МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ,
МОЛОДЕЖИ И СПОРТА УКРАИНЫ
ОДЕССКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ
имени И.И. МЕЧНИКОВА

Институт математики, экономики и механики

Кафедра математического обеспечения компьютерных систем

Т. И. Петрушина, Д. В. Коломиец

МЕТОДИЧЕСКОЕ ПОСОБИЕ
по курсу
«Базы данных и информационные системы»

2011

УДК 004.655
ББК 32.973.26 – 018.2
П31

Методическое пособие по курсу «Базы данных и информационные системы».

Методическое пособие к базовому курсу «Базы данных и информационные системы» содержит краткое описание языка структурированных запросов SQL, описание модельной базы данных DreamHome, для которой в пособии приводятся примеры запросов, описание заданий лабораторных работ по курсу.

В пособии делается акцент на практическое освоение языка SQL для серверов баз данных Microsoft Access и Microsoft SQL Server.

Предназначено для студентов, обучающихся по специальности «Прикладная математика» на дневном отделении и для всех, кто самостоятельно изучает язык SQL.

***Авторы**:*

Т.И. Петрушина,
кандидат физико-математических наук, доцент

Д.В. Коломиец,
ассистент кафедры математического обеспечения компьютерных систем

***Рецензенты**:*

Ю.Н. Крапивный, кандидат физико–математических наук, доцент

В.Г. Пенко, кандидат технических наук, доцент

Рекомендовано к изданию Учёным советом
Института математики, экономики и механики ОНУ
имени И. И. Мечникова.
Протокол № 1 от 9 октября 2009 года.

Содержание

Введение

Данное методическое пособие посвящено традиционно важному разделу основного курса «Базы данных и информационные системы» - изучению языка структурированных запросов к базе данных SQL.

SQL (Structured Query Language) представляет собой непроцедурный язык, используемый для управления данными реляционных СУБД. Термин «непроцедурный» означает, что на данном языке можно сформулировать, что нужно сделать с данными, но нельзя задать конкретный алгоритм, как именно это следует сделать. Иными словами, в этом языке отсутствуют алгоритмические конструкции, такие как присваивания, операторы цикла, разветвления, переключатели и др.

Язык SQL был создан в начале 70-х годов в результате исследовательского проекта IBM, целью которого было создание языка манипуляции реляционными данными. Первоначально он назывался SEQUEL (Structured English Query Language), затем — SEQUEL/2, а затем — просто SQL. Официальный стандарт SQL был опубликован ANSI (American National Standards Institute — Национальный институт стандартизации, США) в 1986 году. Затем этот стандарт был расширен в 1989 и 1992 годах, поэтому стандарт SQL носит название SQL92, и это наиболее часто используемая версия SQL. В настоящее время опубликован стандарт SQL3, содержащий некоторые объектно-ориентированные расширения.

Существует три уровня соответствия стандарту ANSI — начальный, промежуточный и полный. Многие производители серверных СУБД, такие как IBM, Informix, Microsoft, Oracle и Sybase, применяют собственные реализации SQL, основанные на стандарте ANSI (отвечающие как минимум начальному уровню соответствия стандарту) и содержащие некоторые расширения, специфические для данной СУБД.

Более подробную информацию о соответствии стандарту версии SQL, используемой в конкретной СУБД, можно найти в документации, поставляемой с этой СУБД.

1. Общие сведения о языке SQL

1.1. Как работает SQL

Рассмотрим, как работает SQL. Предположим, что у нас имеется база данных, управляемая с помощью какой-либо СУБД. Для извлечения из нее данных используется запрос, сформулированный на языке SQL. СУБД обрабатывает этот запрос, извлекает запрашиваемые данные и возвращает их. Результат выполнения запроса – это всегда таблица, содержащая выбранные из базы данные. Этот процесс схематически изображен на рисунке 1.1.

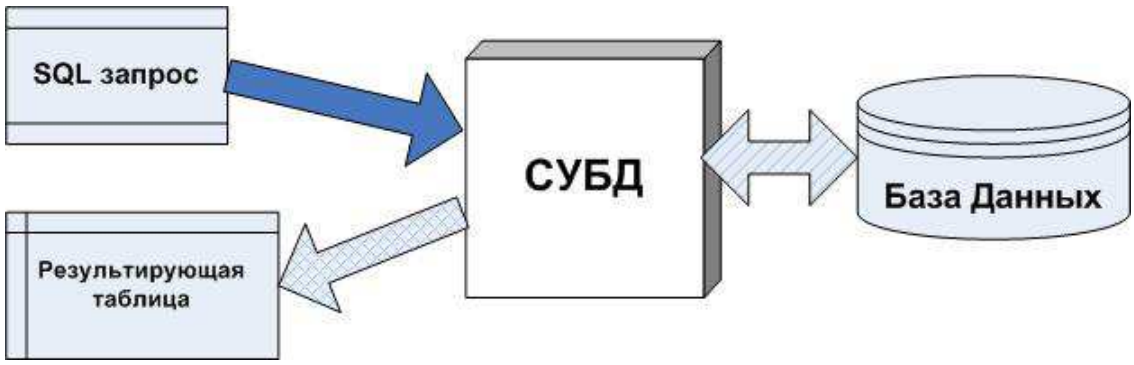

Рис. 1.1. Выполнение SQL запроса.

Как мы увидим позже, SQL позволяет не только извлекать данные, но и добавлять, удалять и изменять данные, определять структуру данных, ограничивать или предоставлять доступ к данным, поддерживать ссылочную целостность и многое другое.

Обратите внимание на то, что SQL сам по себе не является ни СУБД, ни отдельным продуктом. Это язык, применяемый для взаимодействия с СУБД и являющийся в определенном смысле ее неотъемлемой частью.

1.2. Состав операторов языка SQL

SQL содержит примерно 40 операторов для выполнения различных операций над данными, хранящимися в базе данных, с помощью СУБД. Эти операторы подразделяются на пять категорий:

- операторы определения данных (по-английски *Data Definition Language, DDL*),
- операторы манипулирования данными (по-английски *Data Manipulation Language, DML*),
- операторы управления транзакциями (по-английски *Transaction Control Language, TCL*),
- операторы определения доступа к данным (по-английски *Data Control Language, DCL*),
- операторы управления курсорами (по-английски *Cursor Control Language, CCL*).

Далее приводится краткое описание операторов каждой категории.

