность, на которую претендует, пол, образование, возраст.

Реализовать следующие функции системы

Функции менеджера агентства

1. Ввод и обновление сведений о вакансиях.
2. Ввод и обновление сведений о претендентах.
3. Поиск для претендентов подходящей вакансии.

Поиск может осуществлятся следующим образом

1. По должности. Сопоставляются вакансии и претенденты, и для каждого претендента выдается список вакансий. Учесть должность, возраст, пол, образование.
2. Без учета должности (претендент согласен на любую должность). В этом случае учесть только возраст, пол, образование.

Например, если в вакансии указано, что пол мужской, образование не ниже среднетехнического, возраст 30–50 лет, то мужчина с высшим образованием 32 лет подойдет, а мужчина со среднеспециальным 34 лет – нет.

3. Формирование отчета о работе агентства – сколько трудоустроено претендентов за определенный период работы по должностям.

Пример:

Разнорабочих – 122

Поваров – 78

Доцентов – 234

Всего 2056

4. Печать подобранных вакансий с указанием адреса и телефона отдела кадров предприятия.

5. Автоматизация работы диетической столовой.

Описание предметной области.

Задача – калькуляция столовой.

Информация.

В диетической столовой для каждого блюда необходимо указывать его калорийность. Нужно написать программу расчета калорийности блюда исходя из его состава на основании общей таблицы калорийности продуктов.

Таблица калорийности продуктов: продукт, кол-во кал. на 100 гр.

Блюда – название блюда, компоненты, вес каждого компонента (на 1 порцию), калорийность каждого компонента (по весу), общая калорийность блюда, признак наличия блюда сегодня в столовой

Реализовать следующие функции системы

1. Ввод таблицы калорийности.
2. Ввод и обновление состава каждого блюда.
3. Расчет калорийности для блюд – из таблицы калорийности выбираются все компоненты, и на основании веса каждого компонента рассчитывается калорийность компонентов, потом они складываются и получаем калорийность блюда.
4. Каждый день отмечается, какие именно блюда сегодня готовятся. После этого печатается меню из отмеченных блюд с указанием калорийности каждого блюда.

6. Автоматизация работы книжного магазина.

Описание предметной области.

Задача – информационная поддержка деятельности книжного магазина.

Информация.

Книги. Дата поступления книги (тиража), закупочная цена, шифр, автор, название, издательство, число поступивших экземпляров, число еще не проданных экземпляров, цена продажи. Цена может меняться в зависимости от популярности книги (если книгу плохо покупают, то цена на оставшийся тираж снижается.)

Для каждой проданной книги указать дату поступления, дату продажи, цену, по которой была продана. Книги должны быть

объединены по темам: «Детективы», «Программирование», «Фантастика» и т. д.

Реализовать следующие функции системы

Функции менеджера магазина.

1. Ввод и обновление сведений о книгах и тиражах.
2. Поиск книги по шифру, автору или названию и продажа ее (т. е. внесение ее в базу проданных книг с изменением количествава оставшихся экземпляров).
3. Определение самой продаваемой книги – с наибольшим количеством проданных экземпляров.
4. Определение самой непродаваемой книги.
5. Определение чистой прибыли с каждого тиража и по всем тиражам вместе.
6. Определение убыточных тиражей (общая цена всех проданных книг этого тиража не превысила закупочную цену).
7. Определение объема продаж за каждый месяц в виде диаграммы (месяц – количество проданных книг).

Функции покупателя

1. Поиск информации о нужной книге по автору, названию.
2. Печать списка книг по нужной теме.

7. Автоматизация работы детского сада.

Описание предметной области.

Задача – информационная поддержка деятельности детского сада.

Информация.

Группы – название, возраст детей (3, 4, 5, 6 лет), программа, фамилии воспитателей, максимальное количество детей, реальное количество детей.

Дети – фамилия, адрес, родители (место работы, телефон).

Статистика по каждому ребенку: рост, вес при поступлении в садик, динамика роста и веса каждый месяц до полного выхода из садика в 7 лет. Также на каждого ребенка должна храниться информация о периодах болезни.

Реализовать следующие функции программы

1. Ввод и редактирование информации.
2. Печать списка групп.

