Министерство образования и науки Российской Федерации

Орский гуманитарно-технологический институт (филиал)
Федерального государственного бюджетного образовательного учреждения
высшего профессионального образования
«Оренбургский государственный университет»

ПРАКТИКУМ ПО БАЗАМ ДАННЫХ

Методические рекомендации

ИЗДАТЕЛЬСТВО ОРСКОГО ГУМАНИТАРНО ТЕХНОЛОГИЧЕСКОГО ИНСТИТУТА

Орск 2012

УДК 681.3
ББК 32.973-018.2
Б 17

Б 17 Практикум по базам данных : методические рекомендации / сост. М. А. Кузниченко. – Орск : Издательство Орского гуманитарно-технологического института (филиала) ОГУ, 2012. – 47 с.

Составитель

Кузниченко М. А., *старший преподаватель*
кафедры программного обеспечения
Орского гуманитарно-технологического института (филиала) ОГУ

Методические рекомендации предназначены для студентов, обучающихся по программам высшего профессионального образования по специальности 230105 – Программное обеспечение вычислительной техники и автоматизированных систем, по направлениям бакалавриата 230100 – Информатика и вычислительная техника и 231000 – Программная инженерия при изучении дисциплины профессионального цикла «Базы данных».

Содержание

Пояснительная записка 4
1. Реляционная алгебра 5
1.1. Методические указания 5
1.2. Задания 9
2. Функциональные зависимости 13
2.1. Методические указания 13
2.2. Задания 14
3. Теория нормализации 17
3.1. Методические указания 17
3.2. Задания 24
4. SQL-запросы на выборку данных 27
4.1. Методические указания 27
4.2. Задания 35
5. SQL-запросы на модификацию данных 37
5.1. Методические указания 37
5.2. Задания 40
6. SQL-запросы на создание таблиц 42
6.1. Методические указания 42
6.2. Задания 46
Библиографический список 47

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Предметом данных методических рекомендаций являются базы данных, без основательного знакомства с которыми в наше время невозможно быть не только квалифицированным программистом, но даже и грамотным пользователем компьютеров.

Как показывает опыт, без знания основ баз данных трудно на серьезном уровне работать с конкретными системами обработки данных, как бы хорошо они не были документированы. Поэтому важно сформировать у студентов профессиональные навыки и компетенции по эффективному управлению данными. Это в первую очередь касается дисциплины «Базы данных» профессионального цикла дисциплин.

В данных методических рекомендациях не рассматривается какая-либо конкретная система управления базами данных (СУБД), излагаемый материал в равной степени относится к любой современной системе. Знание концептуальных основ проектирования баз данных и организации запросов позволит выпускнику свободно ориентироваться в большом многообразии современных СУБД, с которыми ему придется столкнуться в своей профессиональной деятельности.

Весь материал в рекомендациях разбит на четыре главы, в каждой из которых сначала приводятся краткие теоретические сведения по данной теме и примеры решения задач, а затем – практические задания, которые студенты могут выполнять на практических занятиях совместно с преподавателем или в качестве самостоятельной работы.

В первой части рассматривается один из двух базовых механизмов манипулирования данными: реляционная алгебра. Приводятся описание каждой операции реляционной алгебры и наглядное их применение к наборам данных.

Вторая часть посвящена функциональным зависимостям между атрибутами отношения, что лежит в основе теории нормализации и влияет существенным образом на ликвидацию избыточности хранения данных.

В третьей части излагаются принципы нормализации, на которых основан классический подход к проектированию реляционных баз данных. Рассматриваются нормальные формы, их ограничения, а также алгоритм приведения схемы отношения к третьей нормальной форме.

В четвертой части рассматривается язык реляционных баз данных SQL, организация запросов разных типов: выборка, модификация, создание таблиц.

1. РЕЛЯЦИОННАЯ АЛГЕБРА

1.1. Методические указания

Модель данных – это инструмент для представления данных в базе данных. В реляционной модели данных главной информационной конструкцией является отношение (таблица), которое задается совокупностью атрибутов (имя: тип, ограничение). *Реляционная база данных* представляет собой совокупность взаимосвязанных нормализованных отношений, отражающих некоторую предметную область.

Согласно Дейту К., реляционная модель состоит из трех частей, описывающих разные аспекты реляционного подхода: структурная часть, манипуляционная часть и целостная часть.

В *структурной* части модели фиксируется, что единственной структурой данных, используемой в реляционных (БД), является нормализованное n-арное отношение.

В *манипуляционной* части модели утверждаются два фундаментальных механизма манипулирования реляционными БД – реляционная алгебра и реляционное исчисление. Первый механизм базируется в основном на классической теории множеств, а второй – на классическом логическом аппарате исчисления предикатов первого порядка. Рассмотрим более подробно механизм реляционной алгебры.

В *целостной* части реляционной модели данных фиксируются два базовых требования целостности:

– *целостность сущностей* (наличие первичного ключа);

– *ссылочная целостность* (ограничения внешнего ключа связи).

Реляционная алгебра представляет собой набор таких операций над отношениями, что результат каждой из операций также является отношениям. Это свойство алгебры называется *замкнутостью*.

Опишем расширенный начальный вариант алгебры, который был предложен Коддом. В этом варианте набор основных алгебраических операций состоит из восьми операций, которые делятся на два класса – теоретико-множественные операции и специальные реляционные операции. В состав теоретико-множественных операций входят операции:

– объединения отношений;

– пересечения отношений;

– взятия разности отношений;

– декартово произведение отношений.

Специальные реляционные операции включают:

– ограничение отношения;

– проекцию отношения;

– соединение отношений;

– деление отношений.

Для иллюстрации операций реляционной алгебры приведем два отношения одинаковой структуры, содержащие данные о клиентах банков А и В. В таблицах 1.1 и 1.2 представлено содержание данных отношений. Номера кортежей использованы для демонстрации результатов операций реляционной алгебры.

Таблица 1.1

Отношение А (Фамилия, Город)

Кортеж (номер записи)	**Фамилия**	**Город**
К11	Антонов	Орск
К12	Борисова	Москва
К13	Гришин	Самара
К14	Рашель	Москва

Таблица 1.2

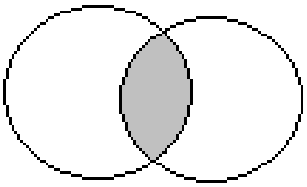
Отношение В (Фамилия, Город)

Кортеж (номер записи)	**Фамилия**	**Город**
К21	Андреев	Самара
К22	Борисова	Москва
К23	Бежин	Орск
К24	Стешина	Орск
К25	Рашель	Москва

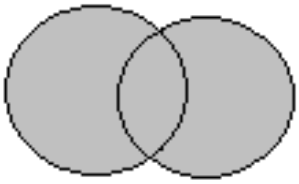
При выполнении операции *объединения* двух отношений производится отношение, включающее все кортежи, входящие хотя бы в одно из отношений-операндов (рис. 1.1 б).

Операция *пересечения* двух отношений производит отношение, включающее все кортежи, входящие в оба отношения-операнда (рис. 1.1 а).

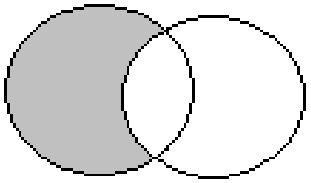
Отношение, являющееся *разностью* двух отношений, включает все кортежи, входящие в отношение – первый операнд, такие, что ни один из них не входит в отношение, являющееся вторым операндом (рис. 1.1 в, г).



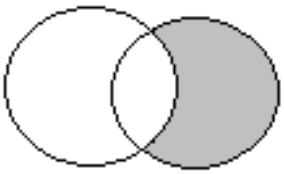
а) Пересечение A ∩ B



б) Объединение A ∪B



в) Разность A – B



г) Разность B – A

Рис. 1.1. Иллюстрация операций реляционной алгебры

Для выполнения операций объединения, пересечения и разности необходимым условием является совместимость по типу отношений-операндов, то есть совпадение схем отношений. Таким образом, результаты операций над отношениями A и B одинаковой структуры будут следующие:

A ∩ B ={K12, K14} – клиенты, у которых есть вклады в обоих банках;

A ∪B = {K11, K12, K13, K14, K21, K23, K24} – клиенты, у которых есть вклады хотя бы в одном из банков;

A – B = {K11, K13} – клиенты только банка A;

B – A = {K21, K23, K24} – клиенты только банка B.

При выполнении *декартова произведения* двух отношений разных структур производится отношение, кортежи которого являются конкатенацией (сцеплением) кортежей первого и второго операндов. Структура нового отношения также является сцеплением структур исходных отношений.

Например, пусть имеются отношения R1(a,b) и R2(c) разной структуры. На рисунке 1.2 показан результат операции декартова произведения R1⊗ R2.