1.2.1. Data Definition Language (DDL)

Data Definition Language содержит операторы, позволяющие создавать, изменять и удалять базы данных и объекты внутри них (таблицы, представления и др.). Эти операторы перечислены в таблице 1.1.

Таблица 1.1. Операторы определения данных

Оператор	**Описание**
CREATE TABLE	Применяется для добавления новой таблицы к базе данных
DROP TABLE	Применяется для удаления таблицы из базы данных
ALTER TABLE	Применяется для изменения структуры имеющейся таблицы
CREATE VIEW	Применяется для добавления нового представления к базе данных

DROP VIEW	Применяется для удаления представления из базы данных
CREATE INDEX	Применяется для создания индекса для данного поля
DROP INDEX	Применяется для удаления существующего индекса
CREATE SCHEMA	Применяется для создания новой схемы в базе данных
DROP SCHEMA	Применяется для удаления схемы из базы данных
CREATE DOMAIN	Применяется для создания нового домена
ALTER DOMAIN	Применяется для переопределения домена
DROP DOMAIN	Применяется для удаления домена из базы данных

1.2.2. Data Manipulation Language (DML)

Data Manipulation Language содержит операторы, позволяющие выбирать, добавлять, удалять и модифицировать данные. Обратите внимание на то, что эти операторы не обязаны завершать транзакцию, внутри которой они вызваны. Операторы DML перечислены в таблице 1.2.

Таблица 1.2. Операторы манипулирования данными

Оператор	**Описание**
SELECT	Применяется для выбора данных
INSERT	Применяется для добавления строк к таблице
DELETE	Применяется для удаления строк из таблицы
UPDATE	Применяется для изменения данных

Иногда оператор SELECT относят к отдельной категории, называемой языком запроса данных, по-английски *Data Query Language (DQL)*.

1.2.3. Transaction Control Language (TCL)

Операторы Transaction Control Language применяются для управления транзакциями, т.е. изменениями, выполненными группой операторов DML. Операторы TCL перечислены в таблице 1.3.

Таблица 1.3. Операторы управления транзакциями

Оператор	Описание
COMMIT	Применяется для завершения транзакции и сохранения изменений в базе данных
ROLLBACK	Применяется для отката транзакции и отмены изменений в базе данных
SET TRANSACTION	Применяется для установки параметров доступа к данным в текущей транзакции

1.2.4. Data Control Language (DCL)

Операторы Data Control Language, иногда называемые операторами языка управления доступом, по-английски, Access Control Language (ACL), применяются для выполнения функций администрирования данных, присваивающих или отменяющих право (привилегию) использовать базу данных, таблицы в базе данных, а также выполнять те или иные операторы SQL. Операторы DCL представлены в таблице 1.4.

Таблица 1.4. Операторы определения доступа к данным

Оператор	Описание
GRANT	Применяется для присвоения прав доступа (привилегии)
REVOKE	Применяется для отмены права доступа (привилегии)

1.2.5. Cursor Control Language (CCL)

Операторы Cursor Control Language используются для определения курсора, подготовки SQL-предложений для выполнения, а также для некоторых других операторов. Операторы CCL представлены в таблице 1.5.

Таблица 1.5. Операторы управления курсорами

Оператор	Описание
DECLARE CURSOR	Применяется для определения курсора для запроса
EXPLAIN	Применяется для описания плана запроса. Этот оператор представляет собой расширение SQL для Microsoft SQL Server 7.0. Он не обязан выполняться в других СУБД.
OPEN CURSOR	Применяется для открытия курсора при получении результатов запроса
FETCH	Применяется для получения строки из результатов запроса
CLOSE CURSOR	Применяется для закрытия курсора
PREPARE	Применяется для подготовки оператора SQL для выполнения
EXECUTE	Применяется для выполнения оператора SQL
DESCRIBE	Применяется для описания подготовленного запроса

1.3. Формат оператора SQL

Все операторы SQL имеют формат, показанный на рисунке 1.2.

Каждый оператор SQL начинается с глагола, представляющего собой ключевое слово, определяющее, что именно делает этот оператор (SELECT, INSERT, DELETE...). Оператор состоит из частей - предложений, определяющих, над какими данными производятся операции. Каждое предложение начинается с ключевого слова, такого как SELECT, FROM, WHERE и др. Структура предложения зависит от его типа — ряд предложений содержит имена таблиц или полей, некоторые

могут содержать дополнительные ключевые слова, константы или выражения.

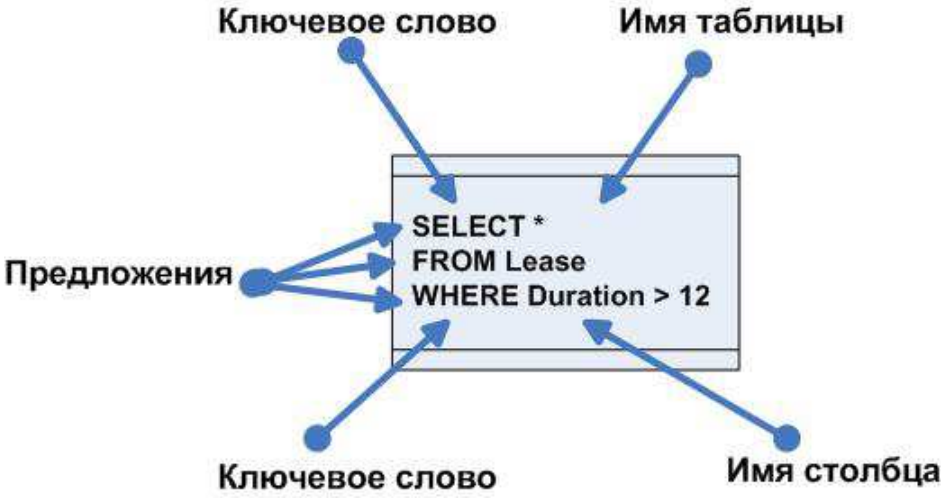

Рис. 1.2. Формат оператора SQL.

1.4. Ключевые слова ANSI/ISO SQL92

Ключевые слова языка SQL задаются определенными словами английского языка, их написание не может быть изменено, а назначение – переопределено.

Ключевые слова, определенные в стандарте ANSI SQL, не могут быть использованы в качестве имен объектов баз данных (таблиц, полей, имен пользователей). Список ключевых слов языка SQL приведен в таблице 1.6.

