Министерство образования
Республики Беларусь

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Теоретическая механика»

ТЕХНОЛОГИИ БАЗ ДАННЫХ

Лабораторный практикум

Минск
БНТУ
2010

Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Теоретическая механика»

ТЕХНОЛОГИИ БАЗ ДАННЫХ

Лабораторный практикум

Минск
БНТУ
2010

УДК 004.65 (076.5)
ББК 32.97я7
Т 38

Составители:
С.А. Пронкевич, А.В. Чигарев

Рецензенты:
С.М. Босяков, В.Ю. Пятницкий

Т 38 Технология баз данных: лабораторный практикум / сост.: С.А. Пронкевич, А.В. Чигарев. – Минск: БНТУ, 2010. – 84 с.

ISBN 978-985-525-193-5.

Изложены основы языка структурированных запросов SQL, предназначенного для выборки и обработки информации, содержащейся в компьютерной базе данных.

Включает 17 лабораторных работ, каждая из которых содержит краткое изложение основ языка, а также примеры работы с учебной базой данных в программе MS ACCESS. Имеется информация для простого приложения по обработке заказов, использующегося в небольшой торговой компании. Кроме того, в конце каждой лабораторной работы приведены задания для самостоятельного выполнения студентами во время практических занятий.

Издание может быть использовано студентами, а также преподавателями в целях повышения квалификации.

УДК 004.65 (076.5)
ББК 32.97я7

ISBN 978-985-525-193-5

Введение

SQL является инструментом, предназначенным для выборки и обработки информации, содержащейся в компьютерной базе данных. SQL – это сокращенное название *структурированного языка запросов* (Structured Query Language). По историческим причинам аббревиатура SQL читается обычно как «сиквел», но используется и альтернативное произношение «эскюзль». Как следует из названия, SQL является *языком программирования,* который применяется для организации взаимодействия пользователя с базой данных. На самом деле SQL работает только с базами данных одного определенного типа, называемыми *реляционными.*

На рис. 1 изображена схема работы SQL. Согласно этой схеме: в вычислительной системе имеется *база данных,* в которой хранится важная информация. Если вычислительная система относится к сфере бизнеса, то в базе данных могут содержаться сведения о материальных ценностях выпускаемой продукции, объемах продаж и зарплате. В базе данных на персональном компьютере может находиться информация о выписанных чеках, телефонах и адресах или информация, извлеченная из более крупной вычислительной системы. Компьютерная программа, которая управляет базой данных, называется *системой управления базой данные* или СУБД.

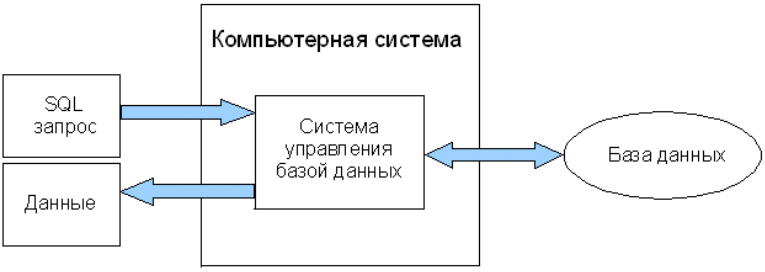


Рис 1. Применение SQL для доступа к базе данных

Если пользователю необходимо получить информацию из базы данных, он запрашивает ее у СУБД с помощью SQL. СУБД обрабатывает запрос, находит требуемые данные и посылает их пользователю. Процесс запрашивания данных и получения результата называется *запросом* к базе данных, отсюда и название – *структурированный язык запросов.*

Однако это название не совсем соответствует действительности. Во-первых, сегодня SQL представляет собой нечто большее, чем просто инструмент создания запросов, хотя именно для этого он и был первоначально разработан. Несмотря на то, что выборка данных по-прежнему остается одной из наиболее важных функций SQL, сейчас этот язык используется для реализации всех функциональных возможностей, которые СУБД предоставляет пользователю. К ним относятся:

– *организация данных.* SQL дает пользователю возможность изменять структуру представления данных, а также устанавливать отношения между элементами базы данных;

– *выборка данных.* SQL дает пользователю или приложению возможность извлекать из базы данных содержащуюся в ней информацию и пользоваться ею;

– *обработка данных.* SQL дает пользователю или приложению возможность изменять базу данных, т.е. добавлять в нее новые данные, а также удалять или обновлять уже имеющиеся;

– *управление доступом.* С помощью SQL можно ограничить возможности пользователя по выборке и изменению данных и защитить их от несанкционированного доступа;

– *совместное использование данных.* SQL координирует совместное использование данных пользователями, работающими параллельно, чтобы они не мешали друг другу;

– *целостность данных.* SQL позволяет обеспечить целостность базы данных, защищая ее от разрушения из-за несогласованных изменений или отказа системы.

Таким образом, SQL является достаточно мощным языком, обеспечивающим эффективное взаимодействие с СУБД.

Лабораторная работа № 1

Учебная база данных

В учебной базе данных содержится информация для простого приложения по обработке заказов, использующегося в небольшой торговой компании.

Учебная база данных состоит из пяти таблиц:

CUSTOMERS (клиенты) содержит по одной строчке для каждого из клиентов компании;

SALESREPS (служащие) содержит по одной строке для каждого из десяти служащих компании;

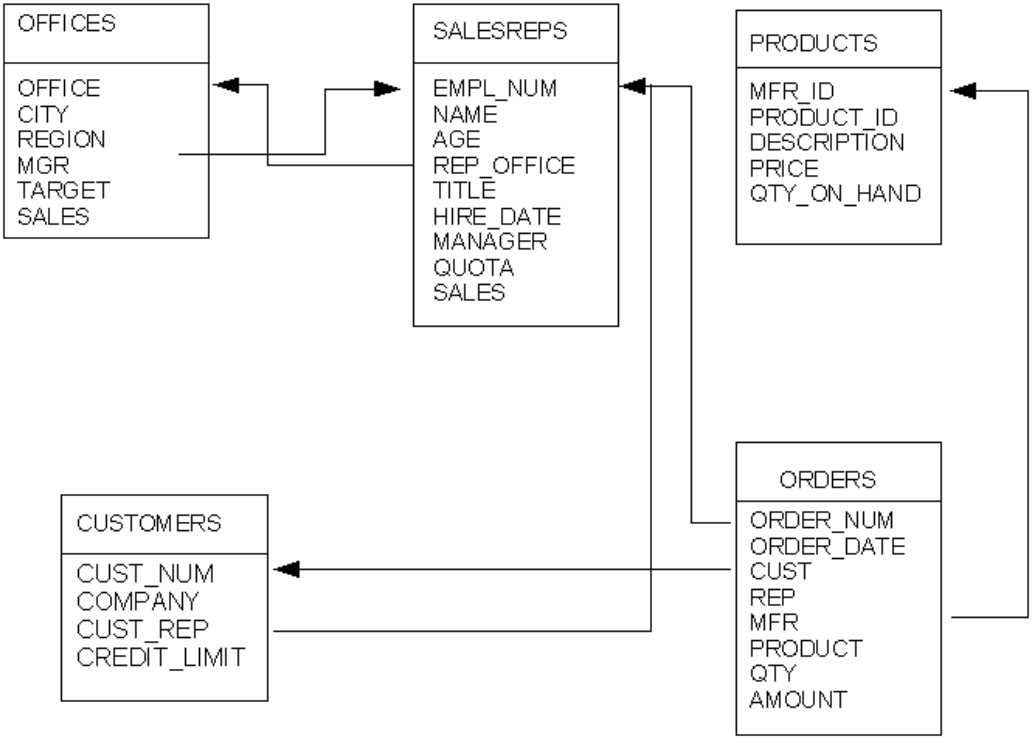
OFFICES (офисы) содержит по одной строке для каждого из пяти офисов компании;

PRODUCTS (товары) содержит по одной строке для каждого наименования товара, продаваемого компанией;

ORDERS (заказы) содержит по одной строке для каждого из заказов, сделанных клиентом. Для простоты считается, что один заказ может содержать один товар.

Эта база данных, как и большинство других, является моделью «реального мира». Данные, содержащиеся в ней, представляют реальные сущности: клиентов, заказы служащих компаний и офисы. Для каждой сущности имеется собственная отдельная таблица. Запросы к базе данных, создаваемые с помощью SQL, отражают события, происходящие в реальном мире: клиенты делают, отменяют и изменяют заказы, владелец компании нанимает и увольняет служащих и т.д. Работа с данными осуществляется посредством SQL.

Структура учебной базы данных



Содержание таблицы Customers

CUST_NUM	*COMPANY*	*CUST_REP*	*CREDIT_LIMIT*
2111	JCP Inc.	103	5000000
2102	First Corp.	101	6500000
2103	Acme Mfg.	105	5000000
2123	Carter & Sons	102	4000000
2107	Ace International	110	3500000
2115	Smithson Corp.	101	2000000
2101	Jones Mfg.	106	6500000
2112	Zetacorp	108	5000000
2121	QMA Assoc.	103	4500000
2114	Orion Corp.	102	2000000
2124	Peter Brothes	107	4000000
2108	Hoim & Land's	109	5500000
2117	J. P. Sinclair	106	3500000
2122	Three-Way Lines	105	3000000
2120	Rico Enterprises	102	5000000
2106	Fred Lewis Corp.	102	6500000
2119	Solomon Inc.	109	2500000
2118	Midmest Systems	108	6000000
2113	Ian & Schmidt	104	2000000
2109	Chen Associates	103	2500000
2105	AAA Investments	101	4500000

CUST_NUM – идентификационный номер клиента;
COMPANY – название компании-клиента;
CUST_REP – идентификационный номер сотрудника, с которым работает покупатель;
CREDIT_LIMIT – сумма кредита для данного клиента

Содержание Salesreps								
EMPL_ NUM	*NAME*	*AGE*	*REP_OF FICE*	*TITLE*	*HIRE_DATE*	*MAN- AGER*	*QUOTA*	*SALES*
105	Bill Adams	37	13	Sales Rep	12-FEB-88	104	35000000	36791100
109	Marry Jones	31	11	Sales Rep	12-OCT-89	106	30000000	39272500
102	Sue Smith	48	21	Sales Rep	10-DEC-86	108	35000000	47405000
106	Sam Clark	52	11	VP Sales	14-JUN-88		27500000	29991200
104	Bob Smith	33	12	Sales Mgr	19-MAY-87	106	20000000	14259400
107	Dan Roberts	45	12	Sales Rep	20-OCT-86	104	30000000	30567300
110	Tom Snyder	41		Sales Rep	13-JAN-90	101		7598500
108	Larry Fitch	62	21	Sales Mgr	12-OCT-89	106	35000000	36186500
103	Paul Cruz	29	12	Sales Rep	01-MAR-87	104	27500000	28677500
107	Nancy Angell	49	22	Sales Rep	14-NOV-88	108	30000000	18604200

EMPL_NUM – идентификационный номер сотрудника;
NAME – имя сотрудника;
AGE – возраст;
REP_OFFICE – номер офиса;
TITLE – должность сотрудника;
HIRE_DATE – дата приема на работу;
MANAGER – руководитель офиса, к которому приписан сотрудник;
QUOTA – план продаж;
SALES – объем продаж

Содержание таблицы Orders							
ORDER_NUM	*ORDER_DATE*	*CUST*	*REP*	*MFR*	*PRODUCT*	*QTY*	*AMOUNT*
1	2	3	4	5	6	7	8
112961	17.12.1989	2117	106	REI	2A44L	7	3150000
113012	11.01.1990	2111	103	ACI	41003	35	374500
112989	03.01.1990	2101	106	FEA	114	6	145800
113051	10.02.1990	2118	108	QSA	XK47	4	142000
112968	12.10.1989	2102	101	ACI	41004	34	397800
113036	30.01.1990	2107	110	ACI	4100Z	9	2255000
113045	02.02.1990	2112	108	REI	2A44R	10	4500000
112963	17.12.1989	2103	105	ACI	41004	28	327600
113013	14.01.1990	2118	108	BIC	41003	1	65200
113058	23.02.1990	2108	109	FEA	112	10	148000
112997	08.01.2009	2124	107	BIC	41003	1	65200
112983	27.12.1989	2103	105	ACI	41004	6	70200
113024	20.01.1990	2114	102	QSA	XK47	20	710000
113062	24.02.1990	2124	107	FEA	114	10	243000
112979	12.10.1989	2114	102	ACI	4100Z	6	1500000
113027	22.01.1990	2103	105	ACI	41002	54	410400
113007	08.01.1990	2112	108	IMM	773C	3	292500
113069	02.03.1990	2109	103	IMM	775C	22	3135000
113034	29.01.1990	2107	110	REI	2A45C	8	63200
112992	04.11.1989	2118	108	ACI	41002	10	76000

Окончание таблицы

1	2	3	4	5	6	7	8
112975	12.11.1989	2111	103	REI	2A44G	6	210000
113055	15.02.1990	2108	109	ACI	4100X	6	15000
113048	10.02.1990	2120	102	IMM	779C	2	375000
112993	04.01.1989	2106	102	REI	2A45C	24	189600
113065	27.02.1990	2106	102	QSA	XK47	6	213000
113003	25.01.1990	2108	109	IMM	779C	3	562500
113049	10.02.1990	2118	108	QSA	XK47	2	71000
112987	31.12.1989	2103	105	ACI	4100Y	10	2750000
113057	18.02.1990	2111	103	ACI	4100X	24	60000
	02.02.1990	2113	104	REI	2A44R	5	2250000

ORDER_NUM – номер заказа;
ORDER_DATE – дата заказа;
CUST – идентификационный номер клиента;
REP – идентификационный номер сотрудника;
MFR – идентификатор изготовителя продукта;
PRODUCT – идентификационный номер продукта;
QTY – количество;
AMOUNT – стоимость

Содержание таблицы Offices					
OFFICE	*CITY*	*MGR*	*REGION*	*TARGET*	*SALES*
22	Denver	108	Western	30000000	18604200
11	New York	106	Eatern	57500000	69263700
12	Chicago	104	Eatern	80000000	73504200
13	Atlanta	105	Eatern	35000000	36791100
21	Los Angeles	108	Western	72500000	83591500

OFFICE – номер офиса;
CITY – город;
MGR – идентификационный номер руководителя офиса;
SALES – объем продаж офиса;
REGION – район;
TARGET – план продаж офиса

Содержание таблицы Products				
MFR_ID	*PRODUCT_ID*	*DESCRIPTION*	*PRICE*	*QTY_ON_HAND*
1	2	3	4	5
REI	2A45C	Ratchet Link	7900	210
ACI	41DDY	Widget Remover	275000	25
QSA	XK47	Reducer	35500	38
BIC	41672	Plate	18000	0
IMM	779C	900-ib Brace	187500	9
ACI	41003	Size 3 Widget	10700	207
ACI	41004	Size 4 Widget	11700	139

Окончание таблицы

1	2	3	4	5
BIC	41003	Handle	65200	3
IMM	887P	Brace Pin	25000	24
QSA	XK48	Reducer	13400	203
REI	2A44L	Left Hinge	450000	12
FEA	112	Housing	14800	115
IMM	887H	Brace Holder	5400	223
BIC	41D89	Retainer	22500	78
ACI	41DD1	Size 1 Widget	5500	277
IMM	775C	500-ib Brace	142500	5
ACI	41DDZ	Widget Installer	250000	28
QSA	XK48A	Reducer	17700	37
ACI	41DD2	Size 2 Widget	7600	167
REI	2A44R	Right Hinge	450000	12
IMM	773C	3DD-ib Brace	97500	28
ACI	41DDX	Widget Adjuster	2500	37
FEA	114	Motor Mount	24300	15
IMM	887X	Brace Retainer	47500	32
REI	2A44G	Hinge Pin	35000	14

MFR_ID – идентификационное название продукта; DESCRIPTION – описание товара; PRODUCT_ID – идентификационный номер; PRICE – стоимость; QTY_ON_HAND – количество товара на складе