R1

a	**b**
a_1	b_1
a_2	b_2
a_3	b_3

R2

c
c_1
c_2

R1⊗ R2.

a	**b**	**c**
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_2	c_1
a_2	b_2	c_2
a_3	b_3	c_1
a_3	b_3	c_2

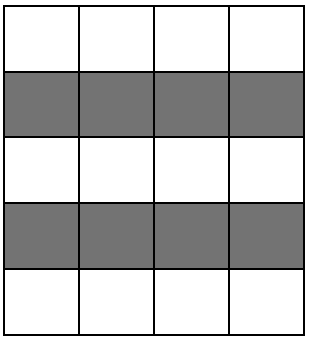
Рис. 1.2. Декартово произведение R1⊗R2

Операции ограничения (выборки) и проекции являются унарными, то есть применяются к одному отношению.

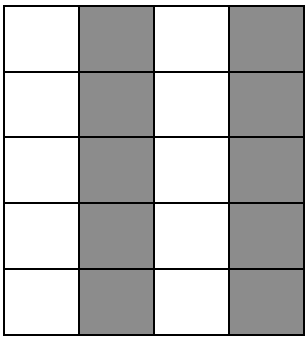
Результатом *ограничения* отношения по некоторому условию является отношение, включающее кортежи отношения-операнда, удовлетворяющее этому условию.

При выполнении *проекции* отношения на заданный набор его атрибутов производится отношение, кортежи которого производятся путем взятия соответствующих значений из кортежей отношения-операнда.

Схематично операции ограничения и проекции изображены на рисунке 1.3, где R – отношение-операнд, r – подмножество атрибутов исходного отношения R, на которые выполняется проекция.



а) ограничение R Where условие



б) проекция R[r]

Рис. 1.3. Операции ограничения и проекции

При *соединении* двух отношений по некоторому условию образуется результирующее отношение, кортежи которого являются конкатенацией кортежей первого и второго отношений и удовлетворяют этому условию.

Соединение выполняется над разными по схеме отношениями, но имеющими хотя бы один общий атрибут, который участвует в формировании условия соединения. Этим атрибутом, как правило, является внешний ключ связи. На рисунке 1.4 показано соединение отношений R1(a,c) и R2(c,d) по условию совпадения внешнего ключа связи – атрибута c: R1.c=R2.c.

R1

a	**c**
a_1	c_1
a_2	c_2
a_3	c_2
a_4	c_3
a_5	c_5

R2

c	**d**
c_1	d_1
c_2	d_2
c_3	d_3
c_4	d_4

R1 * R2 (R1.c=R2.c)

a	**c**	**d**
a_1	c_1	d_1
a_2	c_2	d_2
a_3	c_2	d_2
a_4	c_3	d_3

Рис. 1.4 Соединение отношений

Обратите внимание, что последняя строка отношения R1 не попала в результат соединения, так как для значения внешнего ключа c_5 нет соответствующей записи в отношении R2.

У операции реляционного *деления* два операнда – бинарное и унарное отношения. Результирующее отношение состоит из одноатрибутных кортежей, включающих значения первого атрибута кортежей первого операнда таких, что множество значений второго атрибута (при фиксированном значении первого атрибута) совпадает со множеством значений второго операнда.

Данная операция не имеет широкого применения, поэтому в данном руководстве пропустим ее подробное описание. Желающие докопаться до истины могут обратиться к Дейту «Введение в системы баз данных».

1.2. Задания

Для демонстрации операций реляционной алгебры рассмотрим базу данных «Поставки для проектов» [1], схема которой изображена на рисунке 1.5. База данных состоит из четырех таблиц: три справочные таблицы и одна зависимая. Таблица S (табл. 1.3) содержит сведения о поставщиках (номер, название поставщика, статус, город); таблица P (табл. 1.4) содержит сведения о деталях

(номер, название, цвет, вес, город изготовления), таблица J (табл. 1.5) – данные о проектах (номер, название проекта, город проекта), зависимая таблица SPJ (табл. 1.6) – данные о поставках (номер поставщика, номер детали, номер проекта, количество поставленных деталей, дата поставки).

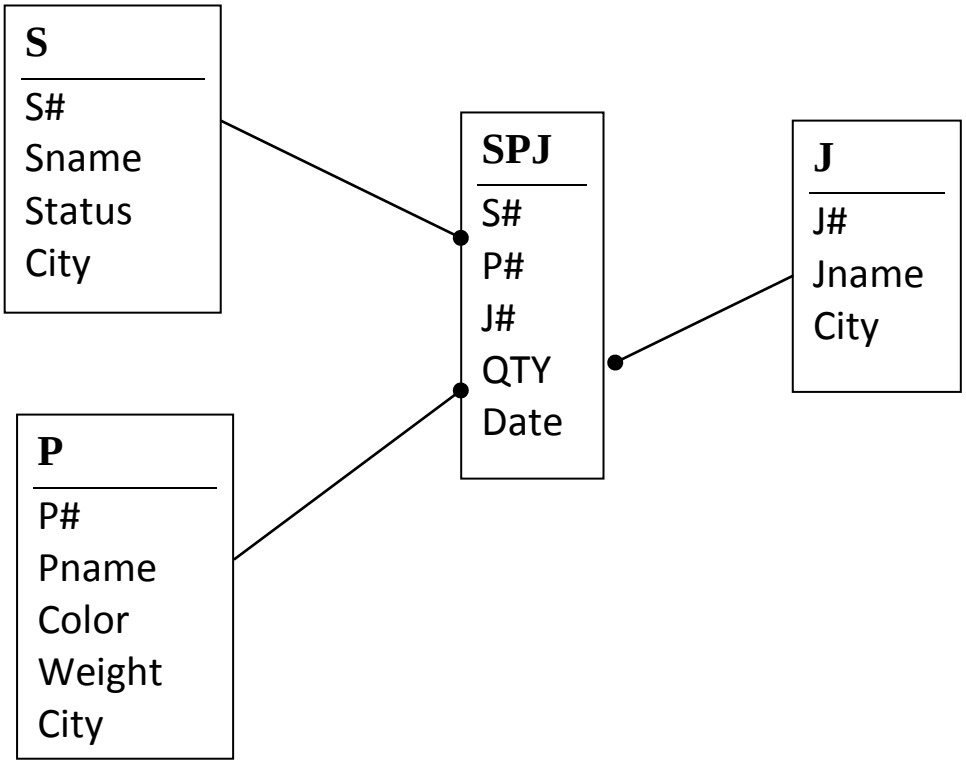


Рис. 1.5. Схема базы данных «Поставки для проектов»

Таблица 1.3

Таблица S (Поставщики)

S#	Sname	Status	City
S1	Смит	20	Лондон
S2	Джонс	10	Париж
S3	Блэк	30	Париж
S4	Кларк	20	Лондон
S5	Адамс	30	Афины

Таблица 1.4

Таблица P (Детали)

P#	Pname	Color	Weight	City
P1	Гайка	Красный	12	Лондон
P2	Болт	Зеленый	17	Париж
P3	Винт	Синий	17	Рим
P4	Винт	Красный	14	Лондон
P5	Узел	Синий	12	Париж
P6	Зубец	Красный	19	Лондон

Таблица 1.5

Таблица J (Проекты)

J#	Jname	City
J1	Сигма	Париж
J2	ДИСП	Рим
J3	ОКР	Афины
J4	Консоль	Афины
J5	РЭД	Лондон
J6	ЕДС	Москва
J7	Тейп	Лондон
J8	Альфа	Москва

Таблица 1.6

Таблица SPJ (Поставки)

S#	P#	J#	Date	QTY (кол-во)
1	*2*	*3*	*4*	*5*
S1	P1	J1	12/02/11	200
S1	P1	J4	11/04/11	700
S2	P3	J1	01/12/11	400
S2	P3	J2	07/02/11	200
S2	P3	J3	22/05/12	200
S2	P3	J4	12/02/11	500
S2	P3	J5	01/04/11	600
S2	P3	J6	12/02/11	400
S2	P3	J7	13/02/12	800
S2	P5	J2	05/11/12	100
S3	P3	J1	02/07/11	200
S3	P4	J2	09/03/12	500
S4	P6	J3	28/02/12	300
S4	P6	J7	11/09/11	300
S5	P2	J2	12/06/11	200
S5	P2	J4	12/06/11	100
S5	P5	J5	12/06/11	500
S5	P5	J7	12/06/11	100
S5	P6	J2	06/12/11	200
S5	P1	J4	12/02/12	100
S5	P3	J4	01/06/12	200
S5	P4	J4	12/02/12	800
S5	P5	J4	17/02/12	400
S5	P6	J4	19/04/12	500

Практические задания

1. Получить список поставщиков из Парижа и Лондона.

2. Определить все поставки в апреле 2010 года.

3. Получить названия поставщиков, которые сделали поставки 12.06.11.

4. Получить названия проектов, для которых были поставки после 01.06.11.

5. Получить пары поставщик-проект, которые находятся в одном городе.