3. Вывод информации о количестве свободных мест (указать возраст, программа).

4. Получать следующие отчеты:

- статистику о количестве заболеваний в садике за определенный период (по группам, по всему саду);
- динамику прироста веса по каждому ребенку за год (по группам, по всему саду);
- динамику прироста веса по каждому ребенку за весь период пребывания в садике (в виде графика);
- статистику по каждому году о среднем приросте веса в каждой группе (в виде диаграммы).

5. График наполняемости группы на каждый день.

8. Тестирование.

Описание предметной области.

Задача – организация автоматического тестирования слушателей.

Информация.

Тесты: название теста, для каких групп предусмотрен тест. Список вопросов с вариантами ответов, время теста.

Вопросы: текст вопроса, тип вопроса (единственный ответ, множественный выбор), количество вариантов ответа, сами варианты ответа, номера правильных ответов.

Информация об ученике (тестируемом) – фамилия, группа, количество правильных ответов на тест, оценка.

Критерий выставления оценки: 90% пр. ответов – 5; 70% пр. ответов – 4; 50% пр. ответов – 3; < 50% – 2.

Реализовать следующие функции

Функции преподавателя

1. Ввод самого теста (защищенная паролем функция).
2. Получение экзаменационной ведомости для каждой группы (список фамилий с оценками).

Функции ученика

1. Регистрация ученика (ввод группы и фамилии).
2. Выбор теста из списка доступных тестов для группы (каждый тест проходится один раз).

3. Тестирование учеников – последовательная печать вопроса с вариантами ответа, ученик указывает номер выбираемого ответа и переходит к следующему вопросу. В конце получает оценку. Оценка заносится в ведомость.

4. Контроль времени прохождения теста.

9. Автоматизация работы супермаркета.

Описание предметной области.

Задача – информационная поддержка работы супермаркета.

Информация.

Список товаров магазина – код, название товара, цена, единица измерения (шт., кг), производитель, дата производства, срок годности, количество.

Карточки со скидками – код карточки, размер скидки в процентах.

Корзина – товары, которые набрал покупатель и предъявляет к оплате на кассе. При расчете с покупателем вводится код товара и считается общая сумма покупки. Покупатель может предъявить карточку, тогда вводится ее код. Если покупатель предъявил карточку, то сумма пересчитывается. Если покупатель набрал товаров на сумму более 500 р. и у него еще нет карточки, то ему она выдается со скидкой 2%. (вносится в базу карточек). Если покупатель набрал товаров на сумму более 3000 р. и у него еще нет карточки, то ему она выдается со скидкой 4%. (вносится в базу карточек). При этом 2%-ная карточка, если она была, аннулируется.

Реализовать следующие функции

1. Заполнение списка товаров.
2. Заполнение списка карточек.
3. Расчет с покупателем на кассе – ввод товаров, которые он набрал, подсчет общей суммы, которую он должен заплатить. Товары выбираются из списка, количество вводится.
4. Накопительная система по каждой карточке – сохранять все суммы с датами покупок.
5. Выдача карточек покупателям.
6. Заказ заканчивающихся продуктов.
7. Списание просроченных продуктов.
8. Подсчет выручки за период (день, неделя, месяц, год).

10. Автоматизация телефонного справочника ЯрГУ.

Описание предметной области.

Задача – информационная поддержка работы справочной службы.

Информация.

Подразделения – факультет (кафедра), адрес, телефон, ФИО декана (заведующего кафедрой), ФИО секретаря.

Сотрудники подразделений – ФИО, должность, телефон, комната

Реализовать следующие функции системы

1. Ввод и обновление сведений о подразделениях и сотрудниках.
2. Поиск по телефону, фамилии, комнате.
3. Поиск по подразделению.
4. Печать справочника в удобном и компактном виде:
   а) по одному факультету;
   б) по всем факультетам.

11. Автоматизация пункта проката видеокассет.

Описание предметной области.

Задача – информационная поддержка работы пункта проката видеокассет.

Информация.

Банк кассет: название фильма, режиссер, аннотация, раздел (детектив, триллер, комедия и т. д.), основные актеры, количество свободных экземпляров, цена за сутки, залог.