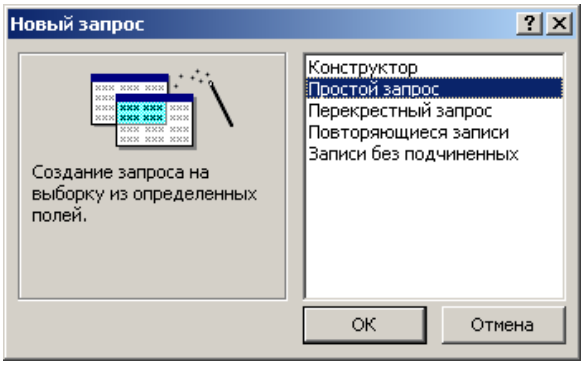
Лабораторная работа № 2

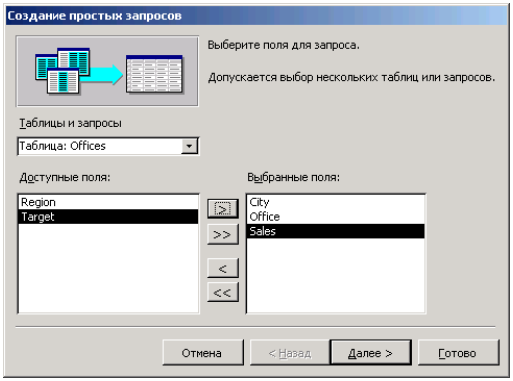
Создание простого запроса. Вычисляемые столбцы

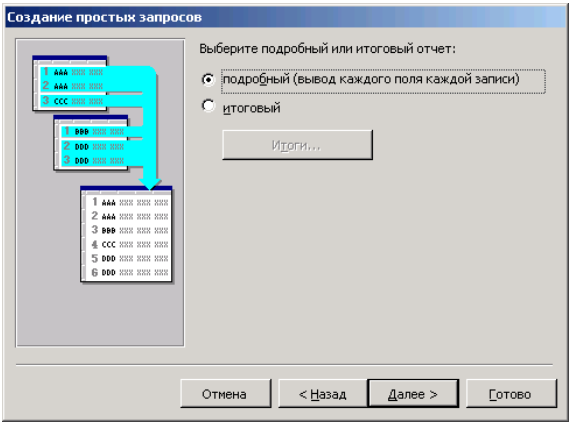
Запрос – команда, которую вы даете вашей программе базы данных, и которая сообщает ей, чтобы она вывела определенную информацию из таблиц в память. Эта информация обычно посылается непосредственно на экран компьютера или терминала, которым вы пользуетесь, хотя, в большинстве случаев, ее можно также послать принтеру, сохранить в файле (как объект в памяти компьютера), или представить как вводную информацию для другой команды или процесса.

Выбираем объект «Запросы» и создаем запрос с помощью мастера («Создание запроса с помощью мастера»).

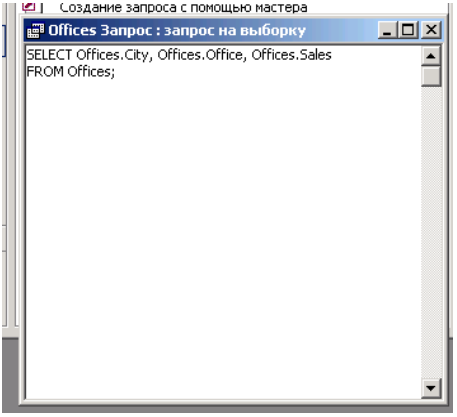
Выберите «Простой запрос».







Просмотрим содержание SQL запроса
Вид -> Режим SQL
SELECT Offices.City, Offices.Office, Offices.Sales
FROM Offices;



В общем случае команда выборки данных начинается с ключевого слова SELECT. После этого должен следовать список имен столбцов, которые вы хотите видеть, отделяемых запятыми. После ключевого слова FROM следует имя таблицы, запрос к которой делается. В заключение должна использоваться точка с запятой (;), чтобы закончить запрос и указать, что команда готова к выполнению.

Инструкция SELECT запрашивает для каждого офиса три вида данных: город, номер офиса и объем продаж. Также она определяет, что данные находятся в таблице OFFICES, в которой хранится информация об офисах.

Изменим запрос на:

SELECT Offices.City, Offices.Sales

FROM Offices.

Сохраняем изменения и нажимаем кнопку запуск (восклицательный знак или через меню: Запрос> запуск).

Вычисляемые столбцы

Кроме столбцов, значения которых непосредственно извлекаются из базы данных, SQL позволяет также запрашивать вычисляемые результаты, на основании значений, хранящихся в базе данных. Например, можно попросить вычислить сумму, на которую каждый служащий опережает план или отстает от него:

SELECT name, sales, quota, (sales-quota)

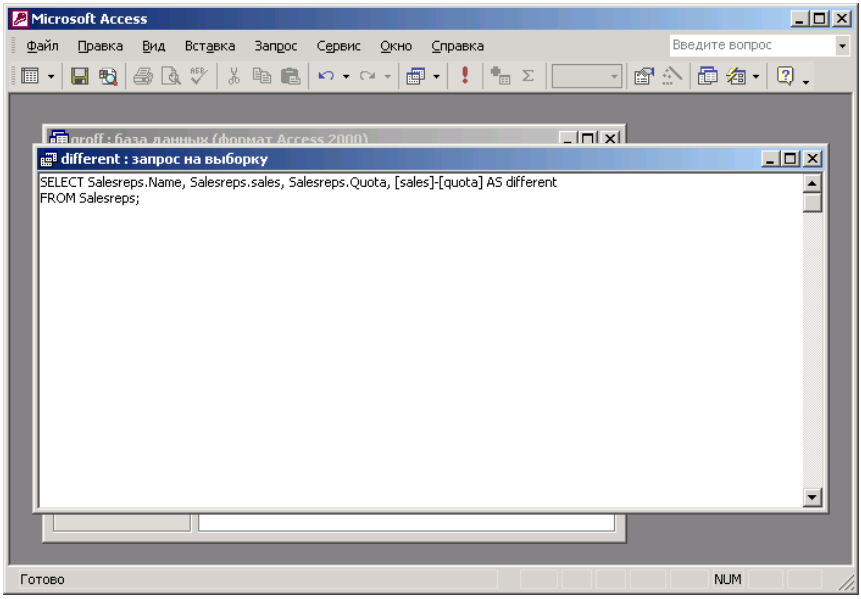
FROM salesreps.

different : запрос на выборку

	Name	sales	Quota	Выражение1
▶	Bill Adams	367 911,00р.	350 000,00р.	17 911,00р.
	Mary Jones	392 725,00р.	300 000,00р.	92 725,00р.
	Sue Smith	474 050,00р.	350 000,00р.	124 050,00р.
	Sam Clark	299 912,00р.	275 000,00р.	24 912,00р.
	Bob Smith	142 594,00р.	200 000,00р.	-57 406,00р.
	Dan Roberts	305 673,00р.	300 000,00р.	5 673,00р.
	Tom Snyder	75 985,00р.	0,00р.	75 985,00р.
	Larry Fitch	361 865,00р.	350 000,00р.	11 865,00р.
	Paul Cruz	286 775,00р.	275 000,00р.	11 775,00р.
	Nancy Angell	186 042,00р.	300 000,00р.	-113 958,00р.
*		0,00р.	0,00р.	

Чтобы избавится от заголовка «Выражение1» открываем запрос в режиме SQL и редактируем следующим образом.

SELECT Name, Ssales, Quota, [Sales]-[Quota] AS different
FROM Salesreps.



Задания

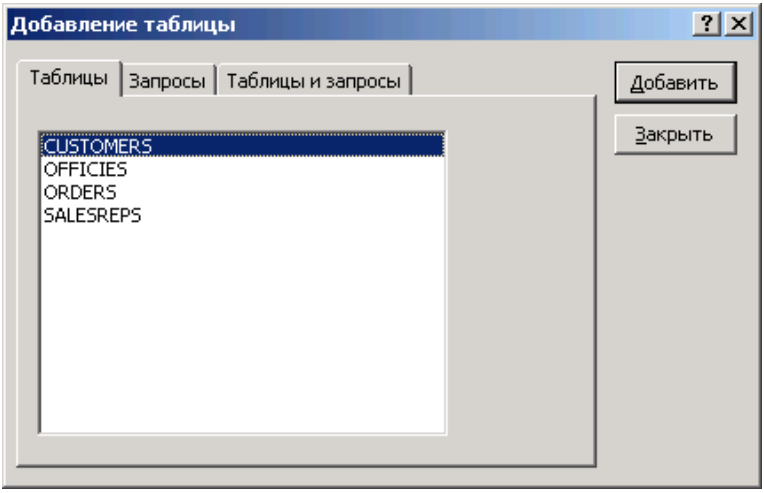
1. Из таблицы Customers выбрать все компании-клиенты и идентификационные номера сотрудников, с которыми они работают.
2. Из таблицы Customers выбрать идентификационный номер и название компании клиента.
3. Из таблицы Offices выбрать номер офиса и город, в котором он расположен.
4. Составить запрос для отображения идентификационного номера служащего и его имени.
5. Отобразить номер заказа, его дату и стоимость.
6. Составить запрос для отображения идентификационного номера продукта и его описания.
7. Составить запрос для отображения идентификационного номера продукта, идентификационного названия продукта и краткого описания.
8. Составить запрос для отображения идентификационного номера продукта, его описания и стоимости.
9. Составить запрос для отображения идентификационного номера продукта, его описания и количества единиц товара на складе.
10. Составить запрос для отображения идентификационного номера продукта, его описания и стоимости каждого товара, хранящегося на складе.

Лабораторная работа № 3

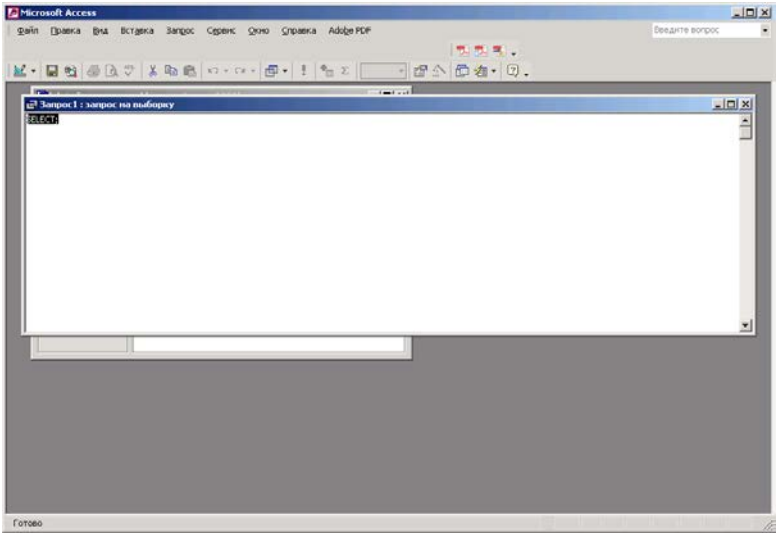
Создание запроса для отбора строк с определенным условием. Использование ключевых слов ORDER BY и TOP

Чтобы непосредственно перейти в окно редактирования SQL запроса необходимо проделать следующее:

Запросы> Создать> Конструктор. В окне «*Добавление таблицы*» нажимаем «*Закрыть*».



Нажимаем *Вид* > *Режим SQL*.



Язык SQL позволяет выполнить выборку с определенным условием. Для этого служит ключевое слово WHERE. Условие, следующее после ключевого слова WHERE, отфильтровывает ненужные данные из результирующего набора, возвращая

только те записи, которые удовлетворяют условиям поиска, указываемым после данного ключевого слова.

Для отображения служащих, которые не выполнили план, необходимо добавить операцию сравнения:

SELECT Salesreps.Name, Salesreps.sales, Salesreps.Quota, [sales]-[quota] AS different

FROM Salesreps

WHERE sales<quota.

С помощью этого же приема можно получить список больших заказов и определить, кто сделал конкретный заказ, какие товары и в каких количествах были заказаны. Также SQL разрешает произвести сортировку заказов по их стоимости:

SELECT order_num, cust, product, qty, amount

FROM orders

WHERE amount > 25000

ORDER BY amount.

Ключевая фраза ORDER BY указывает способ сортировки результирующего набора по возрастанию (по умолчанию, или с указанием ключевого слова ASC) или по убыванию (DESC) с указанием выражения для сортировки, которое представляет собой список столбцов, разделенных запятыми, по которым нужно производить сортировку.

Часто бывает достаточно отобразить несколько первых элементов таблицы. Для этого служит ключевое слово TOP. Модифицируем предыдущий запрос для отображения десяти максимальных заказов. Кроме того, указано, что вторым кри-

терием сортировки является столбец с номером заказа (сортировка происходит по убыванию).

SELECT TOP 10 order_num, cust, product, qty, amount
FROM orders
WHERE amount > 25000
ORDER BY amount, order_num DESC

Показать общую стоимость по каждому товару:

*SELECT mfr_id, product_id, description, (qty_on_hand*price) AS total*
FROM products;

Вывести список имен, а также месяц и год приема на работу всех служащих.

SELECT name, MONTH(hire_date), YEAR(hire_date) FROM salesreps

Иногда требуется получить содержимое всех столбцов таблицы. Для этого используется символ (*), который означает, что требуется извлечь все столбцы.

Показать все данные, содержащиеся в таблице *offices*:

*SELECT * FROM offices*

Задания

1. Вывести список офисов с их плановыми и фактическими объемами продаж. Отсортировать список по возрастанию плановых объемов продаж.
2. Вывести список офисов с их плановыми и фактическими объемами продаж. Отсортировать список по убыванию фактических объемов продаж.
3. Отобразить имена служащих, отсортировав в алфавитном порядке.
4. Отобразить номер заказа, стоимость и дату, начиная с самых последних.
5. Вывести список офисов, расположенных в восточном регионе, с их плановыми и фактическими объемами продаж.

6. Вывести список офисов, расположенных в восточном регионе, в которых фактические объемы продаж превысили плановые. Отсортировать список в алфавитном порядке по названиям городов.

7. Что получится, если увеличить плановый объем продаж для каждого служащего на 3 % от его фактического объема продаж?

8. Отобразить все данные по офисам, в которых фактические объемы продаж превысили плановые.

9. Отобразить заказы, стоимость которых более $300000, отсортировав их по дате заказа.

10. Отобразить список служащих, у которых фактические продажи более $10000000, начиная с наибольших.

Лабораторная работа № 4

Составные условия выбора (AND, OR, NOT). Создание запроса на выборку даты в ACCESS. Ключевое слово BETWEEN

Оператор OR используется для объединения двух условий отбора, из которых как минимум одно должно быть истинным.

Найти служащих, у которых фактический объем продаж меньше планового или меньше $300 000.

SELECT name, quota, sales FROM salesreps WHERE sales< quota OR sales < 300000

Оператор AND используется для объединения двух условий отбора, оба из которых должны быть истинными.

Найти служащих, у которых фактический объем продаж меньше планового и меньше $300 000.

SELECT name, quota, sales FROM salesreps WHERE sales< quota AND sales < 300000

Оператор NOT следует использовать, чтобы выбрать строки, для которых условие отбора ложно.

Найти служащих, у которых фактический объем продаж меньше планового, но больше $150000.

SELECT name, quota, sales FROM salesreps WHERE sales< quota AND NOT sales < 150000

Создание запроса на выборку даты в ACCESS

Выбрать заказы, сделанные за период с 10 декабря 1989 до 13 декабря 1989. Следует обратить внимание на формат задания даты.

SELECT Product FROM orders WHERE order_date BETWEEN #12/10/1989# AND #12/13/1989#;

Оператор BETWEEN определяет диапазон, из которого происходит выборка. Вы должны ввести ключевое слово BETWEEN с начальным значением, ключевое AND и конечное значение. BETWEEN чувствителен к порядку, и первое значение в предложении должно быть первым по алфавитному или числовому порядку. При проверке на принадлежность диапазону начальное и конечное значение считаются его частью.

Вывести список служащих, фактические объемы продаж которых не попадают в диапазон от 80 до 120 % плана.