6. Добавить к предыдущему выражению реляционной алгебры название поставщика и название проекта.

7. Получить названия проектов, для которых поставлялся «Болт зеленый».

8. Получить сведения о поставщиках, которые поставили хотя бы 1 красную деталь.

9. Получить названия поставщиков из Афин, которые поставляли деталь «P5», указав количество и дату поставки.

10. Получить названия и цвета деталей, которые были поставлены для проекта «Альфа», указать их количество и дату поставки.

11. Получить информацию о поставках для всех Афинских проектов в декабре 2011 года.

12. Получить названия проектов, для которых за 1 поставку было поставлено более 500 деталей.

13. Получить номера проектов, для которых не было ни одной поставки (добавить названия проектов).

14. Получить названия поставщиков, которые не поставляли детали в апреле 2011 года.

15. Получить всевозможные пары номер_детали-номер_проекта, для которых эта деталь не поставляется для этого проекта.

16. Получить названия поставщиков, которые в 2010 году делали поставки для проекта «Тейп».

17. Вывести данные о поставках либо для Лондонских проектов, либо поставщиками из Лондона.

2. ФУНКЦИОНАЛЬНЫЕ ЗАВИСИМОСТИ

2.1. Методические указания

Схема БД – совокупность схем отношений, адекватно моделирующих абстрактные объекты предметной области и семантические связи между этими объектами. Основой анализа корректности схемы является анализ функциональных зависимостей между атрибутами отношений БД. Некоторые функциональные зависимости являются нежелательными из-за побочных эффектов и аномалий, возникающих при модификации данных [3].

На этапе инфологического моделирования выполняется построение модели «сущность-связь» (ER-диаграммы). Для продолжения процесса проектирования необходимо проверить полученную схему БД на отсутствие избыточных функциональных зависимостей и при необходимости нормализовать схему БД.

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

Функциональная зависимость между атрибутами.

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

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

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

Частичная функциональная зависимость. Функциональная зависимость X → Y называется частичной, если X является подмножеством составного первичного ключа.

Транзитивная функциональная зависимость. Пусть X, Y, Z – три атрибута некоторого отношения. При этом X → Y и Y → Z, но обратное соответствие отсутствует, то есть Z -/-> Y и Y -/-> X. Тогда Z транзитивно зависит от X.

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

2.2. Задания

1. Определите множество функциональных зависимостей и первичный ключ в следующем отношении W1.

W1	**Продукция**	**Цена продукции**	**Комплектующее изделие**
	Эдв-12	40	Вк-15
	Эдв-12	40	Р-20
	Эдв-30	20	Вк-15
	Эдв-30	20	Р-20
	ЗВИ	120	Р-20

2. Докажите или опровергните наличие функциональных зависимостей в следующем отношении.

T2	**KROL**	**FOUN**	**DIN**	**DOT**	**IMH**
	54	32	LO	KU	V3
	34	78	OL	VD	Z2
	45	27	KE	KE	R9
	94	52	LO	VP	P3
	54	65	KE	WA	E7
	45	32	OL	AD	V3
	34	78	KE	WA	Z2
	94	85	OL	KI	A8

1. DIN-> KROL.
2. IMH -> KROL.
3. FOUN -> IMH.
4. KROL -> FOUN.
5. DOT, KROL -> DIN.

3. Докажите или опровергните наличие функциональных зависимостей в следующем отношении.

T3	POR	TRIG	CHEK	KOV	SKB
	RA	2T	F4	GN	BX
	XO	1A	E7	N1	WX
	CW	9X	G8	VD	AX
	KP	5E	U6	QW	BX
	WA	9X	E7	EP	RX
	RA	5E	U6	KA	XY
	AV	1A	G8	KR	AX
	WE	9X	F4	EP	BX

1. POR->KOV.

2. CHEK -» POR.

3. TRIG -> CHEK.

4. POR, CHEK -> KOV.

5. SKB, KOV -> TRIG.

4. Докажите или опровергните наличие функциональных зависимостей в следующем отношении.

T4	INES	KATR	LID	ELIT	SMID
	LU	01	35	UW	70
	GN	02	54	WA	60
	WA	03	35	YW	70
	KU	01	35	GO	40
	LU	02	60	AR	60
	GN	03	35	CE	20
	WA	01	35	SS	30
	AR	02	54	DP	10

1. ELIT -> SMID.

2. LID -> KATR.

3. SMID -> INES.

4. INES, ELIT -> LID.

5. KATR, LID -> SMID.

5. Докажите или опровергните наличие функциональных зависимостей в следующем отношении.

T5	KOS	KOGR	WKI	DUBL	PRIN
	АК	61	WA	570	WS
	AT	44	PR	751	WA
	АК	20	AI	856	RI
	AU	73	WA	536	TX
	AT	15	PR	751	AX
	АК	94	FY	456	PI
	AU	30	CM	846	CI
	AT	52	VD	751	WS

1. KOS -> KOGR.

2. KOGR -> WKI.

3. DUBL, PRIN -> WKI.

4. WKI -> PRIN.

5. KOS, WKI -> DUBL.

6. Укажите функциональные зависимости в следующих отношениях. Найдите вероятные ключи.

T0 (Таб, ФИО, отдел, руководитель отдела);

T1 (ФИО, отдел, должность);

T2 (отдел, проект, число исполнителей); (*Прим.: над одним проектом могут работать разные отделы*)**;**

T3 (электродвигатель, мощность, покупатель, купленное количество);

T4 (преподаватель, кафедра, факультет);

T5 (автор, статья, журнал, номер журнала);

T6 (преподаватель, кафедра, дисциплина);

T7 (автомобиль, грузоподъемность, расход топлива);

T8 (отрасль, предприятие, бригада);

T9 (изделие, деталь, вес изделия).

7. Укажите следствия из множества функциональных зависимостей {A –> B, B –> C}.

8. Укажите следствия из множества функциональных зависимостей F={X-> X1, X->X2, X -> X3, X1-> X4}.

3. ТЕОРИЯ НОРМАЛИЗАЦИИ

3.1. Методические указания

Целью теории нормализации является ликвидация избыточности хранения данных.

Построение схемы БД может быть выполнено путем *декомпозиции* (разбиения), когда исходное множество отношений, входящих в схему БД, заменяется другим множеством отношений (их число при этом возрастает), являющихся проекциями исходных отношений.

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

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

Различают первую нормальную форму (1НФ), вторую нормальную форму (2НФ), третью нормальную форму (3НФ), нормальную форму Бойса-Кодда (НФБК), четвертую (4НФ) и пятую нормальные формы (5НФ).

На практике считается достаточным приведение схемы БД к 3НФ. Рассмотрим определения нормальных форм и примеры.

Первая нормальная форма.

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

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

Под атомарностью понимается степень структурирования и детализации информации в БД. Глубина структурирования определяется практической необходимостью при манипулировании данными. Примером является глубина

структурирования адреса. Можно хранить в одном поле весь адрес (город, улица, дом, квартира). Данный атрибут будет атомарным, если нет необходимости манипулировать отдельными городами или улицами, в противном случае этот атрибут не является атомарным и необходимо его дальнейшее разбиение на отдельные атрибуты (город), (улица, дом, квартира).

Пример ненормализованного и нормализованного (в 1НФ) отношений приведен на рисунке 3.1.

R1 - Ненормализованное отношение

КодПоставщика	КодПродукта	Продукт
P_1	1	Сахар
	2	Соль
	13	Мука
P_2	26	Рис
	58	Гречка
	130	Крупа манная
	162	Пшено
P_3	474	Молоко
	891	Кефир

R2 - Нормализованное отношение

КодПоставщика	КодПродукта	Продукт
P_1	1	Сахар
P_1	2	Соль
P_1	13	Мука
P_2	26	Рис
P_2	58	Гречка
P_2	130	Крупа манная
P_2	162	Пшено
P_3	474	Молоко
P_3	891	Кефир

Рис. 3.1. Пример нормализации отношения

Вторая нормальная форма (2НФ).

Отношение (таблица) находится во 2НФ, если оно находится в 1НФ, и каждый неключевой атрибут функционально полно зависит от всего ключа, то есть отношение не содержит частичных зависимостей.

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

– создать новое отношение, атрибутами которого будут;

• часть составного ключа (первичный ключ нового отношения);

• атрибут, зависящий от нового ключа;

– из исходного отношения исключить атрибут, включенный в новое отношение.

То есть, если имеется отношение R (kl, k2, al, a2), находящееся в 1НФ, где kl, k2 – составной первичный ключ, а al и a2 – неключевые атрибуты отношения R, и имеются функциональные зависимости:

kl, k2 → al (атрибут al функционально полно зависит от первичного ключа kl, k2);

kl →a2 (атрибут a2 зависит от части первичного ключа kl, то есть имеется частичная функциональная зависимость).