Банк пользователей: адрес, фамилия, залог (руб.), дата возврата кассет, список кассет, которые на руках.

Реализовать следующие функции системы

1. Ведение банка кассет.
2. Ведение банка пользователей.
3. Поиск кассеты по разным параметрам (название фильма, режиссер, основные актеры) и выдача покупателю.
4. Прием, начисление штрафа за порчу и сдачу позднее указанного срока.
5. Печать тематического списка по разделам (детектив, триллер, комедия и т. д.).
6. Поиск должников – тех, кто вовремя не сдал кассеты.

7. Статистика – поиск наиболее востребованных фильмов и печать их (с сортировкой по количеству спроса).

8. Поиск наиболее активных пользователей проката и печать их (с сортировкой по количеству кассет, которые он брал за все время).

12. Автоматизация работы аптеки.

Описание предметной области.

Задача – информационная поддержка деятельности аптеки.

Информация.

Банк лекарств: название лекарства, показания к применению, аннотация лекарства, раздел (простудные, витамины, антибиотики и т. д.), производитель, основное действующее вещество, количество в данной аптеке, цена за единицу, срок годности.

Книга продаж: дата, лекарство, количество.

Книга заказов: лекарство, количество, к оплате.

Реализовать следующие функции системы

1. Ведение банка лекарств.
2. Печать прайс-листа по разделам лекарств.
3. Ведение книги продаж.
4. Поиск лекарства по параметрам (название, болезнь, производитель, основное действующее вещество).
5. Подбор более дешевого лекарства с тем же основным действующим веществом.
6. Статистика продаж – список наиболее часто спрашиваемых лекарств.
7. Если наиболее часто спрашиваемые лекарства закончились, то автоматическое занесение их в книгу заказов.
8. Списание лекарств с истекшим сроком годности.

13. Автоматизация работы деканата вуза.

Описание предметной области.

Задача – информационная поддержка деятельности деканата вуза.

Информация.

Учебный план: предмет, семестр, специальность, курс, количество часов, вид отчетности (зачет, экзамен).

Преподаватели: инициалы, предмет, кафедра.

Студенты: инициалы, зачетная книжка, группа.

Реализовать следующие функции системы

1. Ведение расписания сессии, хранение результатов сессии.
2. Составление зачётных и экзаменационных ведомостей.
3. Составление расписаний экзаменов по группам, кафедрам, для отдельных преподавателей.
4. Проверка корректности расписания экзаменов (уникальность комбинации «время – дата – аудитория»; между экзаменами в одной группе должно пройти не менее трёх дней).
5. Подсчёт по результатам зачётов и экзаменов итоговых значений (количество оценок «5, 4, 3, 2», количество неявок, средний балл по группе).
6. Получение списка экзаменов на текущую дату.
7. Отчисление, перевод на следующий курс.

14. Автоматизация работы адвокатской конторы.

Описание предметной области.

Задача – информационная поддержка деятельности адвокатской конторы.

Информация.

Банк адвокатов: инициалы, адрес, телефон, стаж работы, специализация, список дел.

Банк клиентов: инициалы, адрес, паспортные данные, список дел.

Дела: номер, дата начала/конца, клиент, адвокат, максимальный срок, полученный срок.

Реализовать следующие функции системы

1. Ведение списка адвокатов.
2. Ведение списка клиентов.
3. Ведение архива законченных дел.
4. Получение списка текущих клиентов для конкретного адвоката.
5. Определение эффективности защиты (максимальный срок минус полученный срок) с учётом оправданий, условных сроков и штрафов.

6. Определение неэффективности защиты (полученный срок минус минимальный срок).

7. Подсчёт суммы гонораров (по отдельных делам) в текущем году.

8. Получение для конкретного адвоката списка текущих клиентов, которых он защищал ранее (из архива, с указанием полученных сроков и статей).

15. Автоматизация работы фирмы.

Описание предметной области.

Задача – информационная поддержка деятельности фирмы, занимающейся продажей и арендой жилых и нежилых помещений.

Информация.

Банк объектов: адрес, характеристики (этаж, количество комнат, материал, ремонт и т. д.), вид сделки (продажа, аренда), цена, владелец.