*SELECT name, sales, quota FROM salesreps WHERE sales NOT BETWEEN (0.8*quota) AND (1.2*quota)*

Задания

1. Найти всех служащий, которые: (*a*) были приняты на работу после июня 1988; или (*b*) превысили плановый объем продаж, но не достигли уровня в 600 000.
2. Выбрать все товары (таблица *products*), стоимость которых меньше 1000, и количество которых превышает 100 шт.
3. Вывести суммарную стоимость товаров на складе.
4. Выбрать товары, стоимость которых больше 500, а количество меньше 100.
5. Отобрать заказы, сделанные в ноябре 1990 г.

6. Отобрать заказы, сделанные с сентября 1989 г. по октябрь 1990 г.

7. Отобрать заказы, сделанные в сентябре 1990 г. и январе 1989 г.

8. Отобразить все заказы, сделанные в зимние месяцы.

9. Выбрать служащих, принятых на работу в 1986 или в 1990 гг.

10. Выбрать компании, которые работают с офисом 102 или 103.

Лабораторная работа № 5

Статистические функции

Для подведения итогов по информации, содержащейся в базе данных, в SQL предусмотрены статистические (агрегатные) функции. Статистическая функция принимает в качестве аргумента какой-либо столбец данных целиком, а возвращает одно значение, которое определенным образом подытоживает этот столбец. Например, функция AVG() принимает в качестве аргумента столбец чисел и вычисляет их среднее значение. Ниже приведен запрос, в котором функция AVG() используется для вычисления среднего значения в двух столбцах таблицы *salesreps*.

Каковы средний плановый и средний фактический объемы продаж в нашей компании?

SELECT AVG(quota), AVG (sales) FROM salesreps

В SQL имеется 6 статических функций, которые позволяют получать различные виды итоговой информации:

функция SUM() вычисляет сумму всех значений столбца;

AVG() вычисляет среднее всех значений столбца;

MIN() находит наименьшее среди всех значений столбца;

функция MAX() находит наибольшее среди всех значений столбца;

COUNT() подсчитывает количество значений, содержащихся в столбце;

COUNT(*) подсчитывает количество строк в таблице результатов запроса.

Каков средний процент выполнения плана в компании?

SELECT AVG(100(sales/quota)) FROM salesreps.*

Замечание!!! Возможно появление деления на ноль, в таком случае корректируем запись.

Вычисление суммы значений столбца (функция SUM)

Статистическая функция SUM вычисляет сумму всех значении столбца. При этом столбец должен иметь числовой тип данных (содержать целые, десятичные числа, числа с плавающей запятой или денежные величины). Результат, возвращаемый этой функцией, имеет тот же тип данных, что и столбец, однако точность результата может быть выше. Например, если применить функцию SUM() к столбцу, содержащему 16-разрядные целые числа, она может вернуть в качестве результата 32-разрядное целое число.

Каковы общий плановый и общий фактический объемы продаж в нашей компании?

SELECT SUM (quota), SUM (sales) FROM salesreps

Вычисление среднего значения столбца (функция AVG)

Статистическая функция AVG() вычисляет среднее всех значении столбцов. Как и в случае с функцией SUM(), данные, содержащиеся в столбце, должны иметь числовой тип. Поскольку функция AVG() вначале суммирует все значения, содержащиеся в столбце, а затем делит сумму на число этих значений. Возвращаемый ею результат может иметь не такой тип данных, как столбец. Например, если применить функцию AVG() к столбцу целых чисел, результат будет либо десятичным числом, либо числом с плавающей запятой, в зависимости от используемой СУБД.

Следующий запрос обеспечивает вычисление средней стоимости.

SELECT AVG (amount) FROM orders

Можно также узнать среднюю стоимость всех заказов, сделанных конкретным клиентом.

SELECT AVG (amount) FROM orders WHERE cust=2103

Вычисление экстремумов (функции MIN и MAX)

Статистические функции MIN() и MAX() позволяют найти соответственно наименьшее и наибольшее значения в столбце. При этом столбец может содержать числовые или строковые значения либо значения даты/времени. Результат, возвращаемый этими функциями, имеет точно такой же тип данных, что и сам столбец.

В случае применения функций MIN() и MAX() к числовым данным числа сравниваются по арифметическим правилам (среди двух отрицательных чисел меньше то, у которого модуль больше; нуль меньше любого положительного числа и больше любого отрицательного). Сравнение дат происходит последовательно (более ранние значения считаются меньшими, чем более поздние). Сравнение интервалов времени выполняется на основании их продолжительности (более короткие интервалы времени считаются меньшими, чем более длинные).

В случае применения функций MIN() и MAX() к строковым данным результат сравнения двух строк зависит от используемой таблицы кодировки. На персональных компьютерах, где используется таблица кодировки ASCII, установлен порядок сортировки, при котором цифры идут перед буквами, а все прописные буквы – перед строчными. На мэйнфреймах компании IBM, где используется таблица кодировки EBCDIC, строчные символы расположены перед прописными, а цифры следуют за буквами.

Хранение в таблицах символов национальных алфавитов (например, кириллицы) может вызвать дополнительные проблемы. В некоторых СУБД для каждого языка используется свой ал-

горитм сортировки. В других СУБД такие символы сортируются в соответствии с кодом символа. Дли решения этой проблемы в стандарт SQL2 включено условие поддержки национальных наборов символов, пользовательских наборов символов и альтернативных последовательностей сортировки. Если в приложении используются символы национальных алфавитов, необходимо проверить, как конкретная СУБД обрабатывает их.

Вычисление количества значений в столбце (функция COUNT)

Статистическая функция COUNT() подсчитывает количество значений в столбце. При этом тип данных столбца может быть любым. Функция COUNT() всегда возвращает целое число независимо от типа данных столбца. Ниже приведен ряд запросов, в которых используется эта функция

Сколько клиентов у нашей компании?

SELECT COUNT (cust_num) FROM customers

Мысленно трудно представить запрос вроде «подсчитать, сколько стоимостей заказов» или «подсчитать, сколько номеров заказов». Гораздо проще представить запрос «подсчитать, сколько заказов». Поэтому в SQL была введена специальная статистическая функция COUNT(*), которая подсчитывает строки, а не значения данных.

Сколько имеется заказов более 250000?

SELECT COUNT (*) FROM Orders WHERE amount > 250000

Один из лучших способов понять, как выполняются итоговые запросы со статистическими функциями – представить запрос разбитым на два этапа. Сначала подумайте, как работал бы запрос без статистических функций, возвращая несколько строк результатов. Затем представьте, как СУБД применяет статистические функции к результатам запроса, возвращая одну итоговую строку.

Задания

1. Какова сумма всех заказов, принятых Биллом Адамсом (Bill Adams)?

2. Вычислить среднюю цену товаров от производителя ACI.

3. Вычислить среднюю стоимость заказов, сделанных компанией Acme Mfg (идентификатор клиента 2103).

4. Каковы наибольший и наименьший плановые объемы продаж?

5. Когда был сделан самый первый из всех содержащихся в базе данных заказов?

6. Какой наибольший процент выполнения плана среди служащих?

7. Сколько служащих перевыполнили план?

8. Сколько имеется заказов более 25000?

9. Сколько служащих в восточном регионе?

10. Сколько всего было сделано заказов?

Лабораторная работа № 6

Повторяющиеся строки (предикат DISTINCT). Проверка на членство во множестве (оператор IN) и на соответствие шаблону (оператор LIKE)

Удаление повторяющихся строк (предикат DISTINCT)

Если в списке возвращаемых столбцов указать первичный ключ таблицы, то каждая строка результатов запроса будет уникальной (из-за того, что значения первичного ключа во всех строках таблицы разные). Если же первичный ключ не указан, результаты запроса могут содержать повторяющиеся строки.

Вывести список идентификаторов всех менеджеров офисов.

SELECT mgr FROM offices.

Т.к. Larry Fitch является менеджером двух офисов (в Лос-Анджелесе и Денвере), то его идентификатор отображается дважды.

Повторяющиеся строки из результата запроса можно удалить, если использовать предикат DISTINCT перед списком возвращаемых столбцов:

SELECT DISTINCT mgr FROM offices

Предикат DISTINCT указывается в начале списка возвращаемых столбцов и служит для удаления повторяющихся строк из таблицы результатов запроса. С помощью этого предиката можно также указать, что перед применением статистической функции к столбцу из него следует удалить все повторяющиеся значения. Для этого необходимо включить предикат DISTINCT перед аргументом статистической функции сразу же после открывающейся круглой скобки.

Отобразить различные должности, существующие в компании:

SELECT DISTINCT title FROM salesreps.

Еще одним распространенным условием отбора является проверка на членство в множестве (*оператор IN*). В этом случае проверяется, соответствует ли элемент данных какому-либо значению из заданного списка.

Вывести список, план и фактические продажи служащих, работающих в Нью-Йорке, Атланте или Денвере.

SELECT name, quota, sales FROM salesreps WHERE rep_office IN (11, 12, 13).

С помощью проверки NOT IN можно убедиться в том, что элемент данных не является членом заданного множества.

Некоторые задачи нельзя решить только с помощью операторов сравнения. Например:

«Его фамилия начинается с Mc или Mac, а дальше я не помню», «Мне нужен список телефонов, начинающихся с 415», «я не помню, как точно пишется его фамилия – Carson или Karsen».

В каждом из этих случаев известен некоторый образец, где-то встречающийся в столбце, и с его помощью нужно извлечь всю или часть соответствующей ему строки. Для решения этой задачи и предназначено ключевое слово LIKE. Его можно применять к символьным полям, а в некоторых системах – и к полям даты. Однако ключевое слово LIKE не работает с числовыми полями (целыми, денежными, десятичными и плавающими). Его синтаксис имеет следующий вид:

WHERE имя_столбца [NOT] LIKE 'образец' [ESCAPE ключевой символ]

Образец должен заключаться в кавычки и может содержать один или несколько шаблонов-символов, замещающих в образце пропущенные буквы или строки. Ключевое слово ESCAPE используется в том случае, если в образце содержится один из шаблонов, но его нужно рассматривать как простой литерал. ANSI SQL предусматривает использование двух символов шаблона совместно с ключевым словом LIKE – знак процента (%) и подчеркивание (_). Знак процента (%) замещает любую строку с любым количеством символов, подчеркивание (_) замещает любой одиночный символ.

Показать лимит кредита для Smithson Corp.

SELECT company, credit_limit FROM customers WHERE company = 'Smithson Corp'

Однако очень легко можно забыть, какое именно название носит интересующая нас компания: "Smith", "Smithson" или "Smithsonian". Проверка на соответствие шаблону позволяет выбрать из базы данных строки на основе частичного соответствия имени клиента.

Показать лимит кредита для Smithson Corp. (использование шаблона).

SELECT company, credit_limit FROM customers WHERE company LIKE 'Smith% Corp'

Замечание. Однако в разных системах могут использоваться разные символы шаблонов. В MS ACCESS, например, символ шаблона (*) – для любого количества символов и (?) – для одиночного символа.

Задания

1. Отобразить регионы, где расположены офисы.
2. В каких офисах (города) есть служащие, превысившие плановые объемы продаж?
3. Найти все заказы, сделанные по вторникам в январе 1990 года.

4. Найти все заказы, полученные четырьмя конкретными служащими (107, 109, 101, 103).

5. Вывести список сотрудников, чье имя состоит из трех букв.

6. Вывести список сотрудников, чья фамилия состоит из четырех букв.

7. Отобразить компании-клиенты, работающие с офисами 101, 102, 104.

8. Отобразить компании (и их идентификационные номера), в названии которых присутствует слово “Corp”.

9. Отобразить заказы, сделанные в офисах 105, 107,109.

10. Отобразить компании-клиенты, не работающие с офисами 103, 105, 107 и не содержащие в своем названии слово “Corp”.

Лабораторная работа № 7

Объединение результатов нескольких запросов (операция UNION)

Иногда появляется необходимость объединения результатов двух или более запросом в одну таблицу. SQL поддерживает такую возможность с помощью операции UNION (рис.7.1)

Вывести список всех товаров, цена которых превышает $2000 или которых было заказано более чем на $30000 за один раз.

Первой части основного запроса удовлетворяет запрос, изображенный в верхней части рисунка.

Вывести список всех товаров, цена которых превышает $2000 (14 записей).

SELECT mfr_id, product_id FROM products WHERE price >2000

Подобным же образом вторую часть основного запроса можно выполнить с помощью запроса, изображенного в нижней части рис.7.1.

Вывести список всех товаров, которых было заказано более чем на $30000 за один раз (17 записей).

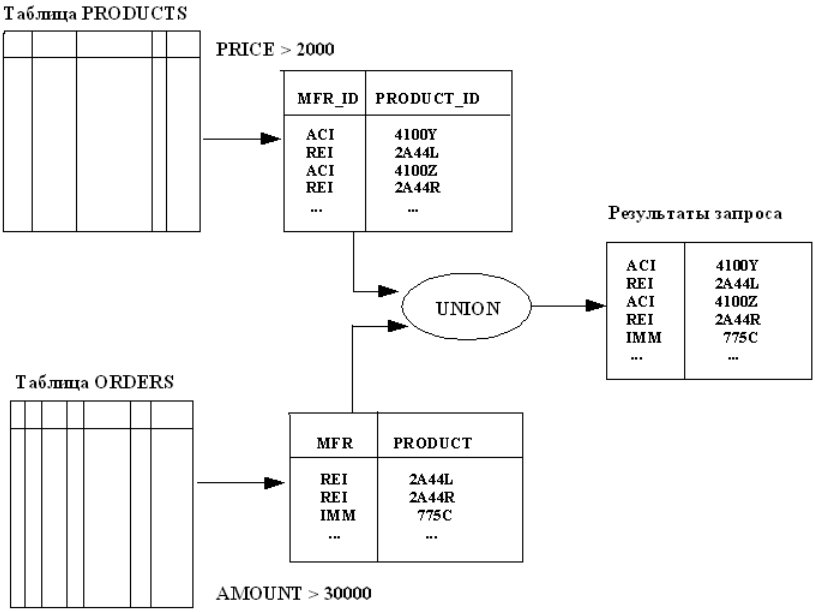


Рис 7.1. Схема работы операции UNION

SELECT DISTINCT mfr, product FROM orders WHERE amount > 30000

Как изображено на рис. 7.1, операция UNION создает одну таблицу результатов запроса, в которой содержатся строки результатов как верхнего, так и нижнего запросов. В запросах операция UNION используется следующим образом.

Вывести список всех товаров, цена которых превышает $2000 или которых было заказано более чем на $30000 за один раз (29 записей).

SELECT mfr_id, product_id FROM products WHERE price >2000.00
UNION
SELECT DISTINCT mfr, product FROM orders WHERE amount > 30000

Чтобы таблицы результатов запроса можно было объединить с помощью операции UNION, они должны соответствовать следующим требованиям:

– две таблицы должны содержать одинаковое число столбцов;

– тип данных каждого столбца первой таблицы должен совпадать с типом данных соответствующего столбца во второй таблице;

– таблицы не могут быть отсортированы с помощью предложения ORDER BY, однако объединенные результаты запроса можно отсортировать, как будет описано ниже.

Обратите внимание на то, что имена столбцов в двух запросах, объединенных с помощью операции UNION, не обязательно должны быть одинаковыми. В предыдущем примере в первой таблице результатов запроса имеются столбцы с именами *mfr_id* и *product_id* в то время, как во второй таблице результатов запроса соответствующие им столбцы имеют имена *mfr* и *product*. Поскольку столбцы в двух таблицах могут носить различные имена, столбцы результатов запроса, возвращенные операцией UNION, являются безымянными.

Стандарт ANSI/ISO накладывает дополнительные ограничения на инструкции SELECT, участвующие в операции UNION. Он разрешает использовать в списке возвращаемых столбцов только имена столбцов или указатель на все столбцы (SELECT *) и запрещает использовать выражения. В большинстве СУБД, тем не менее, это ограничение ослаблено, и в списке возвращаемых столбцов разрешено использовать простые выражения. Однако во многих СУБД не разрешается включать в инструкции SELECT предложения GROUP BY или HAVING, а в некоторых не разрешается использовать в списке возвращаемых столбцов статистические функции.