Для приведения отношения R к 2НФ это отношение декомпозируется на два отношения: Rl (kl, a2) и R2 (k1, k2, al).

Отношения R1 и R2 будут иметь связь – один ко многим по атрибуту kl.

Пример. Дано отношение:

Поставки (КодПоставщика, КодПродукта, ЕдИзмерения, Кол-во)

Поставщик может поставлять различные продукты, один и тот же продукт может поставляться разными поставщиками. Тогда первичным ключом отношения будут атрибуты КодПоставщика и КодПродукта. Значит, существует функциональная зависимость:

КодПоставшика, КодПродукта → ЕдИзмерения, Кол-во.

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

КодПродукта → ЕдИзмерения.

После исключения неполной функциональной зависимости получим отношения:

Поставка (КодПоставщика, КодПродукта, Кол-во) и **Продукт** (КодПродукта, ЕдИзмерения).

При неполной функциональной зависимости возникают **аномалии**:

• включения (пока поставщиком не будет поставлен продукт, нельзя указать единицу измерения);

• удаления (исключение поставщика может привести к потере единицы измерения продукта);

• обновления (при изменении единицы измерения продукта приходится менять данные везде, где встречается данный продукт). Данные виды аномалий возникают при любой избыточной функциональной зависимости.

Третья нормальная форма (3НФ).

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

То есть, если имеется отношение R(kl, al, a2), находящееся в 2НФ, где kl – первичный ключ, а al и a2 – неключевые атрибуты отношения R, и имеются функциональные зависимости:

k1 → al

a1 → a2,

тогда атрибут a2 транзитивно зависит от k1.

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

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

НомГр → Специальность (и наоборот, Специальность -/-> НомГр)

Специальность → Факультет (Факультет -/-> Специальность)

После исключения транзитивной функциональной зависимости получим отношения: **Группа** (НомГр, Специальность) и **Специальность** (Специальность, Факультет).

Нормальная форма Бойса-Кодда.

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

То есть, если имеется отношение R(al, a2, a3, a4), находящееся в ЗНФ, где al, a2 – возможный ключ, a2, a3 – возможный ключ, а a4 – неключевой атрибут отношения R, и имеются функциональные зависимости:

al → a3

a3 → al

al, a2 → a4

a2, a3 → a4

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

Rl (a1, a3) и R2 (al, a2, a4)

или Rl (a3, al) и R2 (a2, a3, a4).

Пример. Дано отношение **Экзамен** (№ зачетки, № паспорта, Дисциплина, Дата, Оценка). Возможными ключами будут атрибуты: (№ зачетки, Дисциплина, Дата) и (№ паспорта, Дисциплина, Дата). Имеются следующие функциональные зависимости:

№ зачетки, Дисциплина. Дата → Оценка

№ паспорта, Дисциплина. Дата → Оценка

№ зачетки → № паспорта

№ паспорта → № зачетки

После приведения отношения к НФБК могут быть получены отношения:

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

или

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

Четвертая нормальная форма (4НФ).

Отношение находится в 4НФ, если оно находится в НФБК, и в нем отсутствуют многозначные зависимости, не являющиеся функциональными зависимостями.

или

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

То есть, если имеется отношение R(al, a2, a3), находящееся в НФБК и имеются функциональные зависимости:

- зависимость множества значений атрибута a2 от множества значений атрибута al (al ->> a2);
- зависимость множества значений атрибута a3 от множества значений ключевого атрибута al (al ->> a3).

Для приведения отношения R к 4НФ это отношение декомпозируется на два отношения: Rl (al, a2) и R2 (al, a3).

Пример. Дано отношение **Книги** (ISBN, Название, Автор, Область знаний). Книга имеет уникальный идентификатор ISBN, книга может быть написана коллективом авторов, книга может относиться к нескольким областям знаний (табл. 3.1).

Таблица 3.1

Данные о книгах

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

Существуют следующие функциональные зависимости:

ISBN → Название

ISBN ->> Автор

ISBN ->> Область знаний

После приведения отношения к 4НФ будут получены отношения:

Книга (ISBN, Название)

АвторыКниг (ISBN, Автор)

ОбластиЗнанийКниг (ISBN, Область знаний)

Алгоритм нормализации к 3НФ

Исходной информацией является описание предметной области проектируемой базы данных.

Задача заключается в том, чтобы получить схему базы данных, каждое отношение (таблица) которой находится в 3НФ.

Процесс декомпозиции требует, чтобы она обладала свойством соединения без потерь. Для этого можно воспользоваться теоремой Хеза.

Теорема Хеза.

Пусть R(A,B,C) является отношением, и A, B, C – атрибуты или множества атрибутов этого отношения. Если имеется функциональная зависимость A->B, то проекции R_1(A,B) и R_2(A,C) образуют декомпозицию без потерь.

Для реализации метода необходимо выполнить следующие шаги.

Шаг 1. На основании описаний предметной области создать исходное отношение R, в которое включить все необходимые атрибуты. Данное отношение должно находиться в 1НФ.

Шаг 2. Получить исходное множество функциональных зависимостей для реквизитов рассматриваемой базы данных.

Если исходные функциональные зависимости не удается определить путем анализа содержательных ролей реквизитов, приходится использовать перечисление и отбраковку допустимых вариантов функциональных зависимостей.

Шаг 3. Определить первичный ключ отношения.

Шаг 4. Проверить, не нарушает ли исходное отношение 2НФ, то есть наличие частичных функциональных зависимостей.

Если нарушается 2НФ, то на основании нежелательных зависимостей выполнить декомпозицию отношения R на две или более подсхемы по теореме Хеза.

$R= \{R_1, R_2, \dots , R_k\}$

Шаг 5. Для каждой подсхемы R_i, полученной на предыдущем этапе, следует проверить, не нарушает ли она 3НФ, то есть на наличие транзитивных функциональных зависимостей. Если какая-то подсхема нарушает 3НФ, то для неё выполнить декомпозицию по теореме Хеза.

3.2. Задания

1. Примените алгоритм получения отношений в 3НФ, если 3НФ не соблюдается. Постройте реализацию полученных отношений средствами СУБД. Проверьте, выполняется ли свойство соединения без потерь.

Атрибуты	Функциональные зависимости
ФИО вкладчика (фио) Номер сберкнижки (номер) Дата Приход Расход Остаток	номер -> ФИО номер, дата -> приход, расход, остаток

2. Примените алгоритм получения отношений в 3НФ, если 3НФ не соблюдается. Выполняйте декомпозицию схем отношений по теореме Хеза.

Атрибуты	Функциональные зависимости
Магазин Книга Цена Издательство Дата Количество проданных книг (кол-во)	книга, магазин -> издательство магазин, книга, дата -> кол-во книга -> цена, издательство

3. Примените алгоритм получения отношений в 3НФ, если 3НФ не соблюдается. Выполняйте декомпозицию схем отношений по теореме Хеза.

Атрибуты	Функциональные зависимости
Табельный номер (таб.№) Фамилия рабочего (фио) Цех Участок Дата Сумма зарплаты (сумма)	участок -> цех таб. № -> цех таб. № -> участок таб. №, дата -> сумма таб. № -> фио таб. № -> фио, участок

4. Примените алгоритм получения отношений в 3НФ, если 3НФ не соблюдается. Выполняйте декомпозицию так, чтобы она обладала свойством соединения без потерь.

Атрибуты	Функциональные зависимости
Отправитель Получатель Адрес получателя (адрес) Изделие Цена Кол-во на месяц (кол-во)	получатель, изделие -> адрес отправитель, получатель, изделие -> кол-во получатель -> адрес отправитель, изделие -> цена

5. Примените алгоритм получения отношений в 3НФ. Декомпозиция должна обладать свойством соединения без потерь.

Атрибуты	Функциональные зависимости
Порт Судно Грузоподъемность дата отплытия порт назначения	Судно, дата отплытия -> порт Судно -> грузоподъемность

6. По описанию предметной области согласно алгоритму создайте схему базы данных «Междугородний автовокзал»:

– на автовокзале имеется перечень маршрутов междугородних автобусов;

– для каждого маршрута известны: номер, пункт отправления, пункт назначения, расстояние в км, время (отправления);

– по каждому маршруту в день производится не более одного рейса;

– рейсы выполняют водители, о которых известны: ФИО, дата рождения, водительский стаж;

– по каждому совершенному рейсу фиксируется количество проданных билетов.

7. По описанию предметной области согласно алгоритму создайте схему базы данных «Зоопарк»:

– в зоопарке необходимо вести учет животных;

– о каждом животном известны его кличка, год рождения, название вида (жираф, кенгуру, волк, …), номер вольера, в котором оно содержится;

– за животными ухаживают смотрители, о которых известны: ФИО, адрес, оклад;

– один смотритель может ухаживать за несколькими животными;

– каждому виду соответствует его зона обитания (Австралия, Сибирь, Африка, …);

– от вида животного зависит его время кормления (1 раз в сутки, 2 раза в сутки, …).