Таблица 1.6. Ключевые слова языка SQL

ABS	CROSS	GET	NEXT	SPACE
ACTION	CURRENT	GLOBAL	NO	SQL
ADD	CURRENT_DATE	GO	NOT	SQLCODE
ALL	CURRENT_TIME	GOTO	NULL	SQLERROR
ALLOCATE	CURRENT_TIMESTAMP	GRANT	OCTET_LENGTH	SQLSTATE
ALTER	CURRENT_USER	GROUP	OF	SUBSTRING
AND	CURSOR	HAVING	ON	SUM
ANY	DATE	HOUR	ONLY	SYSTEM_USER
ARE	DAY	IDENTITY	OPEN	TABLE

AS	**DEALLOCATE**	**IMMEDIATE**	**OPTION**	**TEMPORARY**
ASC	**DEC**	**IN**	**OR**	**THEN**
ASSERTION	**DECIMAL**	**INDICATOR**	**ORDER**	**TIME**
AT	**DECLARE**	**INITIALLY**	**OUTER**	**TIMESTAMP**
AUTHORIZATION	**DEFAULT**	**INNER**	**OUTPUT**	**TIMEZONE_ HOUR**
AVG	**DEFERRABLE**	**INPUT**	**OVERLAPS**	**TIMEZONE_MINUTE**
BEGIN	**DEFERRED**	**INSENSITIVE**	**PAD**	**TO**
BETWEEN	**DELETE**	**INSERT**	**PARTIAL**	**TRAILING**
BIT	**DESC**	**INT**	**POSITION**	**TRANSACTION**
BIT_LENGTH	**DESCRIBE**	**INTEGER**	**PRECISION**	**TRANSLATE**
BOTH	**DESCRIPTOR**	**INTERSECT**	**PREPARE**	**TRANSLATION**
BY	**DIAGNOSTICS**	**INTERVAL**	**PRESERVE**	**TRIM**
CASCADE	**DISCONNECT**	**INTO**	**PRIMARY**	**TRUE**
CASCADED	**DISTINCT**	**IS**	**PRIOR**	**UNION**
CASE	**DOMAIN**	**ISOLATION**	**PRIVILEGES**	**UNIQUE**
CAST	**DOUBLE**	**JOIN**	**PROCEDURE**	**UNKNOWN**
CATALOG	**DROP**	**KEY**	**PUBLIC**	**UPDATE**
CHAR	**ELSE**	**LANGUAGE**	**READ**	**UPPER**
CHARACTER	**END**	**LAST**	**REAL**	**USAGE**
CHAR_LENGTH	**END-EXEC**	**LEADING**	**REFERENCES**	**USER**
CHARACTER_ LENGTH	**ESCAPE**	**LEFT**	**RELATIVE**	**USING**
CHECK	**EXCEPT**	**LEVEL**	**RESTRICT**	**VALUE**
CLOSE	**EXCEPTION**	**LIKE**	**REVOKE**	**VALUES**
COALESCE	**EXEC**	**LOCAL**	**RIGHT**	**VARCHAR**
COLLATE	**EXECUTE**	**LOWER**	**ROLLBACK**	**VARYING**
COLLATION	**EXISTS**	**MATCH**	**ROWS**	**VIEW**
COLUMN	**EXTERNAL**	**MAX**	**SCHEMA**	**WHEN**
COMMIT	**EXTRACT**	**MIN**	**SCROLL**	**WHENEVER**
CONNECT	**FALSE**	**MINUTE**	**SECOND**	**WHERE**
CONNECTION	**FETCH**	**MODULE**	**SECTION**	**WITH**
CONSTRAINT	**FIRST**	**MONTH**	**SELECT**	**WORK**
CONSTRAINTS	**FLOAT**	**NAMES**	**SESSION**	**WRITE**
CONTINUE	**FOR**	**NATIONAL**	**SESSION_USER**	**YEAR**
CONVERT	**FOREIGN**	**NATURAL**	**SET**	**ZONE**
CORRESPONDING	**FOUND**	**NCHAR**	**SIZE**	
COUNT	**FROM**	**NULLIF**	**SMALLINT**	
CREATE	**FULL**	**NUMERIC**	**SOME**	

Стандарт SQL также включает список потенциальных ключевых слов, зарезервированных для последующих версий стандарта SQL. Эти ключевые слова приведены в таблице 1.7.

Таблица 1.7. Список потенциальных ключевых слов языка SQL.

AFTER	EQUALS	OLD	RETURN	TEST
ALIAS	GENERAL	OPERATION	RETURNS	THERE
ASYNC	IF	OPERATORS	ROLE	TRIGGER
BEFORE	IGNORE	OTHERS	ROUTINE	TYPE
BOOLEAN	LEAVE	PARAMETERS	ROW	UNDER
BREADTH	LESS	PENDANT	SAVEPOINT	VARIABLE
COMPLETION	LIMIT	PREORDER	SEARCH	VIRTUAL
CALL	LOOP	PRIVATE	SENSITIVE	VISIBLE
CYCLE	MODIFY	PROTECTED	SEQUENCE	WAIT
DATA	NEW	RECURSIVE	SIGNAL	WHILE
DEPTH	NONE	REF	SIMILAR	WITHOUT
DICTIONARY	OBJECT	REFERENCING	SQLEXCEPTION	
EACH	OFF	REPLACE	SQLWARNING	
ELSEIF	OID	RESIGNAL	STRUCTURE	

1.5. Как выполнять SQL-операторы

Все современные серверные СУБД (а также многие популярные настольные СУБД) содержат в своем составе утилиты, позволяющие выполнить SQL-оператор и ознакомиться с его результатом. Например, Microsoft SQL Server имеет в своем составе утилиту SQL Query Analyzer. Для демонстрации результатов выполнения примеров SQL-операторов в данном пособии используется база данных DreamHome, структура и содержимое которой описано в третьем разделе этого пособия.

2. Операторы языка SQL

В этом разделе мы изучим различные операторы SQL, включая операторы для выборки данных, их добавления, удаления или изменения, изменения метаданных и пр.

2.1. Выборка данных

Выборка данных, хранящихся в базе данных, представляет собой наиболее часто встречающуюся операцию, выполняемую с помощью SQL. Оператор SELECT — один из самых важных операторов этого языка, применяемый для выборки данных. Синтаксис этого оператора имеет следующий вид:

SELECT *список_столбцов*
FROM *список_таблиц*
[WHERE *предложение_where*]
[GROUP BY *предложение* group by]
[ORDER BY *предложение_order_by*]

Операторы SELECT обязательно должны содержать слова SELECT и FROM; другие ключевые слова, такие как WHERE или ORDER BY, являются необязательными.