Запрос на объединение и повторяющиеся строки

Поскольку операция UNION объединяет строки из двух результатов запросов, вполне вероятно, что в полученной таб-

лице будут содержаться повторяющиеся строки. Например, в запросе, представленном на рис. 7.1, стоимость товара REI-2A44L составляет $4500, поэтому он входит в результаты верхнего запроса из числа представленных на рисунке. Кроме того, в таблице ORDERS для этого товара имеется заказ на сумму $3l500, поэтому он входит и в результаты второго запроса. По умолчанию операции UNION в процессе своего выполнения удаляет повторяющиеся строки. Таким образом, в объединенных результатах запроса содержится только одна строка для товара REI-2A44L.

Если в таблице результатов операции UNION необходимо сохранить повторяющиеся строки, сразу за ключевым словом UNION следует указать предикат ALL. Эта форма запроса вернет таблицу с повторяющимися строками.

Вывести список всех товаров, цена которых превышает $2000 или которых было заказано более чем на $30000 за один раз.

SELECT mfr_id, product_id FROM products WHERE price >2000.00
UNION ALL
SELECT DISTINCT mfr, product FROM orders WHERE amount > 30000

Обратите внимание на то, что обработка повторяющихся строк в операции UNION и инструкции SELECT осуществляется по-разному. Инструкция SELECT по умолчанию оставляет такие строки (SELECT ALL). Чтобы удалить их, необходимо явно указать предикат DISTINCT. Операция UNION по умолчанию удаляет повторяющиеся строки. Чтобы оставить их, следует явно задать предикат ALL.

Запрос на объединение и сортировка

Предложение ORDER BY нельзя использовать ни в одной из инструкций SELECT, объединенных операцией UNION. Нет смысла выполнять сортировку результатов таких запросов, поскольку ее все равно не заметно в чистом виде. Однако объеди-

ненные результаты запросов, возвращаемые операцией UNION, можно отсортировать с помощью предложения ORDER BY, следующего за второй инструкцией SELECT. Поскольку столбцы таблицы результатов запросов на объединение не имеют имен, в этом предложении следует указывать номера столбцов.

Ниже показан тот же запрос, в котором результаты дополнительно отсортированы по производителю и номеру товара.

Вывести список всех товаров, цена которых превышает $2000 или которых было заказано более чем на $30000 за один раз, список отсортировать по наименованию производителя и номеру товара.

SELECT mfr_id, product_id FROM products WHERE price >2000.00
UNION
SELECT DISTINCT mfr, product FROM orders WHERE amount > 30000
ORDER BY 1, 2

*Вложенные запросы на объединение **

Операцию UNION можно использовать многократно, чтобы объединить результаты трех или более запросов так, как изображено на рис. 7.2. В результате объединения таблиц *B* и *C* получается одна объединенная таблица. Затем с помощью еще одной операции UNION эта таблица объединяется с таблицей *A*. Синтаксис запроса, представленного на рисунке, имеет следующий вид:

*SELECT * FROM A*
UNION
*(SELECT * FROM B*
UNION
*SELECT * FROM C)*

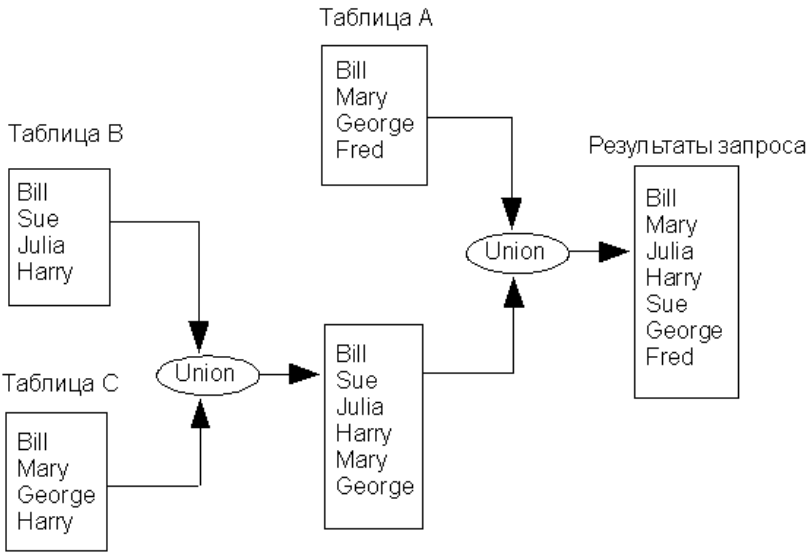


Рис. 7.2. Вложенные операторы UNION

Скобки в запросе показывают, какая операция UNION должна выполняться первой. Независимо от того, удаляют ли все операции UNION повторяющиеся строки или сохраняют их, порядок выполнения инструкций не имеет значения. Следующие выражения полностью эквивалентны и возвращают семь строк результатов запроса.

A UNION ALL (B UNION ALL C)

(A UNION ALL B) UNION ALL C

(A UNION ALL C) UNION ALL B.

Подобным образом три следующих выражения полностью эквивалентны и возвращают 12 строк результатов запроса, поскольку повторяющиеся строки сохраняются:

A UNION (B UNION C)

(A UNION B) UNION C

(A UNION C) UNION B.

Лабораторная работа № 8

Пример двухтабличных запросов. Псевдонимы таблиц

На практике многие запросы считывают информацию сразу из нескольких таблиц базы данных. Например, приведенные ниже запросы к учебной базе данных извлекают данные из двух, трех или четырех таблиц.

Вывести список служащих и офисов, в которых они работают (таблицы *salesreps* и *offises*).

Вывести список заказов, сделанных на прошлой неделе, включая следующую информацию: стоимость заказа, имя клиента, сделавшего заказ, и описание заказанного товара (таблицы *orders*, *customers* и *products*).

Показать все заказы, принятые в восточном регионе, в том числе описания товаров и имена служащих, принявших заказы (таблицы *orders*, *salesreps*, *offices* и *products*).

SQL позволяет получить ответы на эти запросы посредством многотабличных запросов, которые объединяют данные из нескольких таблиц.

Пример двухтабличного запроса

Чтобы понять, как в SQL реализуются многотабличные запросы, лучше всего начать с рассмотрения простого запроса, который объединяет данные из двух различных таблиц.

Вывести список всех заказов, включая номер и стоимость заказа, а также имя и лимит кредита клиента, сделавшего заказ.

Из рис. 8.1 видно, что четыре запрашиваемых элемента данных хранятся в двух различных таблицах.

В таблице *orders* содержится номер и стоимость каждого заказа, но в ней отсутствуют имена клиентов и лимиты предоставленных им кредитов.

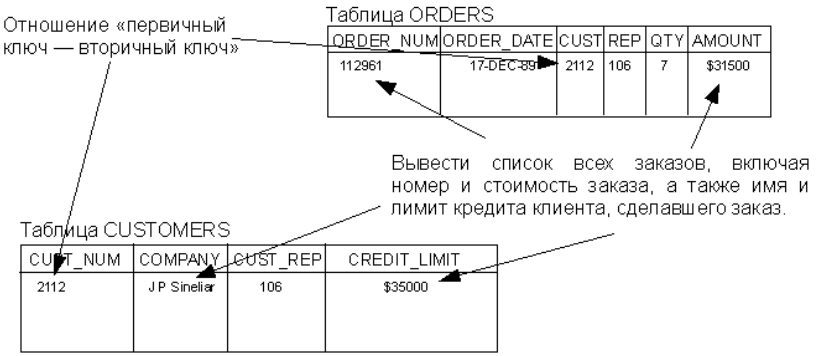


Рис. 8.1. Двухтабличный запрос

В таблице *customers* содержатся имена клиентов и данные о состоянии их счетов, но в ней нет информации о заказах.

Однако между двумя этими таблицами существует связь. В каждой строке столбца *cust* таблицы *orders* содержится идентификатор клиента, сделавшего заказ, соответствующий значению одной из строк столбца *cust_num* таблицы *customers*. Очевидно, чтобы получить требуемые результаты, в инструкции SELECT, с помощью которой осуществляется запрос, необходимо как-то учесть эту связь между таблицами.

Прежде, чем рассматривать инструкцию SELECT, выполняющую этот запрос, будет полезно обдумать, как бы вы выполнили этот запрос вручную с помощью карандаша и бумаги. На рис. 8.2 изображены действия, которые предположительно придется выполнить.

1. Сначала нарисуйте таблицу для результатов запроса, содержащую четыре столбца, и запишите имена столбцов. Затем перейдите к таблице *orders* и начните с первого заказа.

2. Найдите в строке для первого заказа его номер (112961) и стоимость ($31500), а затем перепишите оба значения в первую строку таблицы результатов.

3. В строке для первого заказа найдите идентификатор клиента, сделавшего заказ (2117) Перейдите к таблице *customers* и в

столбце *cust_num* найдите строку с идентификатором клиента 2117.

4. В данной строке таблицы *customers* найдите имя клиента (J.P. Sinclair) и лимит кредита ($35500) и перепишите их в таблицу результатов.

5. Вы создали только одну строку результатов запроса. Вернитесь к таблице *orders* и принимайтесь за следующую строку. Повторяйте процесс, начинал с пункта 2, до тех пор, пока не переберете все заказы.

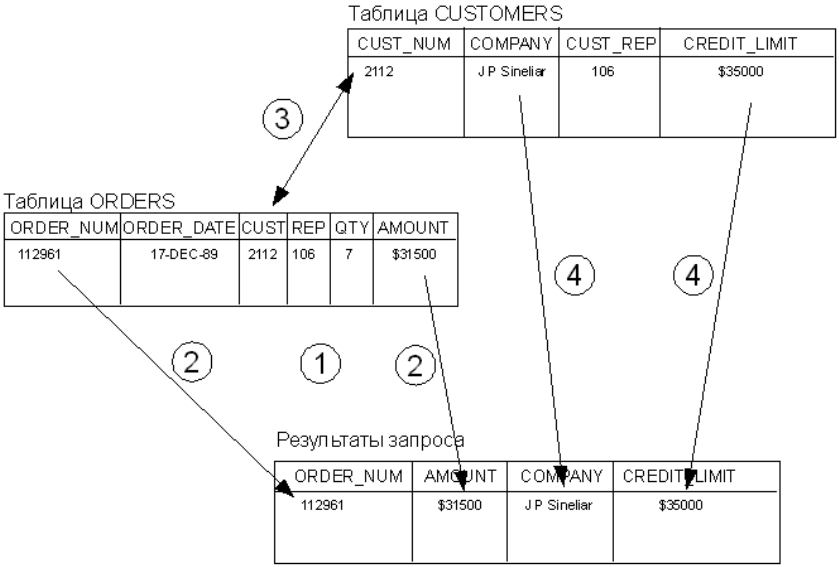


Рис 8.2. Схема выполнения многотабличного запроса

Это не единственный способ получить результаты данного запроса. Но как бы вы их не получали, всегда будут справедливы две вещи:

– каждая строка таблицы результатов запроса формируется из *пары* строк: одна строка относится к таблице *orders*, а другая – к таблице *customers*;

– для нахождения данной пары строк производится сравнение содержимого *соответствующих столбцов* в этих таблицах.

Простое объединение таблиц (объединение по равенству)

Процесс формирования пар строк *путем сравнения* содержимого соответствующих столбцов называется *объединением таблиц.* Таблица, получающаяся в результате формирования пар строк и содержащая данные из обеих таблиц, называется *объединением* двух таблиц. Объединение на основе точного равенства между двумя столбцами более традиционно называется *объединением по равенству.* Объединения могут быть основаны на других видах сравнения столбцов.

Объединения представляют собой основу многотабличных запросов в SQL. В реляционной базе данных вся информация хранится и виде табличных значений данных в столбцах так, что все возможные отношения между таблицами можно сформировать, сопоставляя содержимое соответствующих столбцов. Таким образом, объединения являются мощным (и к тому же единственным, ввиду отсутствия указателей) средством выявления отношений, существующих между данными.

Так как в SQL многотабличные запросы наполняются путем сопоставления столбцов, неудивительно, что инструкция SELECT для многотабличного запроса должна содержать условие отбора, которое определяет взаимосвязь между столбцами. Вот инструкция SELECT для запроса, выполненного вручную.

Вывести список всех заказов, включая номер и стоимость заказа, а также имя клиента и лимит его кредита.

SELECT order_num, amount, company, credit_limit FROM orders, customers WHERE cust = cust_num

Приведенный запрос очень похож на запросы, рассмотренные ранее, однако между ними есть два отличия. Во-первых, предложение FROM содержит две таблицы, а не одну. Во-вторых, в условии отбора *cust=cust_num* сравниваются столб-

цы из двух различных таблиц. Мы будем называть эти столбцы *связанными.* Данное условие отбора, как и любое другое, уменьшает число строк в таблице результатов запроса. А поскольку запрос двухтабличный, условие отбора на самом деле уменьшает число *пар* строк в таблице результатов. Фактически в условии отбора заданы те же самые связанные столбцы, которые использовались при выполнении запроса с помощью карандаша и бумаги. Действительно, в этом условии очень хорошо отражена суть сопоставления столбцов, производимого вручную: «Включить, в таблицу результатов запроса только те пары строк, *у* которых идентификатор клиента (*cust*) в таблице *orders* равен идентификатору клиент (*cust_num*) в таблице *customers*».

Обратите внимание вот на что: в инструкции SELECT ничего не говорится о том, как должен выполняться запрос SQL. Там нет никаких упоминаний вроде «начните с заказов» или «начните с клиентов». Вместо этого в запросе сообщается, что должны представлять собой результаты запроса, а способ их получения оставлен на усмотрение СУБД.

В SQL не требуется, чтобы связанные столбцы были включены в результаты многотабличного запроса. На практике они чаще всего и не включаются, как это было в двух предыдущих примерах. Это связано с тем, что первичные и внешние ключи, как правило, представляют собой идентификаторы (такие, как идентификатор офиса или идентификатор служащего в приведенных примерах), которые человеку трудно запомнить, тогда как соответствующие названия (города, районы, имена, должности) запомнить гораздо легче. Поэтому вполне естественно, что в предложении WHERE для объединения двух таблиц используются идентификаторы, а в предложении SELECT для создания столбцов результатов запроса – более удобные для восприятия имена.

Условия для отбора строк

В многотабличном запросе можно комбинировать условие отбора, в котором задаются связанные столбцы, с другими условиями отбора, чтобы еще больше сузить результаты запроса. Предположим, что требуется повторить предыдущий запрос, но включить в него только офисы, имеющие большие плановые объемы продаж.

Вывести список офисов, в которых план продаж превышает $600000.

SELECT city, name, title FROM offices, salesreps WHERE mgr=empl_num AND target>600000

Вследствие применения дополнительного условия отбора число строк в таблице результатов запроса уменьшилось. Согласно первому условию (*mgr=empl_num*) из таблиц *offices* и *salesreps* отбираются пары строк, которые имеют соответствующее отношение предок/потомок; согласно второму условию производится дальнейший отбор только тех пар строк, где плановый объем продаж превышает $600000.

Несколько связанных столбцов

Таблицы *orders* и *products* в учебной базе данных связаны парой составных ключей. Столбцы *mfr* и *product* в таблице *orders* вместе образуют внешний ключ для таблицы *products* и связаны с ее столбцами *mfr_id* и *product_id* соответственно. Чтобы объединить таблицы на основе такого соотношения, необходимо задать обе пары связанных столбцов, как показано в данном примере.

Вывести список всех заказов, в том числе их стоимости и описания товаров.

SELECT order_num, amount, description FROM orders, products WHERE mfr = mfr_id AND product = product_id

Условие отбора в данном запросе показывает, что связанными парами строк таблиц *orders* и *products* являются те, в которых пары связанных столбцов содержат одни и те же зна-

чения. Объединения посредством нескольких столбцов распространены меньше, чем объединения посредством одного столбца и обычно встречаются в запросах с составными внешними ключами, как в приведенном выше примере.