8. По описанию предметной области согласно алгоритму создайте схему базы данных «Кондитерская»:

– некоторая кондитерская фирма занимается производством и продажей кондитерских изделий, прайс-лист которых известен;

– для каждого изделия известны его название, рецепт, вес, цена, единицы измерения;

– в состав изделия входят различные ингредиенты (масло, сахар, молоко, какао, …) в заданном количестве;

– необходимо учесть ежедневную продажу кондитерских изделий, указывая количество или вес для каждого наименования, чтобы оценить объем продаж и популярность отдельных изделий.

9. По описанию предметной области согласно алгоритму создайте схему базы данных «Библиотека вуза»:

– в библиотеке ведется учет книг и читателей;

– о каждой книге известны: ISBN, название, автор (или авторы), раздел науки, год издания, количество страниц;

– о каждом читателе хранятся № читательского билета, ФИО, телефон, год рождения, пол, статус (студент, сотрудник), место учебы или работы (группа, кафедра, подразделение);

– когда читатель берет книгу на дом, библиотекарь записывает данную книгу в карточку абонента, фиксирует дату выдачи и срок (количество дней);

– по факту возврата книги библиотекарь фиксирует фактическую дату сдачи книги. Если при этом срок возврата просрочен, то может быть читателю начислен штраф, сумма которого определяется и фиксируется библиотекарем согласно правилам аренды книг.

4. SQL-ЗАПРОСЫ НА ВЫБОРКУ ДАННЫХ

4.1. Методические указания

SQL (англ. Structured Query Language – «язык структурированных запросов») – универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных.

SQL основывается на теории множеств (реляционной алгебре) и исчислении предикатов первого порядка (реляционном исчислении).

SQL является основным способом работы пользователя с базой данных и позволяет выполнять следующий набор операций:

– создание в базе данных новой таблицы;

– добавление в таблицу новых записей;

– изменение записей;

– удаление записей;

– выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);

– изменение структур таблиц.

Современные СУБД, а, также, информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

Каждое предложение SQL – это либо запрос данных из базы, либо обращение к базе данных, которое приводит к изменению данных в базе. В соответствии с тем, какие изменения происходят в базе данных, различают следующие типы запросов:

– запросы на создание или изменение в базе данных новых или существующих объектов (при этом в запросе описывается тип и структура создаваемого или изменяемого объекта);

– запросы на получение данных;

– запросы на добавление новых данных (записей);

– запросы на удаление данных;

– обращения к СУБД.

Основным объектом хранения реляционной базы данных является таблица, поэтому все SQL-запросы — это операции над таблицами. В соответствии с этим запросы делятся на:

– запросы, оперирующие самими таблицами (создание и изменение таблиц);

– запросы, оперирующие с отдельными записями (или строками таблиц) или наборами записей.

Каждая таблица описывается в виде перечисления своих полей (столбцов таблицы) с указанием:

– типа хранимых в каждом поле значений;

– связей между таблицами (задание первичных и вторичных ключей);

– информации, необходимой для построения индексов.

Запросы первого типа, в свою очередь, делятся на запросы, предназначенные для создания в базе данных новых таблиц, и на запросы, предназначенные для изменения уже существующих таблиц. Запросы второго типа оперируют со строками, и их можно разделить на запросы следующего вида:

– вставка новой строки;

– изменение значений полей строки или набора строк;

– удаление строки или набора строк.

Самый главный вид запроса – это запрос, возвращающий (пользователю) некоторый набор строк, с которым можно осуществить одну из трёх операций:

– просмотреть полученный набор;

– изменить все записи набора;

– удалить все записи набора.

Таким образом, использование SQL сводится, по сути, к формированию всевозможных выборок строк и совершению операций над всеми записями, входящими в набор.

Организация запросов на выборку данных

Для определения синтаксиса запросов будем использовать нотацию Бэкуса-Наура, которая использует набор спецсимволов, имеющих определенный смысл и не являющихся частью синтаксиса терма. Так, например, квадратные

скобки [] означают необязательность инструкции в записи запроса, вертикальная черта | – «или», треугольные скобки <> означают пользовательский идентификатор.

Синтаксис запроса на выборку:

```
SELECT [ALL | DISTINCT] * |<список полей>|<выражения>
FROM <список таблиц>
[WHERE <предикат отбора> [ AND <условие соединения таблиц>]]
[GROUP BY <список полей> ]
[HAVING <предикат> ]
[ORDER BY <список полей>]
```

Ключевые инструкции SQL-запроса на выборку должны располагаться именно в такой последовательности, но некоторые из них могут отсутствовать.

Инструкция SELECT определяет внешний вид результирующего набора данных, содержит перечень полей (атрибутов), выражений, допустимых стандартом SQL или *, которая означает, что все поля исходной таблицы попадают в результат запроса.

Параметр ALL (установлен по умолчанию) помещает в результат запроса все отобранные записи, включая повторяющиеся значения кортежей, а параметр DISTINCT сформирует набор данных только из уникальных значений кортежей.

Инструкция FROM содержит список таблиц, которых касается запрос. Длинным названиям таблиц можно давать краткие псевдонимы, которые действуют только в пределах данного запроса: <имя таблицы> <псевдоним>. Обратите внимание, что инструкции SELECT и FROM являются обязательными, а остальные могут отсутствовать.

Инструкция WHERE содержит собственно критерий отбора, которому должны удовлетворять записи результирующего набора данных. Критерий может состоять из допустимых предикатов и логических операций: NOT (не), AND (и), OR (или). Предикаты будут рассмотрены ниже.

Если запрос касается двух и более таблиц, то имя поля должно сопровождаться именем таблицы, к которой оно относится в виде:

<таблица>.<поле>

Для длинных имен таблиц могут быть введены краткие псевдонимы через пробел рядом с именем таблицы. Это делается для сокращения записи запроса. Кроме того, в инструкции WHERE должно появиться условие соединения этих таблиц. Как правило, такое соединение выполняется по внешнему ключу связи.

Для демонстрации примеров составления запросов будем использовать схему данных, содержащую информацию об отделах и сотрудниках, которые работают в этих отделах. Схема приведена на рисунке 4.1. Первичным ключом таблицы СОТРУДНИК является его табельный номер (Таб), который однозначно определяет каждого сотрудника организации. Внешним ключом связи между этими таблицами является номер отдела (Ном_отд), который в таблице ОТДЕЛ является первичным ключом.

Например,

```
SELECT T.*, C.*
FROM ОТДЕЛ T, СОТРУДНИК C
WHERE T.Ном_отд = C. Ном_отд
```

Данный запрос позволяет получить полную информацию (все поля) из таблиц ОТДЕЛ и СОТРУДНИК, у которых внешним ключом связи является номер отдела (Ном_отд).

Рис. 4.1. Схема данных СОТРУДНИКИ – ОТДЕЛЫ

Рассмотрим допустимые стандартом SQL предикаты.

Предикаты

1) Предикаты сравнения:

<поле> = | <> | <| <= | > | >= <выражение>

Символами «<>» обозначается операция «неравенства». Арифметические выражения левой и правой частей предиката сравнения строятся по общим правилам построения арифметических выражений и могут включать в общем случае имена столбцов таблиц из раздела FROM и константы или выражения. Типы данных арифметических выражений должны соответствовать друг другу по типу.

Пример. Записать предикат, который выберет данные о сотрудниках, оклад которых меньше 5000.

Оклад < 5000.

2) Предикат попадания в диапазон значений.

<поле> [NOT] BETWEEN <нижн_граница>AND <верхн_граница >

Предикат является истинным, если значение поля попадает в указанный диапазон значений, включая границы. Использование отрицания NOT придает противоположный смысл условию.

По определению результат «x BETWEEN y AND z» тот же самый, что результат логического выражения «x >= y AND x <= z». Результат «x NOT BETWEEN y AND z» тот же самый, что результат «NOT (x BETWEEN y AND z)».

Пример. Записать предикат, который выберет данные о сотрудниках, фамилии которых начинаются на буквы А, Б, В.

Фамилия BETWEEN «А» AND «Г».

3) Предикат принадлежности множеству.

<поле> [NOT] IN (<значение1>, <значение2>, …)

Значение предиката истинно в том и только в том случае, когда значение левого операнда совпадает хотя бы с одним значением списка правого операнда. По определению значение предиката «x NOT IN S» равно значению предиката «NOT (x IN S)».

Типы левого и правого операндов должны совпадать.

Пример. Записать предикат, который выберет данные о сотрудниках 1, 5, 7 и 10 отделов.

Ном_отд IN (1,5,7,10).

4) Предикат сравнения с образцом.

<поле> [NOT] LIKE <шаблон>

Значение предиката является истинным, если значение поля удовлетворяет указанному шаблону. При этом шаблон может содержать два специальных символа: символ подчеркивания («_») обозначает любой одиночный символ; символ процента («%») обозначает последовательность произвольных символов произвольной длины (может быть, нулевой).