Реализовать следующие функции системы

1. Осуществлять ведение списков жилых и нежилых помещений, предназначенных для аренды и/или продажи.

2. Поддерживать архив проданных и сданных в аренду помещений.

3. Производить поиск вариантов в соответствии с требованиями клиента.

4. Необходимо предусмотреть получение разнообразной статистики:

- наличие помещений разных типов;
- изменение цен на рынке;
- уровни спроса и предложения;
- средние показатели (среднее время нахождения помещения в БД (по типам помещений));
- среднюю стоимость аренды/продажи помещений и т. п.

16. Автоматизация работы гостиницы.

Описание предметной области.

Задача – информационная поддержка финансовой деятельности гостиницы.

Информация.

Гостиница предоставляет номера клиентам на определенный срок. Каждый номер характеризуется вместимостью, комфортностью (люкс, полулюкс, обычный) и ценой. Клиентами являются различные лица, о которых собирается некоторая информация (инициалы, паспортные данные, пол и некоторые комментарии). Сдача номера клиенту производится при наличии свободных мест в номерах, подходящих клиенту по указанным параметрам. При поселении фиксируется дата поселения и срок. При выезде для каждого места запоминается дата освобождения.

Реализовать следующие функции системы

1. Ведение и редактирование информации о номерах.
2. Ведение списка клиентов.
3. Поиск свободных номеров по введенным критериям (комфортность, дата заезда, срок поселения).
4. Автоматический расчет клиента.
5. Возможность бронирования номеров и снятия брони (в случае отказа).
6. Предоставление скидок определенным категориям клиентов.

17. Автоматизация работы ломбарда.

Описание предметной области.

Задача – информационная поддержка финансовой деятельности ломбарда.

Информация.

Деятельность компании организована следующим образом: к Вам обращаются различные лица с целью получения денежных средств под залог определенных товаров. Про каждого клиента хранится информация: инициалы, адрес, паспортные данные. После оценивания стоимости принесенного в качестве залога товара (вид, материал, вес, процент износа) определяется сумма, которая может быть выдана на руки клиенту, а также комиссионные. Определяется срок возврата. В случае согласия клиента оформляется договор продажи. Если в указанный срок не происходит возврата денег, то товар переходит в собственность ломбарда и выставляется на продажу.

Реализовать следующие функции системы

1. Оценка товара по его характеристикам.
2. Заключение договоров с клиентами.
3. Возврат товара при возврате денег клиентом (закрытие сделки).
4. Ежедневное отслеживание просроченных договоров с переводом товаров в продажу.
5. Уценка товара (сезонная скидка, ситуация на рынке).
6. Необходимо предусмотреть получение разнообразной статистики:

- суммарной стоимости заключенных договоров;
- списков принятых товаров по категориям и датам;
- выручки от продаж.

18. Автоматизация работы нотариальной конторы.

Описание предметной области.

Задача – информационная поддержка работы нотариальной конторы.

Информация.

Деятельность нотариальной конторы организована следующим образом: фирма готова предоставить клиенту определенный комплекс услуг. Для наведения порядка Вам необходимо формализовать эти услуги, составив их список с описанием каждой услуги. При обращении клиента его стандартные данные (инициалы, вид деятельности, адрес, телефон) фиксируются. По каждому факту оказания услуги клиенту составляется документ, в котором указываются услуга, сумма сделки, комиссионные (доход конторы), описание сделки. В рамках одной сделки клиенту может быть указано несколько услуг. Кроме того, постоянным клиентам компания может предоставлять скидки.

Реализовать следующие функции системы

1. Ведение прайс-листа услуг.
2. Заключение сделок с клиентами.
3. Автоматический расчет суммы сделки и комиссионных.
4. Вывод на печать договора о сделке.

19. Автоматизация распределения учебной нагрузки.

Описание предметной области.

Задача – информационная поддержка учебного процесса в вузе.

Информация.