Полные имена столбцов

В учебной базе данных имеется несколько случаев, когда две таблицы содержат столбцы с одинаковыми именами. Например, столбцы с именем *sales* имеются в таблицах *offices* и *salesreps*. В столбце *sales* таблицы *offices* содержится объем продаж на текущий день для каждого офиса. В аналогичном столбце таблицы *salesreps* содержится текущий объем продаж для каждого служащего. Обычно с этими двумя столбцами затруднений не возникает, поскольку в предложении FROM задается соответствующая таблица, как в следующих примерах.

Показать названия городов, в которых фактический объем продаж превышает плановый.

SELECT city, sales FROM offices WHERE sales > target

Показать имена служащих, у которых объемы продаж превышают $350000.

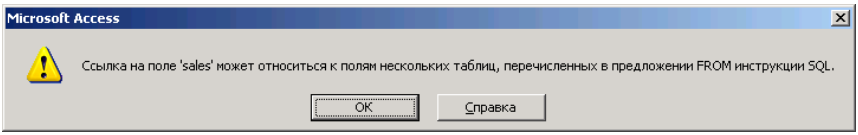
SELECT name, sales FROM salesreps WHERE sales > 350000

Однако вот запрос, в котором одинаковые имена столбцов представляют проблему.

Показать имя, офис и объем продаж каждого служащего.

SELECT name, sales, city FROM salesreps, offices WHERE rep_office=office

При попытке выполнить запрос появится сообщение.



В инструкции SELECT вместо простых имен столбцов всегда можно использовать полные имена. Таблица, заданная в полном имени столбца, должна, конечно, соответствовать од-

ной из таблиц, указанных в предложении FROM. Вот исправленный вариант предыдущего запроса.

Показать имя, офис и объем продаж каждого служащего.

SELECT name, salesreps. sales, city FROM salesreps, offices WHERE rep_office=office

Никогда не помешает использовать в многотабличных запросах полные имена столбцов. Недостатком, естественно, является то, что запрос становится длиннее. В интерактивном режиме можно сначала попробовать выполнить запрос с простыми именами столбцов, чтобы найти все сомнительные столбцы. Если выдается сообщение об ошибке, запрос редактируют и точно указывают столбцы с повторяющимися именами.

Псевдонимы таблиц

Псевдонимы таблиц используются в тех же случаях, что и полное имя таблицы, если это имя очень длинное и его употребление загромождает запрос.

Вот пример предыдущего запроса с использованием псевдонима таблиц.

Показать имя, офис и объем продаж каждого служащего.

SELECT S.name, S.sales, O.city FROM salesreps S, offices O WHERE S.rep_office=O.office

Выборка всех столбцов

Как уже говорилось, инструкция SELECT * используется для выборки всех столбцов таблицы, указанной в предложении FROM. В многотабличном запросе звездочка означает выбор всех столбцов из всех таблиц, указанных в предложении FROM.

Очевидно, что инструкция SELECT * становится менее практичной, когда в предложении FROM указаны две, три или более таблицы.

SQL трактует звездочку как особый вид универсального имени столбца, которое преобразуется в список всех столбцов. В

следующем запросе имя *salesreps*.* означает список имен всех столбцов таблицы *salesreps*:

SELECT salesreps., city, region FROM salesreps, offices WHERE rep_office=office*

Таблица результатов запроса будет иметь одиннадцать столбцов – девять столбцов таблицы *salesreps* и следом за ними два столбца таблицы *offices*, указанных явно. Такой тип «полных имен столбцов» поддерживается многими СУБД. Он был запрещен в стандарте SQL1, но является частью стандарта SQL2.

Задания

1. Вывести список всех служащих, включая города и регионы, в которых они работают.
2. Вывести список офисов, включая имена и должности их руководителей.
3. Сообщить всю информацию о служащих и офисах, в которых они работают.
4. Отобразить номер заказа и заказанный продукт.
5. Отобразить номер заказа и компанию-клиента.
6. В алфавитном порядке отобразить названия компаний-клиентов и имя менеджера, работающего с данной компанией.
7. Вывести список заказов за 1989 год с указанием названий компаний и стоимости заказа.
8. Отобразить список заказов за 1990 год с указанием названий компаний, наименования купленного продукта, количества единиц продукта, стоимости заказа.
9. Найти максимальный заказ, сделанный Orion Corp.
10. Найти минимальную продажу, сделанную Биллом Адамсом (Bill Adams).

Лабораторная работа № 9

Объединения (JOIN)

На рис. 9.1 в упрощенном виде изображена синтаксическая диаграмма предложения FROM в стандарте SQL2.

Рис. 9.1. Синтаксическая диаграмма предложения FROM

Стандартное внутреннее объединение таблиц *salespers* и *officies* на стандарте SQL1 можно выразить так.

SELECT salesreps.name, officies.city
FROM salesreps,officies
WHERE salesreps.rep_office = officies.office;

В стандарте SQL2 это по-прежнему допустимая инструкция. Создатели стандарта SQL2 не смогли отказаться от подобного синтаксиса, т.к. он применяется в миллионах работающих приложений. Тем не менее инструкцию можно также записать следующим образом.

SELECT name,city FROM salesreps
INNER JOIN officies ON salesreps.rep_office = officies.office;

Эти две таблицы объединяются явно посредством операции JOIN, а условие отбора, описывающее объединение, находится теперь в предложении ON внутри предложения FROM. В условии отбора могут быть заданы любые критерии сравнения строк двух объединяемых таблиц.

SELECT name, city
FROM salesreps
INNER JOIN officies
ON (salesreps.rep_office=officies.office) and (salesreps.Sales>officies.target)

В этих простых двухтабличных объединениях все содержимое предложения WHERE просто перешло в предложение ON, т.е. последнее не добавляет ничего нового в язык SQL.

Запрос на внутреннее объединение продавцов и номеров заказов со стоимостью выглядит следующим образом (31 запись).

SELECT name, order_num, amount FROM orders, salesreps WHERE rep=empl_num

Вот вариант того же запроса, определяющий левое объединение (31 запись).

SELECT name, order_num, amount
FROM salesreps LEFT JOIN orders ON orders.rep=salesreps.empl_num

В запрос войдут строки с полями из таблицы ORDERS, присоединяемыми к *name*.

Вот вариант того же запроса, определяющий правое объединение (32 записи).

SELECT name, order_num, amount
FROM salesreps RIGHT JOIN orders ON orders.rep=salesreps.empl_num;

В данном запросе уже *name* присоединяется к коду менеджера (*orders.rep*), а т.к. у менеджера с кодом 101 нет имени, то он в таблице появляется с пустым именем. При использовании LEFT JOIN он вообще не появляется.

Задания

1. Вывести список заказов стоимостью выше 25000, включая имя клиента, сделавшего заказ, и имя служащего, закрепленного за этим клиентом.

2. Вывести список заказов стоимостью выше 25000, включая имя клиента, сделавшего заказ, имя закрепленного за ним служащего и офис, в котором работает этот служащий.

3. Найти все заказы, полученные в тот день, когда на работу был принят новый служащий.

4. Получить список служащих и офисов, где плановый объем продаж служащего больше, чем план офиса.

Лабораторная работа № 10

Запросы с группировкой (предложение GROUP BY), условия отбора групп (предложение HAVING)

Запросы с группировкой (GROUP BY)

Для подведения итогов по запросу, как уже было отмечено ранее, используются статистические функции. Однако часто требуется отобразить и промежуточные результаты. Для этих целей служит предложение GROUP BY.

Какова средняя стоимость заказа?

SELECT AVG (amount) FROM orders

Какова средняя стоимость заказа для каждого служащего?

SELECT rep, AVG (amount) FROM orders GROUP BY rep

Первый запрос представляет собой простой итоговый запрос, аналогичный примерам, рассмотренным ранее. Второй запрос возвращает несколько итоговых строк – по одной строке для каждой группы. На рис 10.1 изображена схема выполнения второго запроса. На логическом уровне запрос выполняется следующим образом.

1. Заказы делятся на группы, по одной группе для каждого служащего. В каждой группе все заказы имеют одно и то же значение в столбце REP.

2. Для каждой группы вычисляется среднее значение столбца *amount* по всем строкам, входящим в группу, и генерируется одна итоговая строка результатов. Эта строка содержит значение столбца rep для группы и среднюю стоимость заказа для данной группы.

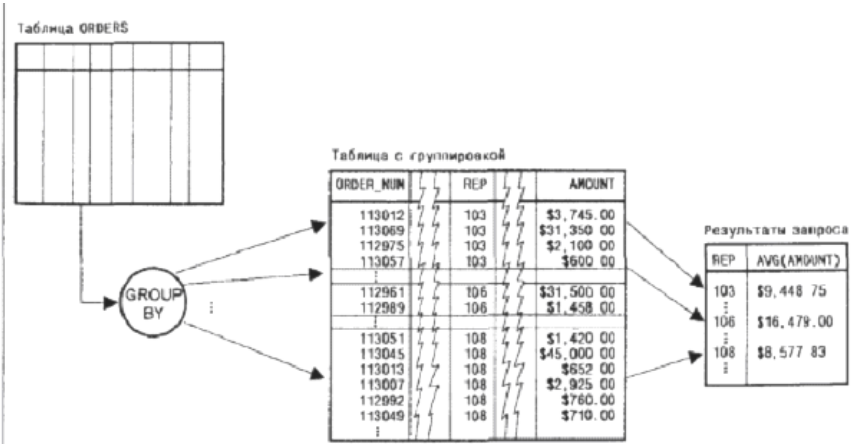


Рис. 10.1. Схема выполнения запроса с группировкой

Запрос, включающий в себя предложение GROUP BY, называется запросом с группировкой, поскольку он объединяет строки исходных таблиц в группы и для каждой группы строк генерирует одну строку в таблице результатов запроса. Столбцы, указанные в предложении GROUP BY, называются столбцами группировки, поскольку именно они определяют, по какому признаку строки делятся на группы.

Несколько столбцов группировки

SQL позволяет группировать результаты запроса на основании двух или более столбцов. Например, предположим, что вам требуется сгруппировать заказы по служащим и клиентам. Эту задачу выполняет приведенный ниже запрос.

Подсчитать общее количество заказов по каждому клиенту для каждого служащего.

SELECT rep, cust, sum(amount) FROM orders GROUP BY rep, cust

Даже при группировке по двум столбцам SQL обеспечивает только один уровень группировки. Приведенный запрос генерирует одну итоговую строку для каждой пары служащий/клиент. С помощью SQL нельзя создать группы и подгруппы с двумя уровнями итоговых результатов. Лучшее, что можно сделать – отсортировать данные таким образом, чтобы строки в таблице результатов шли в нужном порядке. Для этих целей служит предложение ORDER BY.

Подсчитать общее количество заказов по каждому клиенту для каждого служащего; отсортировать результаты запроса по клиентам и служащим.

SELECT cust, rep, SUM(amount) FROM orders GROUP BY cust, rep ORDER BY cust, rep

Условия отбора групп (предложение HAVING)

Точно так же, как предположение WHERE используется для отбора отдельных строк, участвующих в запросе, предложение HAVING можно применить для отбора групп строк. Его формат соответствует формату предложения WHERE. Предложение HAVING состоит из ключевого слова HAVING, за которым следует условие отбора. Таким образом, данное предложение определяет условие отбора для групп строк.

Следующий пример иллюстрирует роль предложения HAVING:

Какова средняя стоимость заказа для каждого служащего из числа тех, у которых общая стоимость заказов превышает $30000?

SELECT rep, AVG(amount) FROM orders GROUP BY rep HAVING SUM(amount)>30000

Вначале предложение GROUP BY разделяет заказы на группы по служащим. После этого предложение HAVING исключает все группы, в которых общая стоимость заказа не

превышает $30000. И, наконец, предложение SELECT вычисляет среднюю стоимость заказа для каждой из оставшихся групп, и генерирует таблицу результатов запроса (рис. 10.2).

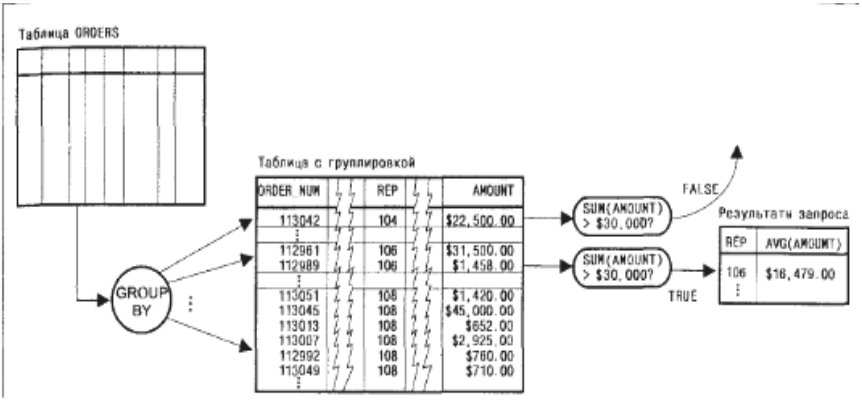


Рис. 10.2 Схема работы оператора GROUP BY

Для каждого офиса, в котором работают два или более человек, вычислить общий плановый и фактический объемы продаж для всех служащих офиса.

SELECT city, SUM(quota), SUM(salesreps.sales) FROM offices, salesreps WHERE offices=rep_office GROUP BY city HAVING COUNT (*)>=2

Задания

1. Какова максимальная сумма заказа для каждого клиента?
2. Какова сумма всех заказов для каждого клиента и насколько она превышает лимит для данной компании?
3. Каков максимум продаж для каждого офиса?
4. Сколько служащих работает в каждом офисе?
5. Подсчитать общее количество заказов для каждого служащего.
6. Показать цену, количество на складе и общее количество заказанных единиц для каждого наименования товара, если

для него общее количество заказанных единиц превышает 75 % от количества товара на складе.

7. Найти максимальную сумму продаж за каждый месяц 1989 года.

8. Найти компанию, сделавшую наибольшее количество заказов.

9. Найти служащего, принявшего наименьшее количество заказов (и уволить).

10. Найти служащего, принявшего заказы на наименьшую сумму (и уволить).

Лабораторная работа № 11

Подчиненные запросы

В SQL существует понятие подчиненного запроса. Механизм подчиненных запросов позволяет использовать результаты одного запроса в качестве составной части другого. Возможность применения одного запроса внутри другого и была причиной появления слова «структурированный» в названии «структурированный язык запросов». Понятие подчиненного запроса не так широко известно, как понятие объединения, но оно играет важную роль в SQL по трем следующим причинам:

– инструкция SQL с подчиненным запросом зачастую является самым естественным способом выражения запроса, так как она лучше всего соответствует словесному описанию запроса;

– подчиненные запросы облегчают написание инструкции SELECT, поскольку они позволяют разбивать запрос на части (на запрос и подчиненные запросы), а затем складывать эти части вместе;

– существуют запросы, которые нельзя сформулировать на SQL, не прибегая к помощи подчиненных запросов.

Подчиненным называется запрос, содержащийся в предложении WHERE или HAVING другой инструкции SQL. Подчиненные запросы позволяют естественным образом обраба-

тывать запросы, выраженные через результаты других запросов. Вот пример такого запроса.

Вывести список офисов, для которых плановый объем продаж превышает сумму плановых объемов продаж всех служащих.

В данном запросе требуется получить список офисов из таблицы *offices*, для которых значение столбца *target* удовлетворяет некоторому условию. Логично предположить, что инструкция *SELECT*, выражающая данный запрос, должна выглядеть следующим образом.

SELECT city FROM offices WHERE target>???

Здесь величина ??? равна сумме плановых объемов продаж всех служащих, работающих в данном офисе. Как можно задать ее в этом запросе? Вам уже известно, что сумму плановых объемов продаж для отдельного офиса (скажем, офиса с идентификатором 21) можно получить с помощью следующего запроса.