Шаблон представляет собой строку, заключенную в кавычки. Он может применяться как к строковым, так и к числовым полям.

Пример. Записать предикат, который выберет данные о сотрудниках, чьи фамилии заканчиваются на «в».

Фамилия LIKE «%в».

Встроенные функции SQL

В тексте запроса могут использоваться вычисляемые поля в виде выражений. Если выражение встречается в инструкции SELECT, то оно формирует новый столбец, которому может быть присвоено смысловое имя:

<выражение> AS <новое имя столбца>.

Это имя может быть использовано в тексте данного запроса.

Язык SQL содержит большую библиотеку функций различных категорий: математических, статистических, агрегатных, функций даты и времени, функции для работы со строками и другие.

На практике часто приходится работать с функциями обработки даты (табл. 4.1).

Таблица 4.1

Функции обработки даты

Функция	**Описание**
Date()	функция возвращает текущую системную дату
Day(поле| выражение)	функция возвращает день от даты, заданной в качестве параметра
Month(поле| выражение)	функция возвращает месяц от даты, заданной в качестве параметра
YEAR(поле| выражение)	функция возвращает номер года от даты, заданной в качестве параметра

Зачастую в расчетах требуется выполнять те или иные действия в зависимости от результата проверки некоторого условия. Для этого используется следующая функция:

IIF (<логическое выражение>, <выражение 1>, <выражение 2>).

Если логическое выражение истинно, то вычисляется выражение 1, иначе – выражение 2.

Например, следующее выражение начисляет студенту стипендию, если его средний балл за сессию равен или превышает 4 балла:

IIF(Сред_балл>=4; 1000; 0) AS Стипендия

Группировка

Язык SQL позволяет группировать данные в запросе по одному или нескольким полям одновременно. При этом записи с одинаковым значением поля группировки объединяются в одну запись в результирующем запросе. К объединенным записям можно применять функции расчета, которые называются агрегатными (или групповыми функциями). Список наиболее часто используемых агрегатных функций приведен в таблице 4.2.

Таблица 4.2

Агрегатные функции

Функция	**Описание**	**Ограничение**
1	***2***	***3***
Count(* | поле)	вычисляет количество непустых записей в каждой группе	применяется как к записи в целом (*), так и к отдельному полю
Sum(поле| выражение)	вычисляет сумму значений поля или выражения в каждой группе	применяется только к числовым полям
AVG(поле| выражение)	вычисляет среднее арифметическое среди значений поля или выражения в каждой группе	применяется только к числовым полям
MAX(поле| выражение)	вычисляет наибольшее значение поля или выражения в каждой группе	применяется только к числовым, строковым полям или полям типа даты
MIN(поле| выражение)	вычисляет наименьшее значение поля или выражения в каждой группе	применяется только к числовым, строковым полям или полям типа даты

Пример. Необходимо вычислить суммарный оклад и количество сотрудников в каждом отделе. Следующий запрос выполнит это:

```
SELECT Ном_отд, SUM(Оклад) AS Общий_оклад, COUNT(*) AS Кол-во сотруд
FROM СОТРУДНИК
GROUP BY Ном_отд
ORDER BY Ном_отд
```

В этом запросе для каждой группы с одинаковым номером отдела будет вычисляться сумма окладов и определяться количество записей в каждой группе.

Если требовалось бы добавить название отдела, то следовало добавить еще одну таблицу ОТДЕЛ, соединив указанные таблицы по номеру отдела.

Пример. Составить запрос, который определит, сколько сотрудников занимают каждую должность. Ввести список должностей в порядке убывания этой величины.

```
SELECT Должность, COUNT(*) AS Количество_ставок
FROM СОТРУДНИК
GROUP BY Должность
ORDER BY Количество_ставок DESC
```

Если в тексте запроса накладывается условие на результат агрегатной функции, то следует его помещать в инструкцию HAVING. Например, если в последнем запросе надо вывести только те должности, количество ставок по которым более 5, то запрос примет следующий вид:

```
SELECT Должность, COUNT(*) AS Количество_ставок
FROM СОТРУДНИК GROUP BY Должность
HAVING Количество_ставок>5
ORDER BY Количество_ставок DESC
```

4.2. Задания

Практические задания относятся к схеме данных СОТРУДНИКИ – ОТДЕЛЫ (рис. 4.1).

1. Получить сведения об отделах, в которых отсутствует телефон.

2. Вывести в алфавитном порядке список сотрудников 3, 17 и 12 отделов по отделам.

3. Получить должности, оклады, фамилии и имена сотрудников, трудоустроенных в организацию в 2011 году.

4. Вывести информацию сначала о мужчинах, а затем – о женщинах по алфавиту фамилий.

5. Составить запрос, который выведет данные о сотрудниках отдела «Бухгалтерия».

6. Вывести данные о сотрудниках, которые работают под начальством А. Н. Петрова.

7. Получить список сотрудников, в названии должности которых есть слово «лаборант», указать их возраст.

8. Получить список (ФИО) женщин с указанием названия отдела, должности и оклада.

9. Создать ведомость заработной платы в алфавитном порядке, указав ФИО и оклад, начислить уральский коэффициент в размере 15% от оклада, вычесть подоходный налог в размере 13% от всех начислений и определить сумму на руки.

10. Создать ведомость для начисления детского пособия для сотрудников, которые имеют детей. На каждого ребенка начислить по 500 руб.

11. Вывести данные о сотрудниках, которые трудоустроились в организацию в текущем году.

12. Получить список сотрудников, день рождения которых в текущем месяце. Указать возраст. Список вывести в порядке возрастания дня поздравления.

13. Создать ведомость заработной платы в алфавитном порядке фамилий, начислив премию в размере 25% от оклада сотрудникам 5-го отдела. Начислить уральский коэффициент в размере 15% от оклада. Рассчитать общее начисление для каждого сотрудника.

14. Получить список сотрудников, указав рядом с фамилией пол, возраст и сообщение «пенсионер» или «работающий». Для мужчин пенсионный возраст 60 лет, а для женщин 55 лет.

15. Получить список должностей организации, указав нижнюю и верхнюю границы оклада для каждой должности.

16. Вывести данные о количестве женщин и мужчин в каждом отделе, а так же их средний возраст.

17. Вычислить, сколько сотрудников было трудоустроено в организацию за каждый месяц с начала текущего года.

18. Вычислить, сколько сотрудников было трудоустроено в организацию за каждый месяц прошлого года.

19. Вывести список отделов, в которых средний возраст сотрудников превышает 45 лет, в следующем формате <название отдела> <средний возраст>.

20. Определить ежегодное количество уволенных сотрудников, начиная с 2000 года.

21. Определить общий фонд заработной платы всей организации.

5. SQL-ЗАПРОСЫ НА МОДИФИКАЦИЮ ДАННЫХ

5.1. Методические указания

Язык манипуляции данными (DML – Data Manipulation Language) включает операторы, изменяющие состояние данных. Этими операторами являются:

INSERT – добавление записи (записей);

DELETE – удаление записи (записей);

UPDATE – обновление значений атрибутов записи (записей).

Оператор INSERT

Оператор INSERT вставляет новые записи в таблицу. При этом значения столбцов могут представлять собой литеральные константы либо являться результатом выполнения другого запроса на выборку. В первом случае для вставки каждой строки используется инструкция VALUES; во втором случае – запрос на выборку SELECT, при этом может быть вставлено столько строк, сколько возвращается запросом.

Синтаксис запроса на добавление следующий:

```
INSERT INTO <имя таблицы>[(<имя столбца>,...)]
{VALUES (<значение столбца>,…)}
| <выражение запроса SELECT >
```

Как видно из представленного синтаксиса, список столбцов не является обязательным (об этом говорят квадратные скобки в описании синтаксиса). В том случае, если он отсутствует, список вставляемых значений должен быть полный, то есть обеспечивать значения для всех столбцов таблицы. При этом

порядок значений должен соответствовать порядку, заданному при определении таблицы, в которую вставляются строки. Кроме того, эти значения должны относиться к тому же типу данных, что и столбцы, в которые они вносятся.

Если для некоторых полей значения не определены или недоступны для изменения, значения могут заменяться константой NULL (или nil).

Пример. Добавить в таблицу ОТДЕЛ сведения о новом отделе маркетинга с номером телефона 234-135.

INSERT INTO ОТДЕЛ VALUES (nil, «Отдел маркетинга», nil, «234-135»)

В тексте программы литеральные константы могут заменяться на значения переменных или полей формы, объединяясь в запрос.

Пример. В отдельную таблицу УВОЛЕННЫЕ переписать всех сотрудников, которые уволены из организации. Таблица УВОЛЕННЫЕ имеет ту же структуру, что и таблица СОТРУДНИК:

INSERT INTO УВОЛЕННЫЕ SELECT * FROM СОТРУДНИК WHERE Дата_увол IS NOT NULL.