Имеются сведения о преподавателях кафедры, включающих наряду с анкетными данными информацию об их ученой степени и стаже работы. Преподаватели должны обеспечить проведение занятий по некоторым предметам учебного плана (учебный план – предмет, семестр, специальность, курс, количество часов (отдельно лекции и практика), вид отчетности (зачет, экзамен)). Нагрузка каждого преподавателя должна соответствовать нормам, установленным вузом (лекционная нагрузка, практические занятия).

Реализовать следующие функции системы

1. Ведение и редактирование учебного плана.
2. Ведение и редактирование информации о кадровом составе.
3. Автоматическое распределение нагрузки.
4. Распечатка нагрузки конкретного преподавателя (конкретной кафедры).
5. Распечатка учебной нагрузки для специальности (дисциплина, количество часов, вид (лекция /практика), преподаватель).

20. Автоматизация работы туристической фирмы.

Описание предметной области.

Задача – информационная поддержка финансовой деятельности туристической фирмы.

Информация.

Работа с клиентами организована следующим образом: у каждого клиента, пришедшего в фирму, собираются некоторые стандартные данные – инициалы, адрес, телефон, паспортные данные. После этого сотрудники выясняют у клиента, где он хотел бы отдыхать. При этом ему демонстрируются различные варианты, включающие страну проживания, особенности местного климата, имеющиеся отели разного класса, вид транспорта, программа тура. Наряду с этим обсуждается возможная длительность пребывания и стоимость путевки. Стоимость путевки за-

висит от длительности тура и отеля. В случае если удалось договориться и найти для клиента приемлемый вариант, то регистрируется факт продажи путевки (или путевок).

Реализовать следующие функции системы

1. Регистрация клиента.
2. Подбор варианта тура и автоматический расчет стоимости.
3. Заключение договора и печать путевки.
4. Услуга «Горящие путевки» со скидкой.
5. Предоставление различной статистической информации:
   - выручка фирмы за период;
   - наиболее популярные маршруты;
   - маршруты, не пользующиеся спросом.
6. Прием претензий клиентов и ведение списков неблагополучных отелей.

21. Автоматизация учета телефонных переговоров.

Описание предметной области.

Задача – информационная поддержка коммерческой службы телефонной компании.

Информация.

Компания предоставляет абоненту телефонные линии для междугородных переговоров. Абонентами компании являются юридические лица, имеющие телефонную точку, ИНН, расчетный счет в банке. Стоимость переговоров зависит от города, в который выполняется звонок, времени суток (день, ночь), длительности звонка. Каждый звонок автоматически фиксируется в БД. В компании введена глубокая система скидок – стоимость минуты уменьшается в зависимости от длительности разговора. Размер скидки для каждого города разный.

Реализовать следующие функции системы

1. Заключение договоров на обслуживание клиентов.
2. Автоматическая регистрация звонков, расчет стоимости.
3. Распечатка звонков клиента с подсчетом общей суммы за период.
4. Закрытие счета после оплаты.
5. Начисление пени в случае задержки оплаты (возможность блокирования номера).

22. Автоматизация работы фирмы по прокату автомобилей.

Описание предметной области.

Задача – отслеживание финансовых показателей работы пункта проката.

Информация.

В автопарк входит некоторое количество автомобилей различных марок, стоимостей и типов. Каждый автомобиль имеет свою стоимость проката, зависящую от марки, года выпуска и срока проката. В пункт проката обращаются клиенты. Все они проходят обязательную регистрацию для сбора стандартной информации (инициалы, адрес, телефон, паспортные данные). Каждый клиент может обращаться в пункт проката несколько раз. Все обращения фиксируются, при этом по каждой сделке запоминаются дата выдачи и ожидаемая дата возврата. Необходимо ввести систему штрафов за возвращение автомобиля в ненадлежащем виде и систему скидок для постоянных клиентов.

Реализовать следующие функции системы

1. Ведение и пополнение информации в банке автомобилей.
2. Регистрация клиентов и подбор автомобилей.
3. Автоматический расчет стоимости проката с учетом возможной скидки.
4. Возврат автомобиля в салон и расчет штрафа, если автомобиль возвращен в ненадлежащем виде.
5. Расчет выручки пункта проката.
6. Статистика по наиболее популярным маркам.

23. Автоматизация работы информационно-аналитического центра коммерческого банка.

Описание предметной области.