SELECT SUM(quota) FROM salesreps WHERE rep_office=21

Но как результаты этого запроса вставить в предыдущий запрос вместо вопросительных знаков? По-видимому, было бы разумно вначале написать первый запрос, а затем заменить ??? вторым запросом.

SELECT city FROM offices
WHERE target > (SELECT SUM(quota) FROM salesreps
WHERE rep_office=office)

Фактически это и есть правильный SQL-запрос. Подчиненный (внутренний) запрос вычисляет для каждого офиса сумму плановых объемов продаж всех служащих, работающих в данном офисе. Главный (внешний) запрос сравнивает план продаж офиса с полученной суммой и, в зависимости от результата сравнения, либо добавляет данный офис в таблицу результатов запроса, либо нет. Совокупно главный и подчиненный запросы выражают исходный запрос и извлекают из базы данных требуемую информацию.

Подчиненные SQL-запросы всегда выступают в качестве части предложения WHERE или HAVING. В предложении

WHERE они помогают отбирать из таблицы результатов запроса отдельные строки, а в предложении HAVING – группы строк.

На рис. 11.1 изображена структура подчиненного SQL-запроса. Подчиненный запрос всегда заключается в круглые скобки, но по-прежнему сохраняет знакомую структуру инструкции SELECT, содержащей предложение FROM и необязательные предложения WHERE, GROUP BY и HAVING, Структура этих предложений в подчиненном запросе идентична их структуре в инструкции SELECT: в подчиненном запросе эти предложения выполняют свои обычные функции.

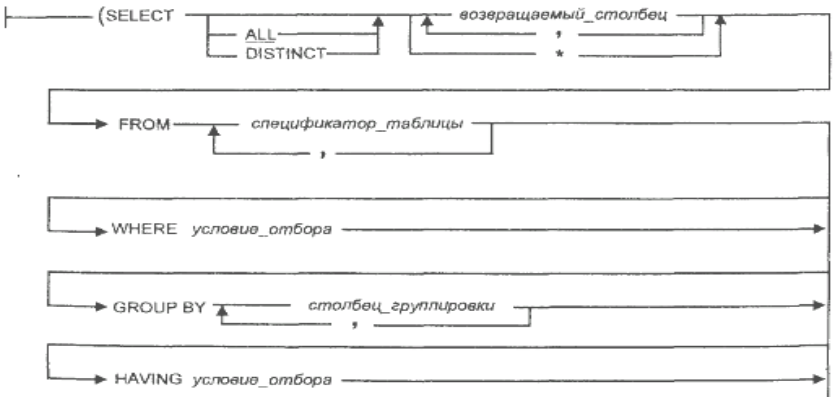


Рис. 11.1. Синтаксическая диаграмма подчиненного запроса

Между подчиненным запросом и инструкцией SELECT имеется ряд отличий.

1. Таблица результатов подчиненного запроса всегда состоит из одного столбца. Это означает, что в предложении SELECT подчиненного запроса всегда указывается один возвращаемый столбец.

2. В подчиненный запрос не может входить предложение ORDER BY. Результаты подчиненного запроса используются только внутри главного запроса и для пользователя остаются невидимыми, поэтому нет смысла их сортировать.

3. Имена столбцов в подчиненном запросе могут являться ссылками на столбцы таблиц главного запроса. Это внешние ссылки (подробно рассматриваются позже).

4. Подчиненный запрос не может быть запросом на объединение (UNION) нескольких различных инструкций SELECT; допускается использование только одной инструкции SELECT. (Стандарт SQL2 ослабляет это ограничение, позволяя, как будет показано ниже, создавать гораздо более мощные запросы).

Подчиненные запросы в предложении WHERE

Чаще всего подчиненные запросы указываются в предложении WHERE инструкции SQL. Когда подчиненный запрос содержится в данном предложении; он участвует в процессе отбора строк. В простейшем случае подчиненный запрос является частью условия отбора и возвращает значение, позволяющее проверить истинность или ложность условия. Рассмотрим пример.

Вывести список служащих, чей плановый объем продаж составляет менее 10 % от планового объема продаж всей компании.

SELECT name FROM salesreps WHERE quota <

*< (0.1 * (SELECT SUM(target) FROM offices).)*

В данном случае подчиненный запрос вычисляет сумму плановых объемов продаж всех офисов, которая затем умножается на 0,1 (10 %). Полученное значение используется в условии отбора при сканировании таблицы *salesreps* на предмет поиска нужных строк. В показанном примере подчиненный запрос возвращает для каждой строки одно и то же значение. По сути, вы самостоятельно могли бы произвести соответствующие вычисления, подсчитать план компании ($275000) и упростить условие отбора.

WHERE QUOTA < 275000.

Внешние ссылки

Часто в теле подчиненного запроса требуется ссылаться на значение столбца в текущей строке главного запроса. Рассмотрим запрос.

Вывести список офисов, для которых плановый объем продаж превышает сумму плановых объемов продаж всех служащих.

SELECT city FROM offices WHERE target > (SELECT SUM(quota) FROM salesreps WHERE rep_office=office).

В приведенной инструкции подчиненный запрос играет следующую роль: он вычисляет сумму плановых объемов продаж для служащих, работающих в одном офисе, а именно в том, который в данный момент времени проверяется предложением WHERE главного запроса. Подчиненный запрос выполняет просмотр таблицы *salesreps*. Однако столбец *offices* в предложении WHERE подчиненного запроса является столбцом не таблицы *salesreps*, а таблицы *offices*, которая входит в главный запрос. Во время последовательной проверки строк таблицы *offices* значение столбца *office* в текущей строке этой таблицы используется для выполнения подчиненного запроса.

Столбец *office* в подчиненном запросе является примером внешней ссылки. Внешняя ссылка представляет собой имя столбца, не входящего ни в одну из таблиц, перечисленных в предложении FROM подчиненного запроса, и принадлежащего таблице, указанной в предложении FROM главного запроса. Как показывает предыдущий пример, значение в столбце внешней ссылки берется из строки, проверяемой в настоящий момент главным запросом.

Операции сравнения

В операции сравнения с результатом подчиненного запроса можно использовать шесть операторов сравнения (=, <>, <, <=, >, >=), что и при простом сравнении. Подчиненный запрос, участвующий в операции сравнения, должен возвращать в качестве результата единичное значение, т.е. *одну строку, содержащую один столбец*.

Если в результате выполнения подчиненного запроса не будет получено ни одной строки или будет получено значение NULL, то операция сравнения возвращает NULL.

Проверка на принадлежность результатам подчиненного запроса (предикат IN)

Проверка на принадлежность результатам подчиненного запроса (предикат IN) является видоизменяемой формой простой проверки на членство во множестве. Одно значение сравнивается со столбцом данных, которые возвращаются подчиненным запросом, и если это значение равно одному из элементов столбца, проверка дает TRUE. Данная проверка используется, когда необходимо сравнить значение из проверяемой строки с множеством значений, отобранных подчиненным запросом.

Вывести список служащих тех офисов, где фактический объем продаж превышает плановый.

SELECT name FROM salesreps WHERE rep_office IN (SELECT office FROM offices WHERE sales > target).

Подчиненный запрос возвращает список идентификаторов офисов, где фактический объем продаж превышает плановый (офисы с номерами 11, 13 и 21). Главный запрос затем проверяет каждую строку таблицы *salesreps*.

Задания

1. Вывести список офисов, для которых плановый объем продаж превышает сумму плановых объемов продаж всех служащих.
2. Вывести список служащих, у которых плановый объем продаж равен или больше планового объема продаж офиса, расположенного в Атланте.
3. Вывести список клиентов, с которыми работает Билл Адамс.
4. Вывести список имеющихся в наличии товаров от компании ACI, количество которых превышает количество товаров ACI-41004.
5. Вывести список всех клиентов, заказавших изделия компании ACI (идентификатор изготовителя – ACI) в период между январем и июнем 1990 года.

6. Вывести список служащих, чей объем продаж превышает средний объем продаж по всей компании.

7. Вывести максимальный заказ и список компаний, заказывающих товары, хоть раз более чем на $1000000.

8. Вывести максимальный заказ и список компаний, хоть один заказ которых превышает средний.

Лабораторная работа № 12

Проверка на существование (EXIST). Использование ANY (SOME), ALL

С помощью проверки на существование (предикат EXISTS) можно выяснить, содержится ли в таблице результатов подчиненного запроса хотя бы одна строка. Аналогичной простой проверки не существует. Проверка на существование допустима только в подчиненных запросах.

Вот пример запроса, который можно легко сформулировать, используя проверку на существование.

Вывести список товаров, на которые получен заказ стоимостью $25000 или больше.

Теперь перефразируем этот запрос таким образом.

Вывести список товаров, для которых в таблице *orders* существует, по крайней мере, один заказ, удовлетворяющий условиям: а) является заказом на данный товар; б) имеет стоимость не менее чем $25000.

Инструкция SELECT, используемая для получения требуемого списка товаров, приведена ниже.

SELECT description FROM products WHERE EXIST (SELECT order_num FROM orders WHERE product=product_id AND mfr=mfr_id AND amount>=25000).

Главный запрос последовательно перебирает все строки таблицы *products*, и для каждого товара выполняется подчиненный запрос. Результатом подчиненного запроса является столбец данных, содержащий номера всех заказов текущего товара на сумму не меньше, чем $25000. Если такие заказы

есть (т.е. столбец не пустой), то проверка EXISTS возвращает TRUE. Если подчиненный запрос не дает ни одной строки заказов, проверка EXISTS возвращает значение FALSE. Эта проверка не может возвращать NULL.

Можно изменить логику проверки EXISTS и использовать форму NOT EXISTS. Тогда в случае, если подчиненный запрос не создает ни одной строки результата, проверка возвращает TRUE, в противном случае – FALSE.

Предикат EXISTS в действительности вовсе не использует результаты подчиненного запроса. Проверяется только наличие результатов. По этой причине в SQL смягчается правило, согласно которому подчиненный запрос должен возвращать один столбец данных, и в подчиненном запросе проверки EXIST допускается использование формы SELECT *. Поэтому предыдущий запрос можно переписать следующим образом.

Вынести список товаров, на которые получен заказ стоимостью $25000 или больше.

*SELECT description FROM products WHERE EXIST (SELECT * FROM orders WHERE products = product_id AND mfr=mfr_id and amount >=25000)*

На практике при использовании подчиненного запроса в проверке EXISTS всегда применяется форма SELECT *.

Многократное сравнение (предикаты ANY (SOME) и ALL)

В проверке IN выясняется, не равно ли некоторое значение одному из значений, содержащихся в столбце результатов подчиненного запроса. В SQL имеются также две разновидности *многократного сравнения* – ANY и ALL, расширяющие предыдущую проверку до уровня других операторов сравнения таких, как больше (>) или меньше (<). В обеих проверках некоторое значение сравнивается со столбцом данных, отобранных подчиненным запросом.

*Предикат ANY **

В проверке ANY для того, чтобы сравнить проверяемое значение со столбцом данных, отобранных подчиненным запросом, используется один из шести операторов сравнения (=, <>, <, >, <=, >=). Проверяемое значение поочередно сравнивается с каждым элементом, содержащимся в столбце. Если любое из этих сравнений дает результат TRUE, то проверка ANY возвращает значение TRUE.

Вывести список служащих, принявших заказ на сумму большую, чем десять процентов от их плана.

SELECT name FROM salesreps
*WHERE (.1 *quota < ANY (SELECT amount FROM orders WHERE rep=empl_num))*

Главный запрос по очереди проверяет все строки таблицы SALESREPS. Подчиненный запрос находит все заказы, принятые текущим служащим, и возвращает столбец, содержащий стоимости эти заказов. Предложение WHERE главного запроса вычисляет десять процентов от плана текущего служащего и использует это число в качестве проверяемого значения, сравнивая его со стоимостью каждого заказа, отобранного подчиненным запросом. Если есть хотя бы один заказ, стоимость которого превышает вычисленное проверяемое значение, то проверка < ANY возвращает значение TRUE, а имя служащего заносится в таблицу результатов запроса. Если таких заказов нет, имя служащего в таблицу результатов запроса не попадает. В соответствии со стандартом ANSI/ISO вместо предиката ANY можно использовать предикат SOME. Обычно можно употреблять любой из них, но некоторые СУБД не поддерживают предикат SOME.

На практике проверка ANY иногда может приводить к ошибкам, которые трудно выявить, особенно когда применяется оператор сравнения «не равно» (<>). Вот пример, иллюстрирующий данную проблему.

Вывести имена и данные о возрасте всех служащих, которые не руководят офисами.

SELECT name, age FROM salesreps WHERE empl_num <> ANY (SELECT mgr FROM offices)

Подчиненный запрос *SELECT MGR FROM OFFICES* в качестве результатов возвращает идентификаторы служащих, являющихся руководителями офисов, поэтому *кажется*, что запрос имеет следующий смысл: «Найти всех служащих, не являющихся руководителями офисов».

На самом же деле: «Найти всех служащих, которые для некоторого офиса не является руководителями этого офиса».

Правильным является следующий запрос.

SELECT name, age FROM salesreps WHERE NOT (empl_num = = ANY (SELECT mgr FROM offices))

Запрос с предикатом ANY всегда можно преобразовать в запрос с предикатом EXISTS, перенося операцию сравнения внутрь условия отбора подчиненного запроса. Обычно так и следует поступать, поскольку в этом случае исключаются ошибки, подобные описанной выше. Вот альтернативная форма запроса с проверкой EXISTS.

*SELECT name, age FROM salesreps WHERE NOT EXISTS (SELECT * FROM offices WHERE empl_num = mgr)*

*Предикат ALL **

Проверка ALL, также как и проверка ANY, используется для сравнения проверяемого значения со столбцом данных, отобранных подчиненным запросом. Проверяемое значение поочередно сравнивается с каждым элементом, содержащимся в столбце. Если все сравнения дают результат TRUE, то проверка ALL возвращает значение TRUE.

Вывести список тех офисов с их плановыми объемами продаж для всех служащих, у которых фактический объем продаж превышает 50 % от плана офиса.

*SELECT city, target FROM offices WHERE (0.50 * target) <ALL (SELECT sales FROM salesreps WHERE rep_office=office).*

Главный запрос поочередно проверяет каждую строку таблицы *offices*. Подчиненный запрос находит всех служащих, работающих в текущем офисе, и возвращает столбец с фактическими объемами продаж для каждого служащего. Предложение WHERE главного запроса вычисляет 50 процентов от плана продаж офиса и сравнивает это значение со всеми объемами продаж, получаемых в результате выполнения подчиненного запроса. Если все объемы продаж превышают проверяемое значение, то проверка <ALL возвращает значение TRUE, и данный офис включается в таблицу результатов запроса. Если нет, то офис не попадает в таблицу результатов.

Проверку WHERE X < ALL (SELECT Y…) следует понимать как «где X меньше чем все выбранные Y».

Задания

1. Вывести список клиентов, закрепленных за Сью Смит (Sue Smith), которые не сделали заказы на сумму более 3000.

2. Вывести список офисов, где имеется служащий, чей план превышает 55 процентов от плана офиса.

3. Вывести список компаний, средний заказ которых превышает общий средний заказ.

Лабораторная работа № 13

Добавление новых данных (инструкция INSERT)

Добавление новой строки в реляционную базу данных происходит тогда, когда во внешнем мире появляется новый объект, представляемый этой строкой. В рассматриваемой базе данных это выглядит следующим образом:

– если вы принимаете на работу нового служащего, в таблицу *salesreps* необходимо добавить новую строку с данными о нем;

– если служащий заключает договор с новым клиентом, в таблицу *customers* должна быть добавлена новая строка, представляющая этого клиента;

– если клиент делает заказ, в таблицу *orders* требуется добавить новую строку, содержащую информацию об этом заказе.

Во всех приведенных примерах новая строка добавляется для того, чтобы база данных оставалась точной моделью реального мира. Наименьшей единицей информации, которую можно добавить в реляционную базу данных, является одна строка.

В реляционной СУБД существует три способа добавления новых строк в базу данных.

Однострочная инструкция INSERT позволяет добавить в таблицу одну новую строку. Она широко используется в повседневных приложениях, например в программах ввода данных.