Оператор DELETE

Удаляет одну или несколько строк из таблицы или запроса. Инструкция WHERE является необязательной. Если в инструкции DELETE она отсутствует, то из таблицы или запроса, указанных в предложении FROM, будут удалены все строки. При использовании предложения WHERE к каждой строке таблицы или запроса применяется условие отбора, и, если оно принимает значение True, запись удаляется.

Синтаксис:

```
DELETE [имя-таблицы.* | список-столбцов]
FROM имя-таблицы [AS псевдоним ] | имя-запроса-на-выборку
[WHERE условие-отбора]
```

Если в инструкции DELETE указано имя запроса, последний не должен содержать агрегатных функций SQL, предиката DISTINCT, предложений GROUP BY и HAVING или подчиненного запроса, использующего ту же базовую таблицу, что и DELETE.

Если в предложении FROM задана только одна таблица, список полей необязателен. Будут удаляться строки целиком.

При задании имен столбцов в предложении WHERE вместо полных имен таблиц или запросов можно использовать псевдонимы. Псевдоним обязателен при ссылке на имя столбца, которое встречается более чем в одной из таблиц, перечисленных в предложении FROM.

Если в условии отбора используется подчиненный запрос, в нем не должно быть ссылок на базовые и конечные таблицы или запросы.

Пример. Чтобы удалить все строки из таблицы СОТРУДНИК, введите следующую инструкцию:

DELETE FROM СОТРУДНИК

Пример. Следующий запрос удаляет из таблицы СОТРУДНИК всех сотрудников отдела кадров:

DELETE C.*

FROM СОТРУДНИК С, ОТДЕЛ Т

WHERE Т.Название= «Отдел кадров» AND С.Ном_отд =Т. Ном_отд

Обратите внимание, что запрос касается обеих таблиц, но удаление выполняется только из одной таблицы.

Оператор UPDATE

Синтаксис запроса на удаление записей:

```
UPDATE имя-таблицы [[AS] псевдоним] |
<таблица-объединения>}, . . .
SET имя-столбца = выражение | NULL,...
[WHERE условие-отбора]
```

В указанной таблице или запросе обновляются выбранные столбцы (присваивая значение заданного выражения либо NULL) во всех строках, удовлетворяющих условию отбора. Если в инструкции не используется предложение WHERE, то обновляются все строки таблицы или запроса.

В предложении SET нельзя указывать имя столбца более одного раза. Значения, присваиваемые столбцам, должны быть совместимы с характеристиками соответствующих столбцов. Если вы присваиваете значение NULL, свойство Обязательное поле соответствующего столбца не может быть установлено в значение Да.

Пример. Следующий запрос увеличивает зарплату сотрудников 3 отдела на 15%:

```
UPDATE СОТРУДНИК
SET Оклад =Оклад *1.15
WHERE Ном_отд = 3
```

Пример. У сотрудника с табельным номером 123 родился ребенок. Следует увеличить количество детей у данного сотрудника:

```
UPDATE СОТРУДНИК
SET Дети = Дети+1
WHERE Таб_ном=123
```

5.2. Задания

Для составления примеров SQL-запросов на модификацию данных рассмотрим новую схему базы данных Торговля, изображенную на рисунке 5.1.

В базе данных имеется 6 таблиц:

«Клиент» – сведения о компаниях-клиентах;

«Товар» – сведения о товарах, реализуемых торговой фирмой;

«Офис» – сведения об офисах организации;

«Продавец» – сведения о продавцах, которые работают в разных офисах;

«Счет» – сведения о покупках, которые совершает клиент, а оформляет продавец торговой фирмы;

«Детально» – сведения о том, какие позиции товара относятся к данному счету и в каком количестве.

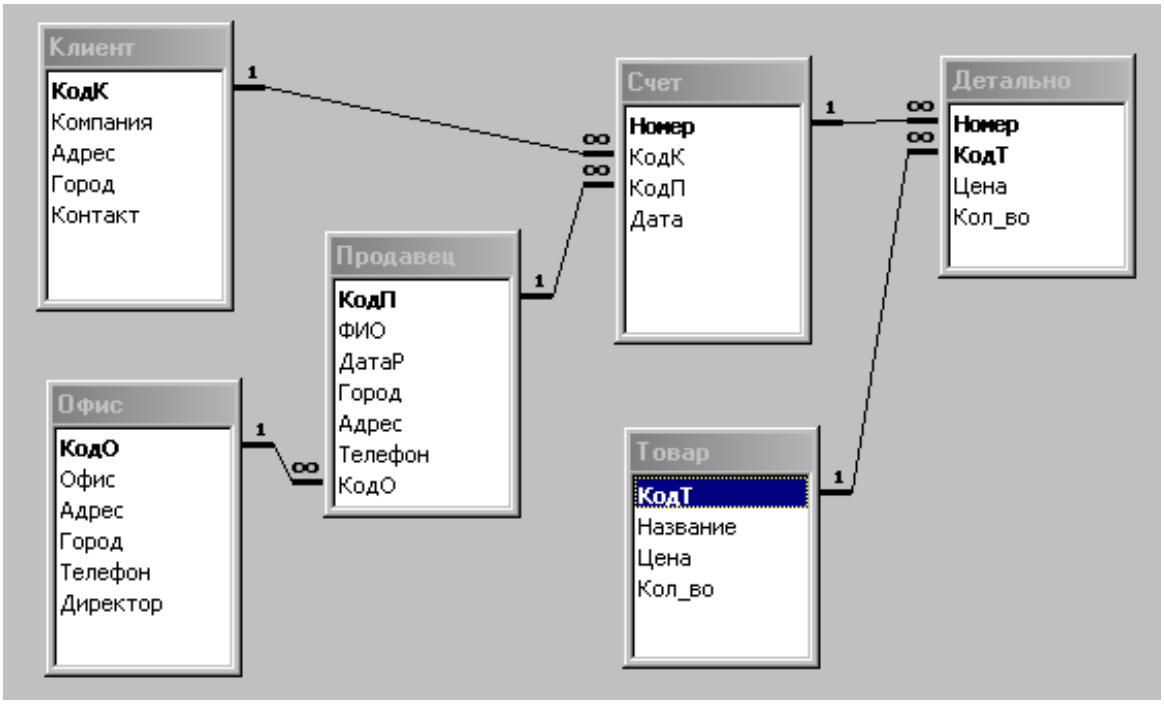


Рис. 5.1. Схема базы данных Торговля

1. Создать запрос, который переписывает в таблицу Товары_0 товары, которые закончились на складе.

2. Создать запрос, который удаляет со склада эти товары.

3. Снизить цены на 20% на все товары, в названии которых есть название фирмы Samsung.

4. Сотрудница Стешина Алла Петровна поменяла фамилию на Антонова.

5. Сотрудник Петров Иван Андреевич поменял номер телефона на новый «33-21-78». Выполнить запрос, который внесет в базу эти изменения.

6. Отредактировать запрос п.5.5 так, чтобы новый номер телефона запрашивался у пользователя с клавиатуры в поле редактирования Edit1.

7. Добавить на склад новый товар «Телевизор LG» по цене 6000 р. в количестве 15 штук.

8. Добавить запись о новом клиенте, если данные о нем пользователь ввел в поля формы: edit1 – название компании, edit2 – адрес, combobox1 – название города. Код клиента КодК является счетчиком.

9. Уменьшить на складе количество товара с кодом 160 на 5 штук.

10. Уменьшить на складе количество товара с кодом 160 на значение, введенное пользователем с клавиатуры в поле Edit1.

11. Удалить все заказы продавца Воробьева Ильи Петровича.

12. В новую таблицу Итоги (Дата, Сумма) поместить итог продаж сегодняшнего дня.

13. Из таблицы Итоги удалить данные за прошлый год.

14. В новую таблицу ИтогиМ (Месяц, Сумма) поместить итог продаж прошлого месяца.

15. В новую таблицу Табель (КодП, Месяц, Сумма) поместить сведения об объемах продаж каждого продавца за текущий месяц.

6. SQL-ЗАПРОСЫ НА СОЗДАНИЕ ТАБЛИЦ

6.1. Методические указания

Язык SQL позволяет создавать объекты базы данных: базы данных, таблицы, индексы и другие. Декларативные ограничения целостности задаются на уровне операторов создания таблиц.

В стандарте SQL оператор создания таблиц имеет следующий синтаксис:

<определение таблицы>::= CREATE TABLE <имя таблицы>

(<описание элемента таблицы> [<описание элемента таблицы>...])