За ключевым словом SELECT следуют сведения о том, какие именно поля необходимо включить в результирующую таблицу данных. Звездочка (*) обозначает все поля таблицы, например:

```
SELECT *
```

Для выбора одного столбца таблицы применяется следующий синтаксис:

```
SELECT LName
```

Пример выборки нескольких колонок имеет вид:

```
SELECT LName, FName, Position
```

Если выборка данных осуществляется из нескольких таблиц и при этом выбираются одноименные поля из разных таблиц, надо дополнительно указывать имена таблиц для полной идентификации полей, включаемых в результирующую таблицу данных, например:

```
SELECT Staff.LName, Client.LName
```

2.1.1. Предложение SELECT

В предложении SELECT указывается, какие данные надо получить в результирующей таблице.

Как было сказано ранее, в список выбора могут входить поля таблиц, из которых выполняется выборка. В результирующей таблице элементам списка выбора можно дать новые имена с помощью ключевого слова AS. Например:

```
SELECT StaffNo AS [Номер сотрудника]
FROM Staff
```

Над полями таблиц, из которых выполняется выборка, можно выполнять вычисления в предложении SELECT. Например:

```
SELECT StaffNo AS [Номер сотрудника], Salary/8.30 AS [Оклад в долларах]
FROM Staff
```

К полям таблиц в списке отбора можно применять математические и строковые функции (см. таблицу 2.4), а также агрегатные функции (см. таблицу 2.3).

2.1.2. Предложение FROM

Для указания имен таблиц, из которых выбираются записи, применяется ключевое слово FROM, например:

```
SELECT *
FROM Staff
```

Этот запрос возвратит все поля из таблицы Staff.

Если в результирующей таблице нужны только поля LName и FName, мы можем ввести следующее предложение SELECT:

```
SELECT LName, FName
FROM Staff
```

Пример запроса более чем к одной таблице приведен ниже:

```
SELECT Staff.LName, Client.LName
FROM Staff, Client
```

2.1.3. Предложение WHERE

Для фильтрации результатов, возвращаемых оператором SELECT, можно использовать предложение WHERE, синтаксис которого имеет вид:

WHERE *выражение1* [{AND | OR} *выражение2* […]]

Например, вместо получения полного списка сотрудников можно ограничиться только теми из них, у которых значение поля Position (должность) равно ‘Менеджер’:

```
SELECT *
FROM Staff
WHERE Position = ‘Менеджер’
```

В предложении WHERE можно использовать различные выражения, например:

```
SELECT *
FROM Lease
WHERE LeaseClientNo = ‘C0012’ AND Duration > 6
```

или:

```
SELECT PropertyNo, City, Street, PostCode
FROM PropertyForRent
WHERE Type = ‘Квартира’ OR Rooms < 3
```

или:

```
SELECT LeaseNo, Rent,
FROM Lease
WHERE RentFinish IS NOT NULL
```

Выражение ‘IS NOT NULL’ означает, что соответствующий столбец результирующей таблицы не может содержать пустые значения.

В предложении WHERE можно использовать один из шести операторов отношений, определенных в SQL. Эти операторы приведены в таблице 2.1.

Таблица 2.1. Операторы отношений, применяемые в выражениях

Оператор	**Описание**
<	Меньше
<=	Меньше или равно
<>	Не равно
=	Равно
>	Больше
>=	Больше или равно

Помимо перечисленных выше простых операторов сравнения, можно использовать и специальные операторы сравнения, приведенные в таблице 2.2.

Таблица 2.2. Специальные операторы отношений

Оператор	**Описание**
ALL	Применяется совместно с операторами сравнения при сравнении со списком значений
ANY	Применяется совместно с операторами сравнения при сравнении со списком значений
BETWEEN	Применяется при проверке нахождения значения внутри заданного интервала (включая его границы)
IN	Применяется для проверки наличия значения в списке
LIKE	Применяется при проверке соответствия значения заданной маске

Приведем несколько примеров применения этих операторов. Для сопоставления данных с маской применяется ключевое слово LIKE:

```
SELECT LName, FName
FROM Client
WHERE LName LIKE ‘M%’
```

В данной маске символ ‘%’ (процент) заменяет любую последовательность символов, а символ ‘_’ (подчеркивание) — один любой символ. Тот же самый результат может быть получен следующим способом:

```
SELECT LName, FName
FROM Client
WHERE FName BETWEEN ‘M’ AND ‘H’
```

В последнем примере мы можем расширить область поиска. В частности, при поиске Клиентов по фамилиям, начинающимся с букв от А до Д, можно выполнить следующий оператор SELECT:

```
SELECT LName, FName
FROM Client
WHERE LName BETWEEN ‘А’ AND ‘Д’
```

Используя оператор LIKE, мы можем сузить диапазон поиска, применив более сложную маску для сравнения. Например, чтобы найти клиентов, фамилии которых содержат подстроку ‘ван’, можно применить следующий запрос:

```
SELECT LName, FName
FROM Client
WHERE LName LIKE ‘%ван%’
```

Маска ‘%ван%’ показывает, что до и после искомой подстроки может быть любое количество произвольных символов.

Используя оператор IN, можно задать список значений, в котором должно содержаться значение поля:

```
SELECT LName, FName
FROM Client
WHERE ClientNo IN (‘C0127’, ‘C0315’, ‘C0012’)
```

2.1.4. Операторы AND, OR и NOT

Мы уже рассматривали пример применения оператора AND для логических операций, связанных с требованием, чтобы запись удовлетворяла двум разным критериям. Рассмотрим следующий запрос:

```
SELECT LName, FName
FROM Client
WHERE LName LIKE ‘C%’ AND PrefType = ‘дом’
```

Результатом выполнения этого запроса будет список клиентов, которые хотят арендовать дом, фамилия которых начинается с буквы ‘C’.

Оператор OR позволяет выбрать записи, удовлетворяющие хотя бы одному из перечисленных условий, в то время как оператор NOT используется для исключения из выборки записей, удовлетворяющих данному условию. Например, можно применить оператор OR для поиска всех клиентов, которые либо хотят арендовать дом, либо имеющих фамилию, начинающуюся с буквы ‘C’ (и при этом не важно, что они хотят арендовать):

```
SELECT LName, FName
FROM Client
WHERE LName LIKE ‘C%’ OR PrefType = ‘дом’
```

В этом случае результирующая таблица будет содержать записи, в которых значение поля LName удовлетворяет первому условию, плюс все записи, в которых значение поля PrefType удовлетворяет второму условию.