Многострочная инструкция INSERT обеспечивает извлечение строк из одной части базы данных и добавление их в другую таблицу. Она обычно используется в конце месяца или года, когда старые строки таблицы пересылаются в неиспользуемую таблицу для сохранения.

Однострочная инструкция INSERT

Однострочная инструкция INSERT, синтаксическая диаграмма которой представлена на рис. 13.1, добавляет в таблицу новую строку. В предложении *INTO* указывается таблица (целевая таблица), в которую добавляется новая строка, а в предложении VALUES содержатся значения данных для новой строки. Список столбцов определяет, какие значения в какой столбец заносятся.

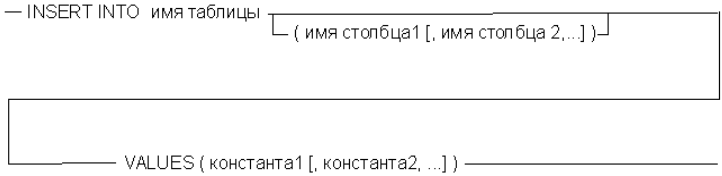


Рис. 13.1. Схема работы инструкции INSERT

Предположим, что вы только что приняли на работу нового служащего Генри Якобсена (Henry Jacobsen) со следующими данными.

Имя	Генри Якобсен
Возраст	36
Идентификатор	111
Должность	Менеджер по продажам
Офис	Атланта (идентификатор офиса 13)
Дата приема	25 июля 1990 года
Личный план	Еще не установлен
Объем продаж на текущую дату	$0.00
Руководитель офиса	105

Ниже приведена инструкция INSERT, которая добавляет информацию о служащем Якобсене в базу данных.

Добавить информацию о новом служащем Генри Якобсене

INSERT INTO salesreps (name, age, empl_name, sales, title, hire_date, rep_office, manager)

VALUES ('henry jacobsen', 36, 111, 0, 'sales mgr', '25/10/1990', 13, 105).

На рис. 13.2 представлена графическая схема выполнения инструкции INSERT. Вначале инструкция создает новую строку, структура которой повторяет структуру столбцов таблицы, а затем заполняет ее значениями из предложения VALUES. После чего добавляет эту строку в таблицу. Т.к. строки в таблице не упорядочены, то нет никаких указаний о том, где вставлять строку «вверху», «внизу» или «между двух строк» таблицы. Эта строка будет входить в результаты последующих запросов на выборку таблицы *salesreps*, но в таблице результатов запроса она может находиться в любом месте.

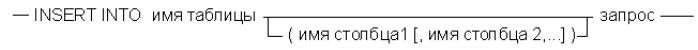


Рис. 13.2. Синтаксическая диаграмма многострочной инструкции INSERT

Предположим, что служащий Якобсен получает свой первый заказ от компании InterCorp, нового клиента, которому присвоен идентификатор 2126. Это заказ на 20 изделий ACI-41004 общей стоимостью $2340, и ему присваивается номер 113069. Ниже приведены инструкции INSERT, добавляющие в базу данных информацию о новом клиенте и заказе.

Добавить для служащего Якобсена информацию о новом клиенте и заказе.

INSERT INTO customers (company, cust_num, credit_limit, cust_rep) VALUES (‘InterCorp', 2126, 15000.00, 111)

INSERT INTO orders (amount, mfr, product, qty, order_date, order_num, cust, rep) VALUES (2340.00, 'ACI', '41004', 20,#25/07/1990#, 113070, 2126, 111)

Как показывает приведенный пример, если в таблице много столбцов, то инструкция INSERT может оказаться довольно длинной, однако ее структура по-прежнему останется очень простой.

Для добавления текущей даты используется системная константа *CURRENT DATE*, что обеспечивает ввод текущей даты в качестве даты получения заказа. Эта системная константа определена в стандарте SQL2 и поддерживается многими ведущими СУБД. В остальных СУБД для получения текущих даты и времени используются другие системные константы или встроенные функции. В Microsoft Access используется выражение *Date()*.

Предыдущий запрос с добавлением текущей даты.

INSERT INTO orders (amount, mfr, product, qty, order_date, order_num, cust, rep) VALUES (2340.00, 'ACI', '41004', 20, Date(), 113070, 2126, 111)

Добавление всех столбцов

Для удобства в SQL разрешается не включать список столбцов в инструкцию INSERT. Если список столбцов опущен, он генерируется автоматически и в нем слева направо перечисляются все столбцы таблицы. При выполнении ин-

струкции SELECT * генерируется такой же список столбцов. Пользуясь этой сокращенной формой записи, добавим еще одного сотрудника.

INSERT INTO salesreps VALUES (112, 'woodrow wilson', 36, 13, 'sales rep', #07/25/1990#, 105, null, null).

Как видно из данного примера, если список столбцов опущен, то в списке значений необходимо явно указывать значения NULL. Кроме того, последовательность значений данных должна в точности соответствовать порядку столбцов в таблице.

В интерактивном режиме удобно не включать в инструкцию INSERT список столбцов, так как это уменьшает длину инструкции. В случае программного использования SQL список столбцов должен быть задан всегда, поскольку такую программу легче читать и понимать.

Многострочная инструкция INSERT

Многострочная инструкция INSERT, синтаксическая диаграмма которой изображена на рис. 13.2, добавляет в целевую таблицу несколько строк (более одной). В этой разновидности инструкции INSERT значения для новых строк явно не задаются. Источником новых строк служит запрос на выборку, содержащийся внутри инструкции INSERT.

Процедура добавления строк со значениями, взятыми из той же базы данных, может изначально показаться странной, но иногда оказывается необходимой. Предположим, что нам требуется скопировать номера, даты и стоимости всех заказов, сделанных до 1 января 1990 года, из таблицы *orders* в другую таблицу с именем *oldorders*. Многострочная инструкция INSERT позволяет скопировать данные компактно и быстро.

Создайте таблицу *oldorders.*

Имя поля	Тип данных
order_num	числовой
order_date	дата/время
amount	числовой

Скопировать старые заказы, сделанные до 1 января 1990 года в таблицу *oldorders*.

INSERT INTO oldorders (order_num, order_date, amount) SELECT order_num, order_date, amount FROM orders WHERE order_date <#01/01/1990#

Хотя многострочная инструкция INSERT выглядит сложнее однострочной, в действительности она является очень простой. В ней, как и в однострочной инструкции INSERT, задаются таблица и столбцы, в которые заносятся новые элементы данных. Оставшаяся часть инструкции представляет собой запрос, извлекающий данные из таблицы *orders*.

Вот еще одна ситуация, когда можно использовать многострочную инструкцию INSERT. Предположим, требуется проанализировать, что именно приобретают клиенты, и для этого необходимо просмотреть информацию о клиентах и служащих, имеющих большие заказы (стоимостью свыше $15000). Запросы, которые необходимо для этого выполнить, объединяют информацию из таблиц *customers*, *salesreps* и *orders*. В рассматриваемой базе данных эти трехтабличные запросы будут выполняться довольно быстро, но в реальной корпоративной базе данных, содержащей тысячи строк информации, выполнение таких запросов заняло бы длительное время.

Вместо выполнения нескольких длинных трехтабличных запросов лучше создать для требуемых данных новую таблицу с именем *bigorders*, имеющую следующую структуру.

Столбец	Информация	Тип столбца
amount	Стоимость заказа (из таблицы orders)	числовой
company	Имя клиента (из таблицы customers)	текстовый
name	Имя служащего (из таблицы salesreps)	текстовый
perf	Перевыполнение/недовыполнение плана (вычисляется по таблице salesreps)	числовой
mfr	Идентификатор производителя (из таблицы orders)	текстовый
product	Идентификатор товара (из таблицы orders)	текстовый
qty	Заказанное количество (из таблицы orders)	числовой

После создания таблицы *bigorders* ее можно заполнить данными с помощью следующей инструкции *INSERT*.

Загрузить в таблицу *bigorders* данные для анализа.

INSERT INTO bigorders (amount, company, name, perf, product, mfr, qty) SELECT amount, company, name, (sales - quota), product, mfr, qty from orders, customers, salesreps WHERE cust = cust_num AND rep = empl_num AND amount > 15000.00

В больших базах данных выполнение такой инструкции INSERT займет некоторое время, поскольку она содержит запрос к трем таблицам. После того, как выполнение инструкции завершится, в таблице в *bigorders* будет содержаться копия данных из других таблиц. Кроме того, таблица *bigorders* не будет автоматически изменяться при добавлении в базу данных новых заказов, поэтому данные в ней могут быстро устареть. Оба этих фактора выглядят как недостаток. Однако последующие запросы на выборку к таблице *bigorders* будут представлять собой запросы к одной таблице. Следует также отметить, что каждый из этих запросов будет выполняться намного быстрее, чем при использовании объединения. Поэтому копирование данных можно назвать хорошим методом проведения анализа, особенно если исходные таблицы являются большими.

На запрос, содержащийся внутри многострочной инструкции INSERT, стандарт SQL1 накладывает несколько логических ограничений:

- в запрос нельзя включать предложение ORDER BY. Не имеет смысла сортировать таблицу результатов запроса, поскольку она добавляется в таблицу, которая, как и все остальные, не упорядочена;
- таблица результатов запроса должна содержать количество столбцов, равное длине списка столбцов в инструкции INSERT (или полностью всю целевую таблицу, если список столбцов опущен), а типы данных соответствующих столбцов таблицы результатов запроса и целевой таблицы должны быть совместимыми;

• запрос не может быть запросом на объединение (UNION) нескольких различных инструкций SELECT;

• имя целевой таблицы инструкции INSERT не может присутствовать в предложении FROM запроса на выборку или любого запроса, вложенного в него. Тем самым запрещается добавление таблицы самой к себе.

Первые два ограничения очевидны, а последние два были добавлены только для того, чтобы избежать излишней сложности инструкции INSERT. Как результат, в стандарте SQL2 они были ослаблены, и в запросе допускаются операции UNION и JOIN; разрешается также «самодобавление».

Задания

1. Создать в городе Вашингтон новый офис: с номером 14, в восточном регионе, с идентификационным номером руководителя 113, с планом продаж $23000000 и фактическими продажами $18000000.
2. Набрать сотрудников:

а) в Вашингтон: идентификационный номер – 113, возраст – 60, номер офиса – 14, должность – Sales Mgr, дата приема на работу – 21.05.1987, руководитель офиса 113, план продаж – 40000000, объем продаж – 54000000;

б) в Джефферсон: 114, 54, 14, Sales Rep, 23.06.1988, 133, 5400000, 40000000;

в) в Линкольн: 115, 43, 14, Sales Reps, 28.02.1988, 133, 34000000, 23500000;

г) в Гамильтон: 116, 37, 14, Sales Reps, 11.04.1990, 133, 12000000, 20000000.

3. Добавить пару заказов в таблицу orders.
4. Добавить пару продуктов в таблицу products.
5. Добавить компанию-клиента «БНТУ».

Лабораторная работа № 14

Удаление данных (инструкция DELETE)

Удалять ту или иную строку из базы данных приходится тогда, когда объект, представляемый этой строкой, исчезает из «внешнего мира». Например: при увольнении служащего из компании необходимо удалить соответствующую строку из таблицы *salesreps*; при отмене заказа клиентом удаляется соответствующая строка в таблице *orders*; и т.д.

Наименьшей единицей информации, которую можно удалить из реляционной базы данных, является одна строка.

Инструкция DELETE удаляет выбранные строки из одной таблицы. В предложении FROM указывается таблица, содержащая строки, которые требуется удалить. В предложении WHERE указывается критерий отбора строк, которые должны быть удалены.

Предположим, что недавно принятый на работу Генри Якобсен решил уволиться. Следующая инструкция DELETE удаляет относящуюся к данному служащему информацию.

DELETE FROM salesreps WHERE name = 'Henry Jacobsen'

Аналогично инструкции SELECT инструкция DELETE с помощью предложения WHERE позволяет удалять как одну строку (как в приведенном выше примере), так и набор строк.

Предположим, что клиент InterCorp (идентификатор 2126) отменил все свои заказы. Следующая инструкция DELETE удаляет заказы из таблицы *orders*.

DELETE FROM orders WHERE cust=2126.

В данном случае предложение WHERE выбирает несколько строк таблицы *orders*, которые затем удаляются из нее. Все строки таблицы проверяются на соответствие условиям отбора, и строки, соответствующие условиям, удаляются.

Для удаления всех строк таблицы предложение WHERE опускается.

Следующий запрос удаляет все заказы.

DELETE FROM orders

Из таблицы *orders* удаляются все данные. Она становится пустой, но по-прежнему остается в базе данных. При необходимости таблица удаляется с помощью инструкции DROP TABLE.

Иногда удаление строк необходимо проводить, используя данные из других строк. Допустим необходимо удалить все заказы, принятые Сью Смит. Для этого необходимо знать ее идентификатор, и, соответственно, произвести запрос к двум таблицам. Вот запрос, отображающий заказы, принятые Сью Смит.

SELECT order_num, amount FROM orders, salesreps WHERE rep=empl_num AND name='Sue Smith'

Однако в инструкции DELETE запрещено использовать объединение таблиц. Инструкция DELETE с параллельным удалением из двух таблиц является неправильной.

DELETE FROM orders, salesreps WHERE rep=empl_num AND name='Sue Smith'

Чтобы выполнить поставленную задачу, необходимо использовать условие отбора с подчиненным запросом.

Следующая инструкция удаляет все заказы, принятые Сью Смит.

DELETE FROM orders WHERE rep=(SELECT empl_num FROM salesreps WHERE name='Sue Smith')

Удалить данные обо всех клиентах, обслуживаемых работниками, у которых фактический объем продаж меньше, чем 80 % от плана.

*DELETE FROM customers WHERE cust_rep IN (SELECT empl_num FROM salesreps WHERE sales < (0.8*quota))*

Подчиненные запросы в инструкции DELETE могут содержать несколько уровней вложенности, а также содержать внешние ссылки на целевую таблицу инструкции DELETE.

Удалить данные о всех клиентах, которые не делали заказов с 10 ноября 1989 года.

*DELETE FROM customers WHERE NOT EXISTS (SELECT * FROM orders WHERE cust=cust_num AND order_date > #10.11.1989#*

Задания

1. Удалить все заказы, сделанные до 15.11.1989.
2. Удалить данные обо всех клиентах, обслуживаемых Биллом Адамсом, Мэри Джонс и Дэном Робертсом (идентификаторы служащих 105, 109, 101).
3. Удалить данные обо всех служащих, принятых на работу до июля 1988 и еще не имеющих личного плана.
4. Удалить данные обо всех служащих, у которых сумма текущих заказов меньше, чем два процента их личного плана.
5. Удалить Вашингтон.
6. Удалить Линкольна, Джефферсона и Гамильтона.

Лабораторная работа № 15

Обновление существующих данных (инструкция UPDATE)

Часто при работе с базами данных достаточно произвести обновление информации, хранящейся в таблице, без добавления или удаления строк. Для учебной базы это происходит, например, в следующих случаях:

– если клиент изменяет количество заказанного товара, в столбце QTY таблицы ORDERS должна быть обновлена соответствующая строка;

– если руководитель переходит из одного офиса в другой, столбец MGR таблицы *offices* и столбец *rep_office* таблицы *salesreps* необходимо обновить, чтобы отобразить новое назначение;

– если личные планы продаж в нью-йоркском офисе увеличиваются на пять процентов, значения столбца QUOTA в соответствующих строках таблицы *salesreps* должны быть обновлены.

Наименьшей единицей информации, которую можно обновить является значение одного столбца в одной строке.

Для обновления одного или нескольких столбцов в выбранных строках таблицы используется инструкция UPDATE.

Увеличить лимит кредита для компании Acme Manufacturing до $60000 и закрепить ее за Мэри Джонс (идентификатор 109).