<описание элемента таблицы>::= <определение столбца>|

<определение ограничений таблицы>

определение столбца>::= <имя столбца> <тип данных>

[<значение по умолчанию>] [<дополнительные ограничения столбца>…]

<значение по умолчанию>::= DEFAULT { <litera1> | USER | NULL}

<дополнительные ограничения столбца>::= NOT NULL

[ограничение уникальности столбца>] |[ограничение по ссылкам столбца>|

CHECK (<условия проверки на допустимость>)

<ограничение уникальности столбца>: := UNIQUE

<ограничение по ссылкам столбца>::=FOREIGN KEY <спецификация ссылки>

<спецификация ссылки>::= REFERENCES <имя основной таблицы> (<имя первичного ключа основной таблицы>)

При описании таблицы задается имя таблицы, которое является идентификатором в базовом языке СУБД и должно соответствовать требованиям именования объектов в данном языке.

Оператор CREATE TABLE определяет так называемую базовую таблицу, то есть реальное хранилище данных.

Как видно, кроме обязательной части, в которой задается имя столбца и его тип данных, определение столбца может содержать два необязательных раздела:

– значение столбца по умолчанию;

– раздел дополнительных ограничений целостности столбца.

В разделе значения по умолчанию указывается значение, которое должно быть помещено в строку, заносимую в данную таблицу, если значение данного столбца явно не указано. В соответствии со стандартом языка SQL значение по умолчанию может быть указано в виде литеральной константы с типом, соответствующим типу столбца; путем задания ключевого слова USER, которому при выполнении оператора занесения строки соответствует символьная строка, содержащая имя текущего пользователя (в этом случае столбец должен иметь тип символьных строк); или путем задания ключевого слова NULL, означающего, что значением по умолчанию является неопределенное значение. Если значение столбца по умолчанию не специфицировано и в разделе ограничений целостности столбца указано NOT NULL (то есть наличие неопределенных значений запрещено), то попытка занести в таблицу строку с незаданным значением данного столбца приведет к ошибке.

Задание в разделе ограничений целостности столбца выражения NOT NULL приводит к неявному порождению проверочного ограничения целостности для всей таблицы. Если ограничение NOT NULL не указано и раздел умолчаний отсутствует, то неявно порождается раздел умолчаний DEFAULT NULL.

Если в разделе ограничений целостности указано ограничение по ссылкам данного столбца, то порождается соответствующее определение ограничения по ссылкам для таблицы:

FOREIGN КЕУ(<имя столбца>) <спецификация ссылки>

это означает, что значения данного столбца должны быть взяты из соответствующего столбца родительской таблицы. Родительской таблицей в данном случае называется таблица, которая связана с данной таблицей связью 1:M. При этом каждая строка родительской таблицы может быть связана с несколькими строками определяемой таблицы. Трансляция операторов SQL проводится в режиме интерпретации, поэтому важно, чтобы сначала была бы описана родительская таблица, а потом уже все подчиненные (дочерние) таблицы, связанные с ней. Иначе транслятор определит ссылку на неопределенный объект.

Наконец, если указано проверочное ограничение столбца, то условие поиска этого ограничения должно ссылаться только на данный столбец, и неявно порождается соответствующее проверочное ограничение для всей таблицы. В проверочных ограничениях, накладываемых на столбец, нельзя задавать сравнение со значениями других столбцов данной таблицы.

Пример. Напишем оператора создания таблицы BOOKS из базы данных «Библиотека».

При этом будем предполагать наличие следующих ограничений целостности:

Шифр книги (ISBN) – последовательность символов длиной не более 14, однозначно определяющая книгу, значит, это фактически первичный ключ таблицы BOOKS.

Название книги (TITLE) – последовательность символов, не более 120. Обязательно • должно быть задано.

Автор (AUTOR) – последовательность символов, не более 30, может быть не задан.

Соавтор (COAUTOR) – последовательность символов, не более 30, может быть не задан.

Год издания (YEAR_P) – целое число, не менее 1960 и не более текущего года. По умолчанию ставится текущий год.

Издательство (PUBLISH) – последовательность символов, не более 20, может отсутствовать.

Количество страниц (PAGES) – целое число не менее 5 и не более 1000.

```
CREATE TABLE BOOKS (
ISBN      C(14) PRIMARY KEY,
TITLE    C(120) NOT NULL,
AUTOR      C (30) NULL,
COAUTOR C(30) NULL,
YEAR_P  D DEFAULT Year(Date()) CHECK (YEAR_P >= 1960 AND YEAR_P <= YEAR(Date())),
PUBLICH C(20) NULL,
PAGES   I CHECK(PAGES > = 5 AND PAGES <= 1000) );
```

Пример. Теперь зададим описание таблицы «Читатели», которой соответствует отношение READERS:

Номер читательского билета (READER_ID) – это целое число в пределах 32 000, и он уникально определяет читателя.

Имя (FIRST_NAME), фамилия (LAST_NAME) читателя – это последовательность символов, не более 30.

Адрес (ADRES) – это последовательность символов, не более 50.

Номера телефонов рабочего WORK_PHON и домашнего HOME_PHON – последовательность символов, не более 12.

Дата рождения (BIRTH_DAY) – календарная дата. В библиотеку принимаются читатели не младше 17 лет.

```
CREATE TABLE READERS (
READER_ID  I PRIMARY KEY,
FIRST_NAME char (30)    NOT NULL.
LAST_NAME   char(30)    NOT NULL,
ADRES       char(50)
HOME_PHON   char(12).
WORK_PHON   char(12),
BIRTH_DAY date CHECK(year(Date())-year(BIRTH_DAY) >=17) );
```

Пример. Создадим таблицу «Аренда», которая содержит сведения о том, кто (читатель) и когда (DATE_1) взял некоторую книгу. Необходимо фиксиро-

вать дату возврата данной книги (DATE_2) этим читателем. Эти поля будут типа даты и времени.

В данной таблице необходимо реализовать связь с таблицами Читатели (READERS) и Книги (BOOKS), для чего будем использовать ограничение внешнего ключа FOREIGN KEY. Следует соблюдать соответствие типов для ключей связи.

```
CREATE TABLE ARENDA (
DATE_1 D DEFAULT Date() NOT NULL,
READER_ID I FOREIGN KEY REFERENCES READERS (READER_ID),
ISBN C(14) FOREIGN KEY REFERENCES BOOKS (ISBN),
DATE_2 D NULL);
```

6.2. Задания

1. Создать запрос, который создает таблицу ОТДЕЛ (рис. 4.1). Номер отдела сделать первичным ключом. Использовать для этого автоинкрементный тип auto.

2. Создать запрос, который создает таблицу СОТРУДНИК (рис. 4.1). Табельный номер сотрудника сделать первичным ключом. Использовать для этого автоинкрементный тип auto. Предусмотреть, чтобы при приеме на работу сотруднику было не менее 18 лет. Реализовать связь этой таблицы с таблицей ОТДЕЛ.

3. Создать таблицу СЧЕТ (рис. 5.1), реализовав связи по внешним ключам и всевозможные ограничения. Дату оформления счета сделать текущей, дополнить проверку того, что дата не может быть меньше текущей даты.

4. Сформулировать проверку CHECK для скрипта создания таблицы BOOKS, которая реализует следующее условие: если автор книги не указан, то соавтора не должно быть, а обратное – верно.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Грабер, М. Язык SQL / М. Грабер. – М., 1998.

2. Дейт, К. Введение в системы баз данных / К. Дейт. – 6-е изд. – СПб. : Издательский дом «Вильямс», 1999.

3. Дунаев, В. В. Базы данных. Язык SQL для студентов / В. В. Дунаев. – СПб. : БХВ-Петербург, 2007. – ISBN 978-5-9775-0113-2.

4. Карпова, Т. С. Базы данных : модели, разработка, реализация / Т. С. Карпова. – СПб. : Питер, 2001. – 304 с. – ISBN 5-2720-0278-4.

5. Мишенин, А. И. Теория экономических информационных систем. Практикум : учебное пособие / А. И. Мишенин, С. П. Салмин. – М. : Финансы и статистика, 2005. – 192 с.

6. Петров, В. Информационные системы / В. Петров, Ю. Избачков. – СПб. : Питер, 2005. – ISBN 5-4690-0641-7.

7. Электронный ресурс. – Режим доступа : http://sql-language.ru/.

8. Электронный ресурс. – Режим доступа : http://www.mstu.edu.ru/study/materials/zelenkov/ch_4_6_1.html.

9. Электронный ресурс. – Режим доступа : http://articles.org.ru/docum/sql/sqlnew.php.

10. Электронный ресурс. – Режим доступа : http://cent-sp.com/index.html.

Редактор
И. В. Юматова

Старший корректор
Е. А. Феонова

Ведущий инженер
Г. А. Чумак

Подписано в печать 26.09.2012 г.
Формат 60х84 1/16. Усл. печ. л. 2,8.
Тираж 60 экз. Заказ _____.

**Издательство Орского гуманитарно-технологического института (филиала)
Федерального государственного бюджетного образовательного учреждения
высшего профессионального образования
«Оренбургский государственный университет»**

462403, г. Орск Оренбургской обл., пр. Мира, 15 А