Теперь рассмотрим пример применения оператора NOT. Для исключения некоторых клиентов из результирующей таблицы можно использовать запрос вида:

```
SELECT LName, FName
FROM Client
WHERE PrefType NOT IN (‘дача’, ‘офис’)
```

В результате выполнения этого запроса мы получим список клиентов, которые хотят арендовать любой объект недвижимости, кроме дачи и офиса.

2.1.5. Предложение ORDER BY

Предложение ORDER BY (необязательное) применяется для сортировки строк результирующей таблицы по одному или нескольким столбцам. Для определения порядка сортировки используются ключевые слова ASC (по возрастанию) или DESC (по убыванию). По умолчанию данные сортируются по возрастанию. Синтаксис предложения ORDER BY имеет вид:

ORDER BY *столбец1* [{ASC | DESC}][, *столбец2* [{ASC | DESC}] [,…]

Например, для сортировки сотрудников по фамилии и затем по имени следует использовать следующий SQL-запрос:

```
SELECT LName, FName, Position
FROM Staff
ORDER BY LName, FName
```

Если сортировка данных требуется в убывающем порядке (например, требуется список объектов недвижимости в порядке убывания арендной платы), используется ключевое слово DESC:

```
SELECT PropertyNo, City, Street, PostCode, Rent
FROM PropertyForRent
ORDER BY Rent DESC
```

2.1.6. Связывание таблиц

Как мы уже убедились, можно создавать запросы, позволяющие извлечь данные из нескольких таблиц. Одна из возможностей сделать это заключается в связывании таблиц по одному или нескольким полям. Обратите внимание на то, что без связывания таблиц в результате запроса

получится таблица, содержащая все возможные комбинации строк каждой из исходных таблиц (эта операция известна также как декартово произведение):

```
SELECT Staff.*, Branch.*
FROM Staff, Branch
```

в то время как запрос, показанный ниже, приводит к отображению списка сотрудников с указанием, в каком филиале он работает:

```
SELECT Staff.*, Branch.*
FROM Staff, Branch
WHERE Staff.StaffBranchNo = Branch.BranchNo
```

В общем случае синтаксис для связывания таблиц имеет вид:

```
SELECT список_полей
FROM таблица1, таблица2
WHERE таблица1.столбец1 = таблица2.столбец2
```

Следующие несколько примеров связывания таблиц характерны для Microsoft Access и Microsoft SQL Server и могут не работать с другими СУБД.

Существует несколько типов связывания таблиц. Например, следующий оператор SQL осуществляет так называемое внутреннее соединение таблиц (inner join) — в этом случае в результирующей таблице содержатся записи, в которых значения в связанных полях совпадают:

```
SELECT Staff.*, Branch.*
FROM Staff INNER JOIN Branch ON Staff.StaffBranchNo = Branch.BranchNo
```

Так называемые внешние соединения (outer joins) позволяют нам включить в результат запроса все строки из одной таблицы и соответствующие им строки из другой таблицы. Если в другой таблице нет строк, соответствующих какой-то строке, то в результат включается пустая строка, т.е. строка, состоящая их всех пустых полей. Например:

```
SELECT Client.*, Lease.*
FROM Client LEFT OUTER JOIN Lease ON Client.ClientNo = Lease.LeaseClientNo
```

Это было так называемое левое внешнее соединение (left outer join). В результирующую таблицу попадут все клиенты, каждая строка будет соединена с договором этого клиента, а в тех случаях, когда у клиента нет ни одного договора – с пустой строкой.

Существуют также правые внешние соединения (right outer join), возвращающие все строки из второй (то есть правой) таблицы и соответствующие им строки из другой таблицы:

```
SELECT Lease.*, PropertyForRent.*
FROM Lease RIGHT OUTER JOIN PropertyForRent ON Lease.LeasePropertyNo =

    PropertyForRent.PropertyNo
```

В результирующую таблицу попадут все объекты недвижимости, каждая строка будет соединена с договором аренды этого объекта, а в тех случаях, когда на объект договора аренды не заключались – с пустой строкой.

Комбинируя левое и правое внешние соединения, можно получить полное внешнее соединение, возвращающее все данные из обеих таблиц:

```
SELECT Client.*, PropertyForRent.*
FROM Client FULL OUTER JOIN PropertyForRent ON  Client.PrefType =
                                                    PropertyForRent.Type
```

Для получения всех комбинаций строк из обеих таблиц (декартова произведения) можно использовать ключевое слово CROSS JOIN без указания связываемых полей:

```
SELECT Staff.*, Branch.*
FROM Staff CROSS JOIN Branch
```

Результат будет этого запроса совпадать с результатом первого запроса этого раздела.

Если в запросе используется более трех таблиц, можно использовать вложенные соединения.

2.1.7. Предложение GROUP BY

Для вычисления суммарных значений на основе данных одной или нескольких таблиц можно использовать предложение GROUP BY, имеющее такой синтаксис:

GROUP BY *столбец1* [, …]

Например, следующий запрос связывает две таблицы, сортирует их по полю BranchNo, для каждого значения BranchNo создает одну строку в результирующей таблице и вычисляет количество значений поля StaffBranchNo для каждого значения BranchNo:

```
SELECT Branch.BranchNo, COUNT (Staff.StaffBranchNo)
FROM Branch INNER JOIN Staff ON Branch.BranchNo = Staff.StaffBranchNo
GROUP BY Branch.BranchNo
```

В приведенном выше примере запроса мы использовали в предложении SELECT агрегатную функцию COUNT, вычисляющую количество значений. В таблице 2.3 указан список наиболее часто используемых агрегатных функций.

Таблица 2.3. Перечень некоторых агрегатных функций языка SQL

Функция	**Назначение**
AVG	Вычисляет среднее арифметическое
COUNT	Вычисляет количество непустых значений в столбце
MAX	Вычисляет наибольшее значение в столбце
MIN	Вычисляет наименьшее значение в столбце
SUM	Вычисляет сумму значений в столбце

STDEV	Вычисляет несмещенное стандартное отклонение для значений в столбце. Эта функция используется только в Microsoft Access и Microsoft SQL Server.
STDEVP	Вычисляет смещенное стандартное отклонение для значений в толбце. Эта функция используется только в Microsoft Access и Microsoft SQL Server
VAR	Вычисляет несмещенную дисперсию отклонения для значений в столбце. Эта функция используется только в Microsoft Access и Microsoft SQL Server.
VARP	Вычисляет смещенную дисперсию отклонения для значений в столбце. Эта функция используется только в Microsoft Access и Microsoft SQL Server

Помимо перечисленных выше агрегатных функций можно использовать также математические и строковые функции, приведенные в таблице 2.4.