UPDATE customers SET credit_limit = 60000, cust_rep = 109 WHERE company = 'Acme Mfg'

В инструкции указывается целевая таблица (*customers*), которая должна быть модифицирована (UPDATE), задаются условия отбора строк для обновления (*WHERE company = 'Acme Mfg'*) и указывается, какие столбцы должны быть обновлены (*credit_limit = 60000, cust_rep = 109*).

Инструкция UPDATE во многом аналогична инструкции DELETE и может обновить одновременно несколько строк.

Перевести всех служащих из чикагского офиса (идентификатор 12) в нью-йоркский офис (идентификатор 11) и понизить их личные планы на десять процентов.

*UPDATE salesreps SET rep_office = 11, quota = 0.9*quota WHERE rep_office =12.*

Если в инструкции SELECT отсутствует предложение WHERE, то обновляются все строки целевой таблицы.

Увеличить все личные планы на пять процентов.

*UPDATE salesreps SET quota = 1.05*quota*

Использование подчиненных запросов в инструкции UPDATE

В инструкции UPDATE, так же как и в инструкции DELETE, подчиненные запросы дают возможность отбирать строки для обновления, используя данные из других таблиц.

Увеличить на $5000 лимиткредита для тех клиентов, которые сделали заказ на сумму более $25000.

UPDATE customers SET credit_limit + 5000 WHERE cust_num IN (SELECT DISTINCT cust FROM orders WHERE amount>25000)

Всех служащих, обслуживающих более трех клиентов, подчинить непосредственно Сэму Кларку (идентификатор 106).

UPDATE salesreps SET manager=106 WHERE 3 < (SELECT COUNT(*) FROM customers WHERE cust_rep=empl_name)

Задания

1. Переименовать Marry Jones в Marry Adams.
2. Переименовать Jones Mfg в Adams Mfg и увеличить лимит кредита на 20 %.
3. Закрепить за Биллом Адамсом клиентов, обслуживаемых работниками, чей объем продаж меньше, чем 80 % их личного плана.
4. Перевести всех клиентов, обслуживаемых работниками с идентификатором 105, 106 и 107, к служащему с идентификатором 102. (использовать предложение IN).
5. Установить личный план $100000 всем служащим, не имеющим в настоящий момент плана.
6. Перевести Paul Cruz в Денвер и повысить до должности менеджера, увеличить его план продаж на 30 %.

Лабораторная работа № 16

Создание таблицы (CREATE TABLE)

Вплоть до этого места, мы запрашивали таблицы данных и выполняли команды по изменению этих данных, считая, что эти таблицы уже были созданы кем-то до нас. Это действительно наиболее реальная ситуация, когда небольшое количество людей создают таблицы, которые затем используются другими людьми. Работой с объектами (создание таблиц, удаление таблиц и т.д.) в SQL занимается раздел, называемый DDL (Язык Определения Данных).

Таблицы создаются командой CREATE TABLE. Эта команда создает пустую таблицу без строк. Команда CREATE TABLE в основном определяет имя таблицы в виде описания набора имен столбцов, указанных в определенном порядке. Она также определяет типы данных и размеры столбцов. Каждая таблица должна иметь по крайней мере один столбец.

Определения столбцов

Столбцы новой таблицы задаются в инструкции CREATE TABLE. Определения столбцов представляют собой заключенный в скобки список, элементы которого отделены друг от друга запятыми. Порядок следования определений столбцов в списке соответствует расположению столбцов в таблице. Каждое такое определение содержит следующую информацию:

– *имя столбца,* которое используется для ссылки на столбец в инструкциях SQL. Каждый столбец в таблице должен иметь уникальное имя, но в разных таблицах имена столбцов могут совпадать;

– *тип данных столбца,* показывающий, данные какого вида хранятся в столбце. Для некоторых типов, например varchar и decimal, требуется дополнительная информация такая, как размерность или число цифр после запятой. Эта информация заключается в скобки и должна следовать за ключевым словом, задающим тип данных;

– *значение по умолчанию,* которое заносится в столбец в том случае, если в инструкции insert для таблицы не указано значение данного столбца;

– *указание на то, обязательно ли столбец должен содержать данные.* Ограничение NOT NULL, если оно задано, предотвращает занесение в столбец значений NULL, в противном случае значения NULL допускаются.

Стандарт SQL2 позволяет указывать в определении столбца несколько дополнительных элементов, с помощью которых можно ограничить значения данных в столбце, а также установить, что столбец должен содержать уникальные значения либо являться первичным или внешним ключом.

Ниже приведено несколько простых инструкций CREATE TABLE для таблиц из базы данных.

Определить таблицу *offices* и ее столбцы.

CREATE TABLE offices (office INTEGER NOT NULL, city VARCHAR(15) NOT NULL, region VARCHAR(10) NOT NULL, mgr INTEGER, target INTEGER, sales INTEGER NOT NULL)

Определить таблицу *orders* и ее столбцы.

CREATE TABLE orders (order_num INTEGER NOT NULL, order_date DATE NOT NULL, cust INTEGER NOT NULL, rep INTEGER, mfr CHAR(3) NOT NULL, product CHAR(5) NOT NULL, qty INTEGER NOT NULL, amount INTEGER NOT NULL)

В различных СУБД инструкции CREATE TABLE для одной и той же таблицы могут немного отличаться, поскольку каждая СУБД поддерживает собственный набор типов данных и использует собственные ключевые слова для их идентификации в определениях столбцов.

В определении каждого столбца указывается, допускается ли хранение в нем значений NULL. В большинстве СУБД и в стандарте SQL по умолчанию считается, что значения NULL допускаются. Если же столбец обязательно должен содержать данные, необходимо включить ограничение NOT NULL.

Как стандарт SQL2, так и многие реляционные СУБД поддерживают задание для столбцов значений по умолчанию, которое указывается в определении столбца. Вот пример инструкции CREATE TABLE для таблицы *offices*, задающей значения по умолчанию.

Определить таблицу *offices* со значениями по умолчанию (синтаксис стандарта ANSI/ISO).

CREATE TABLE offices (office INTEGER NOT NULL, city VARCHAR(15) NOT NULL, region VARCHAR(10) NOT NULL DEFAULT 'Eastern', mgr INTEGER DEFAULT 106, target INTEGER DEFAULT NULL, sales INTEGER NOT NULL DEFAULT 0)

Задания

1. Создать таблицу с образованием служащего (и заполнить ее значениями: *none, bachelor, master, doctor*).

2. Создать таблицу с должностями (и заполнить ее).
3. Создать таблицу с днями рождения служащих.

Лабораторная работа № 17

Удаление таблицы (DROP TABLE). Изменение определения таблицы (ALTER TABLE)

Удаление таблицы (инструкция DROP TABLE)

С течением времени структура базы данных изменяется. Для представления новых объектов создаются новые таблицы, а некоторые старые таблицы становятся ненужными. Эти ненужные таблицы можно удалить из базы данных посредством инструкции DROP TABLE.

Инструкция содержит имя удаляемой таблицы. Обычно пользователь удаляет одну из своих собственных таблиц и указывает в инструкции простое имя таблицы. Имея соответствующее разрешение, можно также удалить таблицу другого пользователя, но в этом случае необходимо указать полное имя таблицы. Вот несколько примеров инструкции DROP TABLE.

Таблица customers была заменена двумя новыми таблицами (*cust_info и account_info*) и больше не нужна.

DROP TABLE customers

Сэм дает вам разрешение удалить таблицу birthdays.

DROP TABLE SAM.*BIRTHDAYS*

Когда инструкция DROP TABLE удаляет из базы данных таблицу, ее определение и все содержимое теряется. Восстановить данные невозможно, и, чтобы повторно создать определение таблицы, пришлось бы использовать новую инструкцию CREATE TABLE. Так как выполнение инструкции DROP TABLE может привести к серьезным последствиям, пользоваться ею следует осторожно.

Стандарт SQL2 требует, чтобы инструкция DROP TABLE включала в себя либо параметр *cascade*, либо *restrict*, которые

определяют, как влияет удаление таблицы на другие объекты базы данных, зависящие от этой таблицы. Если задан параметр cascade и в базе данных имеются объекты, которые содержат ссылку на удаляемую таблицу, то выполнение инструкции DROP TABLE закончится неуспешно. В большинстве коммерческих СУБД допускается применение инструкции DROP TABLE без каких-либо параметров.

Изменение определения таблицы (инструкция ALTER TABLE)

В процессе работы с таблицей у пользователя часто возникает необходимость добавить в нее некоторую информацию. Например, в учебной базе данных может потребоваться:

– добавить в каждую строку таблицы *customers* имя и номер телефона служащего компании-клиента, через которого поддерживается контакт, если необходимо использовать эту таблицу для связи с клиентами;

– добавить в таблицу *products* столбец с указанием минимального количества, чтобы база данных могла автоматически предупреждать о том, что запас какого-либо товара стал меньше допустимого предела;

– сделать столбец *region* в таблице *offices* внешним ключом для вновь созданной таблицы *regions,* первичным ключом которой является название региона;

– удалить определение внешнего ключа для столбца *cust* таблицы *orders,* связывающего ее с таблицей *customers,* и заменить его определениями двух внешних ключей, связывающих столбец *cust* с двумя вновь созданными таблицами *cust_info* и *account_info.*

Эти и другие изменения можно осуществить с помощью инструкции ALTER TABLE, синтаксическая диаграмма которой изображена на рис. 17.1. Данная инструкция, как и DROP TABLE, обычно применяется пользователем по отношению к своим собственным таблицам. Однако, имея соответствующее разрешение и используя полное имя таблицы, можно изменять

таблицы других пользователей. Как видно из рисунка, инструкция ALTER TABLE может:

– добавить в таблицу определение столбца;
– удалить столбец из таблицы;
– изменить значение по умолчанию для какого-либо столбца;
– добавить или удалить первичный ключ таблицы;
– добавить или удалить внешний ключ таблицы;
– добавить или удалить условие уникальности;
– добавить или удалить условие назначения.

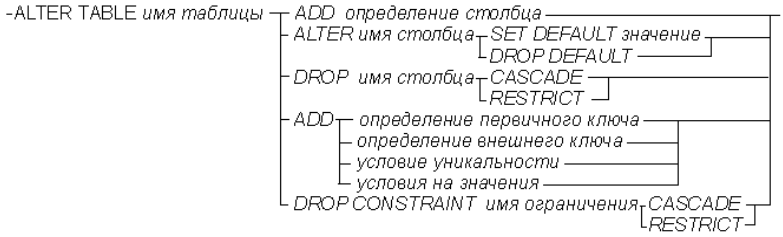


Рис. 17.1. Синтаксическая диаграмма инструкции ALTER TABLE

Предложения на рис. 17.1 изображены в соответствии со стандартом SQL. Во многих СУБД некоторые из них не используются либо используются специфические для конкретной СУБД предложения, которые изменяют другие, не представленные здесь характеристики таблицы. Стандарт SQL2 требует, чтобы инструкция ALTER TABLE применялась для единичного изменения таблицы. Например, для добавления столбца и определения нового внешнего ключа потребуются две различные инструкции. В некоторых СУБД это ограничение ослаблено и допускается присутствие нескольких предложений в одной инструкции ALTER TABLE.

Добавление столбца

Чаще всего инструкция ALTER TABLE применяется для добавления столбца в существующую таблицу. Предложение с определением столбца в инструкции ALTER TABLE имеет точ-

но такой же вид, как и в инструкции CREATE TABLE и выполняет ту же самую функцию. Новое определение добавляется в конец определений столбцов таблицы, и в последующих запросах новый столбец будет крайним справа. СУБД обычно предполагает, что новый столбец во всех существующих строках содержит значения NULL. Если столбец объявлен как NOT NULL WITH DEFAULT, то СУБД считает, что он содержит значения по умолчанию. Обратите внимание на то, что нельзя объявлять столбец просто как NOT NULL, поскольку СУБД подставляла бы в существующие строки значения NULL, нарушая тем самым заданное условие. В действительности, когда вы добавляете новый столбец, СУБД не заносит во все существующие строки нового столбца значения NULL или значения по умолчанию. СУБД обнаруживает тот факт, что строка слишком коротка для нового определения таблицы, только при выборке этой строки пользователем, и расширяет ее значениями NULL или значениями по умолчанию непосредственно перед выводом на экран или передачей в программу пользователя.

Ниже даны примеры инструкций ALTER TABLE, добавляющих новые столбцы.

Добавить контактный телефон и имя служащего компании-клиента в таблицу *customers*.

ALTER TABLE customers ADD contact_name VARCHAR(30)
ALTER TABLE customers ADD contact_phone CHAR(10)

Добавить в таблицу *products* столбец с данными о минимально допустимом количестве товара на складе.

ALTER TABLE products ADD min_qty INTEGER NOT NULL WITH DEFAULT 0

В первом примере новые столбцы будут иметь значения NULL для существующих клиентов. Во втором примере столбец *min_qty* для существующих товаров будет содержать нули (0), что вполне уместно.

Когда инструкция ALTER TABLE впервые появилась в реляционных СУБД, единственными структурными элементами

таблиц были определения столбцов, поэтому понятно, что означает предложение ADD этой инструкции. Со временем таблицы стали включать определения первичных и внешних ключей и прочих ограничений, а в предложении ADD указывался тип добавляемого ограничения. В целях унификации стандарт SQL2 позволяет добавлять после ключевого слова ADD слово COLUMN для явного указания на то, что создается столбец. С учетом этого предыдущий пример можно записать так.

Добавить в таблицу products столбец с данными о минимально допустимом количестве товара на складе.

ALTER TABLE products ADD COLUMN min_qty INTEGER NOT NULL WITH DEFAULT 0

Удаление столбца

С помощью инструкции ALTER TABLE можно удалить из существующей таблицы один или несколько столбцов, если в них больше нет необходимости. Вот пример удаления столбца *hire_date* из таблицы *salesreps*.

Удалить столбец hire_date из таблицы salesreps.

ALTER TABLE salesreps DROP hire_date

Стандарт SQL2 требует, чтобы одна инструкция ALTER TABLE использовалась для удаления только одного столбца, но в ряде ведущих СУБД такое ограничение снято.

Следует учитывать, что операция удаления столбца вызывает те же проблемы целостности данных, что и при обновлении таблиц. В частности, при удалении столбца, являющегося первичным ключом в каком-либо отношении, связанные с ним внешние ключи становятся недействительными. Похожая проблема возникает, когда удаляется столбец, участвующий в проверке ограничения на значения другого столбца.

Описанные проблемы в стандарте SQL2 решены так же, как и в случае инструкций DELETE и UPDATE, (с помощью правила удаления). Можно выбрать одно из двух правил:

– *restrict:* если с удаляемым столбцом связан какой-либо объект в базе данных (внешний ключ, ограничение и т.п.), инструкция ALTER TABLE завершится выдачей сообщения об ошибке и столбец не будет удален;

– *cascade:* любой объект базы данных (внешний ключ, ограничение и т.п.), связанный с удаляемым столбцом, также будет удален.

Правило *cascade* может вызвать целую лавину изменений, поэтому применять его следует с осторожностью. Лучше указывать правило *restrict*, а связанные внешние ключи или ограничения обрабатывать с помощью дополнительных инструкций типа ALTER или DROP.

Задания

1. Добавить в таблицу служащих:
а) пол служащего;
б) уровень образования;
в) количество детей;
г) дату рождения.
2. Разбить дату рождения по годам, дням, месяцам.

СОДЕРЖАНИЕ

Учебное издание

ТЕХНОЛОГИЯ БАЗ ДАННЫХ

Лабораторный практикум

С о с т а в и т е л и :
ПРОНКЕВИЧ Сергей Александрович
ЧИГАРЕВ Анатолий Власович

Редактор И.Ю. Никитенко
Компьютерная верстка Л.А. Адамович

Подписано в печать 16.04.2010.
Формат 60×84 1/16. Бумага офсетная.
Отпечатано на ризографе. Гарнитура Таймс.
Усл. печ. л. 4,88. Уч.-изд. л. 3,82. Тираж 50. Заказ 723.

Издатель и полиграфическое исполнение:
Белорусский национальный технический университет.
ЛИ № 02330/0494349 от 16.03.2009.
Проспект Независимости, 65. 220013, Минск.