Задача – отслеживание динамики работы кредитного отдела.

Информация.

Основной вид деятельности банка – выдача кредитов юридическим лицам. В зависимости от условий получения кредита, процентной ставки и срока возврата все кредиты делятся на несколько основных видов. Каждый из этих видов имеет свое назва-

ние. Кредит может получить клиент, при регистрации предоставивший следующие сведения: название, вид собственности, адрес, телефон, контактное лицо. Каждый факт выдачи кредита регистрируется банком, при этом фиксируются сумма кредита, идентификатор клиента и дата выдачи. Кредит может гаситься частями, и за задержку возврата кредита начисляются штрафы.

Реализовать следующие функции системы

1. Введение и пополнение информации об услугах банка.
2. Регистрация клиентов.
3. Подбор кредита в соответствии с желанием клиента.
4. Заключение договора на кредитование с распечаткой сроков и сумм выплат.
5. Начисление штрафа за задержку возврата.
6. Получение клиентом информации об остатке долга.
7. Перерасчет долга, если клиент вносил суммы, превышающие указанные в договоре.

24. Автоматизация работы ювелирной мастерской.

Описание предметной области.

Задача – отслеживание финансовой деятельности ювелирной мастерской.

Информация.

Ювелирная мастерская осуществляет изготовление ювелирных изделий для частных лиц на заказ. Мастерская работает с определенными материалами (платина, золото, серебро, различные драгоценные камни и т. д.) При обращении потенциального клиента вы определяете, какое именно изделие ему необходимо (выбор из каталога или индивидуальный заказ). Все изготавливаемые изделия принадлежат некоторому типу (серьги, кольца, браслеты, броши), выполнены из определенного материала (или нескольких материалов), имеют некоторый вес и цену (включающую стоимость материалов и работы). Постоянным клиентам может предоставляться скидка. В случае отказа клиента получить заказ, изделие поступает в продажу.

Реализовать следующие функции системы

1. Ведение и пополнение информации о материалах, имеющихся на данный момент в мастерской.

2. Подбор изделия клиенту и расчет стоимости.

3. Оформление заказа, расчет стоимости с учетом возможной скидки (редактирование информации о материалах, имеющихся на складе.)

4. Предоставление каталога для выбора изделия, редактирование каталога.

5. Реализация изделий, от которых отказались заказчики.

25. Автоматизация работы по сдаче в аренду торговых площадей.

Описание предметной области.

Задача – поддержка работы крупного торгового центра, сдающего в арену коммерсантам свои торговые площади.

Информация.

Работа торгового центра построена следующим образом: в результате планирования центр определил некоторое количество торговых точек в пределах здания, которые могут сдавать в аренду. Для каждой из торговых точек важными данными являются этаж, площадь, наличие кондиционера и стоимость аренды в день. С потенциальных клиентов собираются данные (название, адрес, телефон, реквизиты, контактное лицо). Клиенту показывают имеющиеся свободные площади. При достижении соглашения заключается договор, фиксируя в БД точку, клиента, срок аренды.

Реализовать следующие функции системы

1. Ведение и обновление информации о торговых точках.

2. Регистрация клиента.

3. Подбор варианта аренды, расчет оплаты с возможностью предоставления некоторых видов скидок.

4. Заключение договора.

5. Поддержка информации о ежемесячных платежах, поступающих от арендаторов.

6. Расторжение договора в случае неуплаты, с уведомлением арендатора.

Список литературы

1. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт; пер. с англ. – 6-е изд. – К.; М.; СПб.: Вильямс, 2000. – 848 с.

2. Карпова, Т. С. Базы данных: модели, разработка, реализация / Т. С. Карпова. – СПб.: Питер, 2002. – 304 с.

3. Конноли, Т. Базы данных: проектирование, реализация и сопровождение: Теория и практика / Т. Конноли, К. Бэгг, А. Страчан; пер. с англ. – 2-е изд. – М.: Вильямс, 2000. – 1120 с.

4. Крёнке, Д. Теория и практика построения баз данных / Д. Крёнке. – 8-е изд. – СПб.: Питер, 2003. – 800 с.

Оглавление

О. В. Власова

Системы управления базами данных