Таблица 2.4 . Математические и строковые функции

Функция	**Назначение**
ABS	Возвращает абсолютное значение числа
CEIL	Округляет дробное число
FLOOR	Удаляет дробную часть числа
GREATEST	Возвращает наибольшее из двух значений. Эта функция используетсятолько в Microsoft Access и Microsoft SQL Server
LEAST	Возвращает наименьшее из двух значений. Эта функция используется только в Microsoft Access и Microsoft SQL Server
MOD	Возвращает остаток от деления одного числа на другое
POWER	Возвращает значение, равное одному числу в степени, равной другому числу
ROUND	Округляет число с точностью до указанного десятичного знака
SIGN	Возвращает –1, если число отрицательное, и 1, если положительное

SQRT	Вычисляет квадратный корень числа
LEFT	Возвращает указанное число знаков строки, начиная слева. Эта функция используется только в Microsoft Access и Microsoft SQL Server
RIGHT	Возвращает указанное число знаков строки, начиная справа. Эта функция используется только в Microsoft Access и Microsoft SQL Server
UPPER	Заменяет все буквы в строке на прописные
LOWER	Заменяет все буквы в строке на строчные
INITCAP	Расставляет заглавные буквы в начале слов в строке
LENGTH	Вычисляет число символов в строке
LPAD	Добавляет указанный символ в левую часть строки в количестве, необходимом для того, чтобы строка имела заданную длину
RPAD	Добавляет указанный символ в правую часть строки в количестве, необходимом для того, чтобы строка имела заданную длину
SUBSTR	Извлекает подстроку нужной длины из строки, начиная с указанной позиции

2.1.8. Предложение HAVING

Предложение HAVING имеет назначение, сходное с предложением WHERE, но используется с агрегатными данными. Например:

```
SELECT Branch.BranchNo, COUNT (Staff.StaffBranchNo)
FROM Branch INNER JOIN Staff ON Branch.BranchNo = Staff.StaffBranchNo
GROUP BY Branch.BranchNo
HAVING COUNT(Staff.StaffNo)  >=  10
```

Этот запрос аналогичен предыдущему, но в результирующую таблицу включены только филиалы, в которых работают десять или более сотрудников.

2.1.9. Ключевые слова ALL и DISTINCT

До этого момента мы рассматривали, как извлечь все или заданные столбцы из одной или нескольких таблиц. Для управления выводом дублирующихся строк в результирующей таблице можно использовать ключевые слова ALL или DISTINCT в предложении SELECT. Ключевое слово DISTINCT указывает, что строки в результирующей таблицы должны быть уникальны, тогда как ключевое слово ALL указывает, что возвращать следует все строки. Например, для извлечения названий городов, в которых имеются филиалы фирмы, можно использовать следующий запрос:

```
SELECT DISTINCT City
FROM Branch
```

Отметим, что ключевое слово ALL используется по определению. Если в запросе требуется вывести более одного столбца и при этом использовано слово DISTINCT, то результирующая таблица будет содержать различные строки, но некоторые значения одного и того же поля в разных строках могут совпадать.

2.1.10. Ключевое слово TOP

Ключевое слово TOP может быть использовано для возврата первых n строк или первых n процентов таблицы. Например, запрос:

```
SELECT TOP 10 *
FROM Lease
ORDER BY LeaseNo
```

возвращает первые 10 договоров из таблицы, тогда как запрос:

```
SELECT TOP 25 PERCENT *
FROM Lease
ORDER BY LeaseNo
```

вернет первую четверть записей таблицы.

2.2. Операторы модификации данных

До сих пор мы изучали операторы SQL для извлечения данных. Помимо этого язык SQL может быть использован для обновления и удаления данных, копирования записей в другие таблицы и выполнения многих других операций. Ниже мы рассмотрим операторы UPDATE, DELETE и INSERT, используемые для решения некоторых из этих задач.

2.2.1. Оператор UPDATE

Для изменения значений в одной или нескольких колонках таблицы применяется оператор UPDATE. Синтаксис этого оператора имеет вид:

UPDATE *таблица* SET *столбец1 = выражение1* [, *столбец2 = выражение2*] [,…]
[WHERE *условие_отбора*]

Выражение в предложении SET может быть константой или результатом вычислений. Например, для повышения зарплаты на 10% всем сотрудникам, получающим меньше 1000 грн., можно выполнить следующий запрос:

```
UPDATE Staff
SET Salary = Salary * 1.1
WHERE Salary < 1000
```

2.2.2. Оператор DELETE

Для удаления строк из таблиц следует использовать оператор DELETE, синтаксис которого имеет вид:

DELETE
FROM *таблица*
[WHERE *условие_отбора*]

Внимание! Предложение WHERE не является обязательным, но если вы забудете его включить, из таблицы будут удалены все записи.

Например, для удаления из таблицы всех завершенных договоров, можно выполнить следующий запрос:

```
DELETE
FROM Lease
WHERE RentFinish IS NOT NULL
```

Отметим, что полезно использовать оператор SELECT с тем же синтаксисом, что и оператор DELETE, чтобы проверить, какие именно записи будут удалены, прежде чем действительно их удалять. Ниже показан оператор SELECT для приведенного выше запроса на удаление данных:

```
SELECT *
FROM Lease
WHERE RentFinish IS NOT NULL
```

Можно использовать в предложении WHERE более сложный критерий для отбора тех записей, которые должны быть удалены. Предположим, нам нужно удалить из списка клиентов тех из них, кто не имеет действующий договор. Сначала для этого следует выполнить следующий SELECT, чтобы определить, что именно мы удаляем:

```
SELECT Client.*
FROM Client
WHERE ClientNo NOT IN
        (SELECT LeaseClientNo
         FROM Lease
         WHERE RentFinish IS NULL)
```

а затем заменить оператор SELECT на оператор DELETE:

```
DELETE
FROM Client
WHERE ClientNo NOT IN
        (SELECT LeaseClientNo
         FROM Lease
         WHERE RentFinish IS NULL)
```

2.2.3. Оператор INSERT

Для добавления записей в таблицы следует использовать оператор INSERT, синтаксис которого имеет вид:

INSERT [INTO] *таблица*
  ([*список_столбцов*]
    { VALUES ({ DEFAULT | NULL | *выражение* }
    } [, …]
)

Например, для добавления нового клиента в таблицу Client можно использовать следующий запрос:

INSERT INTO Client (ClientNo, LNmae, FName, TelNo, PrefType, MaxRent)
    VALUES (‘C0098’, ‘Петренко’, ‘Семен’, 7171247, ‘квартира’, 800)

2.3. Модификация метаданных

Существует несколько операторов SQL для управления метаданными, используемых для создания, изменения или удаления баз данных и содержащихся в них объектов (таблиц, представлений и др.). Мы рассмотрим некоторые из них: CREATE TABLE, ALTER TABLE и DROP.

2.3.1. Оператор CREATE TABLE

Для создания новой таблицы необходимо использовать оператор CREATE TABLE, синтаксис которого имеет вид:

CREATE TABLE *таблица*
  (*столбец1 тип1* [(*длина1*)][CONSTRAINT *ограничение_на_столбец1*]
  [, *столбец2 тип2* [(*длина2*)][CONSTRAINT *ограничение_на_столбец2*] [, ...]]
[CONSTRAINT *ограничение_на_таблицу1* [, *ограничение_на_таблицу2* [, ...]]])

В этом операторе следует указать имя поля, тип данных для него (тип данных должен поддерживаться данной СУБД), длину (для некоторых типов полей) и, если нужно, серверные ограничения (с применением

ключевого слова CONSTRAINT). Например, следующий запрос создает таблицу с именем Simple с четырьмя колонками — LastName, FirstName, EMail и HomePage:

```
CREATE TABLE Simple
 ( FirstName varchar(50) NOT NULL,
   LastName varchar(50) NOT NULL,
   EMail varchar(50),
  HomePage varchar(255)
 )
```

Мы можем расширить эту таблицу добавлением поля PersonID, которое будет использовано как первичный ключ:

```
CREATE TABLE Simple
 ( PersonID Integer NOT NULL PRIMARY KEY,
    FirstName varchar(50) NOT NULL,
    LastName varchar(50) NOT NULL,
    EMail varchar(50),
    HomePage varchar(255)
  )
```

и указать, что комбинация полей LastName и FirstName должна быть уникальна:

```
CREATE TABLE Simple
 (  PersonID Integer NOT NULL PRIMARY KEY,
     FirstName varchar(50) NOT NULL,
     LastName varchar(50) NOT NULL,
     EMail varchar(50),
     HomePage varchar(255),
 CONSTRAINT SimpleConstraint UNIQUE (FirstName, LastName)
 )
```

Используя предложение SELECT и ключевое слово INTO, мы можем заполнять новые таблицы значениями из существующих таблиц, отбирая для них данные по условию, указанному в предложении WHERE. Например:

```
SELECT *
INTO NewLease
FROM Lease
```

```
WHERE RentStart > Date('01/01/09') AND RentFinish IS NULL
```

Этот запрос создаст новую таблицу NewLease и заполнит ее данными о действующих договорах аренды, с 1 января 2009 года.

Замечание. При использовании в операторах SQL даты или времени, а также полей, содержащих такие данные, следует уточнить синтаксис таких предложений в документации из комплекта поставки используемой СУБД.

2.3.2. Оператор ALTER TABLE

Для изменения структуры существующей таблицы можно использовать оператор ALTER TABLE. Применяя его, можно добавить или удалить поле или серверное ограничение. Существует четыре разновидности оператора ALTER TABLE.

Первая разновидность этого оператора используется для добавления колонки к таблице, и ее синтаксис имеет вид:

ALTER TABLE *таблица* ADD [COLUMN] *столбец тип_данных* [(*длина*)]
[CONSTRAINT *ограничение_на_столбец*]

В запросах такого вида определяется имя таблицы, имя нового поля, его тип данных и, если нужно, размер. Помимо этого можно указать серверное ограничение, связанное с данным полем. Например, для добавления поля Phone к таблице Simple, созданной ранее, можно выполнить следующий запрос:

```
ALTER TABLE Simple ADD Phone varchar(30)
```

Вторая разновидность оператора ALTER TABLE применяется для добавления серверных ограничений к таблице, а ее синтаксис имеет вид:

ALTER TABLE *таблица* ADD CONSTRAINT *ограничение_на_таблицу*

Такие запросы позволяют только добавлять индексы, позволяющие использовать соответствующие поля в качестве первичных или внешних ключей.

Третья разновидность предложения ALTER TABLE применяется для удаления поля из таблицы:

ALTER TABLE *таблица* DROP [COLUMN] *столбец*

Ключевое слово COLUMN использовать не обязательно. Например:

ALTER TABLE Simple DROP Phone

Обратите внимание на то, что для удаления проиндексированных полей следует сначала удалить индекс. Это можно сделать с помощью четвертой разновидности предложения ALTER TABLE:

ALTER TABLE *таблица* DROP CONSTRAINT *индекс*

Ниже приведен пример такого запроса:

ALTER TABLE Simple DROP CONSTRAINT PrimaryKey

2.3.3. Оператор DROP

Для удаления таблиц или индексов можно использовать оператор DROP, имеющий две разновидности. Первая из них применяется для удаления таблицы из базы данных:

DROP TABLE *таблица*

Вторая разновидность используется для удаления индекса:

DROP INDEX *индекс* ON *таблица*

2.4. Другие операторы SQL

Как было отмечено ранее, существует около 40 операторов SQL. Мы рассмотрели большинство из них. Некоторые из не рассмотренных нами операторов перечислены ниже:

- *операторы CR*EATE, такие как CREATE DATABASE, CREATE VIEW и CREATE TRIGGER;
- *операторы ALTER*, такие как ALTER DATABASE, ALTER VIEW и ALTER TRIGGER;
- *операторы DROP*, такие как DROP DATABASE, DROP VIEW и DROP TRIGGER;
- BEGIN TRANSACTION, COMMIT TRANSACTION и ROLLBACK TRANSACTION для выполнения группы нескольких операторов как единой логической группы;
- DECLARE CURSOR, OPEN и FETCH для работы с курсорами;
- GRAND и REVOKE для добавления или удаления прав на использование объектов базы данных, а также CREATE USER, ALTER USER, DROP USER, CREATE GROUP, ALTER GROUP и DROP GROUP для управления списком пользователей и групп пользователей.

3. Описание базы данных DreamHome

Для демонстрации принципов организации баз данных в курсе «Базы данных и информационные системы» используется модельная база данных DreamHome. Далее приводится описание структуры этой базы.

В модельной базе данных DreamHome хранится информация вымышленной компании «DreamHome», занимающейся сдачей объектов недвижимости в аренду.

Схематически структура базы данных представлена концептуальной схемой в виде ER-диаграммы на рисунке 3.1.

В базе представлены следующие объекты и их связи.

Организационная структура компании – филиалы (таблица **Branch**).

О каждом филиале должны храниться такие данные, как номер филиала, адрес (включая город, улицу и номер дома, почтовый индекс), табельный номер менеджера (управляющего) филиала. В каждом филиале есть один и только один менеджер из числа сотрудников этого филиала. Работой всей компании в целом руководит директор, который является сотрудником одного из филиалов.

Менеджер филиала руководит работой всего филиала, в котором, кроме него, работают также инспектора и их помощники-ассистенты.

Персонал компании (таблица Staff).

О каждом сотруднике должны храниться такие данные, как табельный номер, имя (включая имя и фамилию), должность, пол, дата рождения (Date Of Birth) и имя руководителя (если он имеется). Сотрудники компании, занимающие должность инспектора, могут руководить работой нескольких ассистентов (количество которых в любой момент времени не может превышать максимального значения, равного 10).

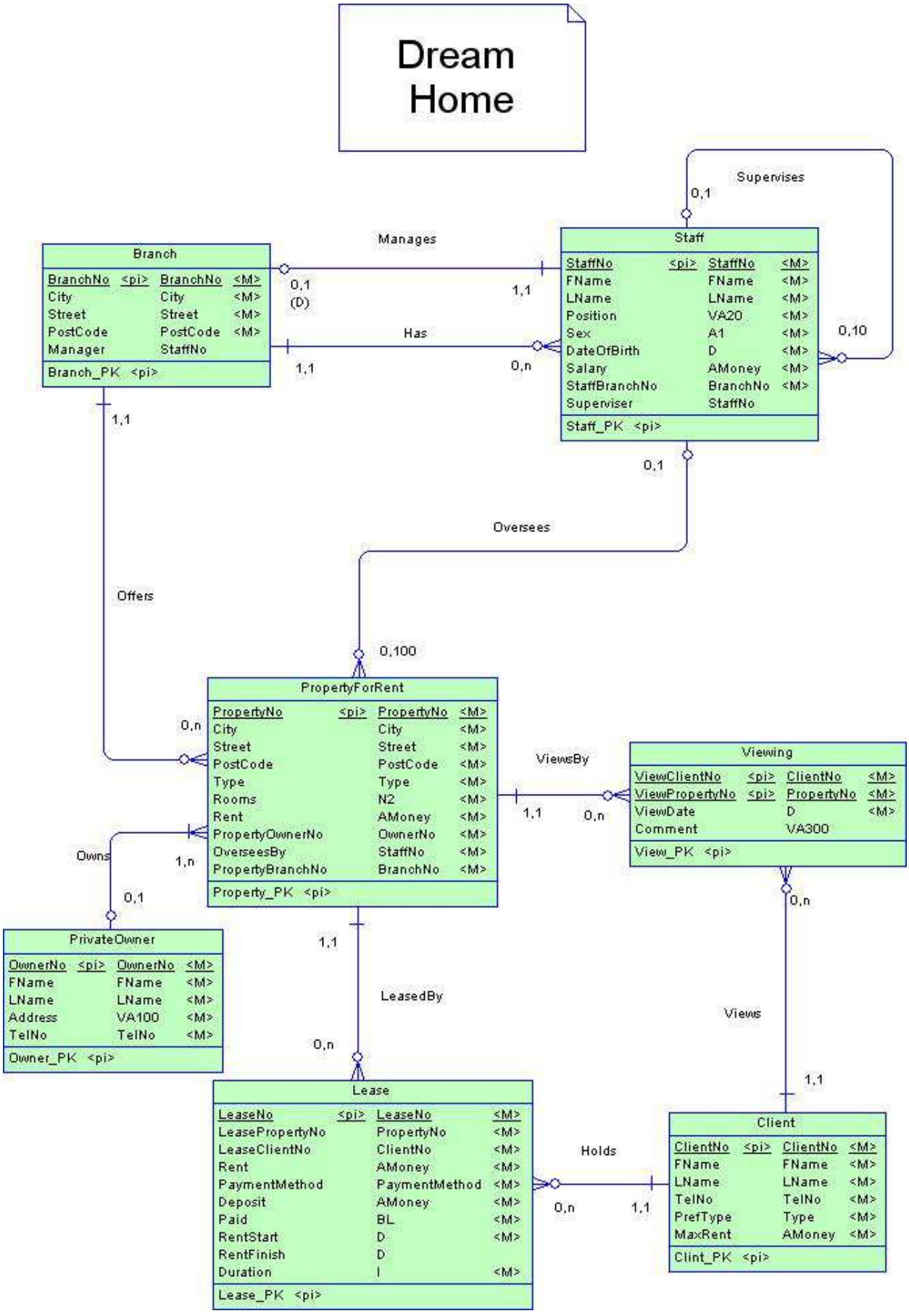

Рис. 3.1. Концептуальная схема базы данных DreamHome

Объекты недвижимости, предназначенные для сдачи в аренду (таблица PropertyForRent)

О каждом объекте недвижимости, предназначенном для сдачи в аренду, должны храниться такие данные, как номер объекта недвижимости, адрес (улица, город и почтовый индекс), тип объекта недвижимости, количество комнат, ежемесячная арендная плата и сведения о владельце. Ставка ежемесячной арендной платы для каждого объекта недвижимости пересматривается один раз в год. Основную часть объектов недвижимости, сдаваемых в аренду компанией DreamHome, составляют квартиры. Однако могут предлагаться в аренду дома, офисы, дачи и другие типы объектов. Управление объектом недвижимости, который сдается в аренду или требуется для аренды, возлагается на одного из сотрудников компании. Любой сотрудник компании может управлять одновременно несколькими объектами недвижимости, количество которых не может превышать 100. Каждый объект недвижимости предлагается в аренду одним из филиалов.

Владельцы объектов недвижимости (таблица PrivateOwner**)**

Владельцы объектов недвижимости подразделяются на два типа: владельцы частной собственности и владельцы деловых предприятий. О каждом владельце частной собственности хранятся такие данные, как номер владельца, имя (включая имя и фамилию), адрес и номер телефона.

Клиенты (таблица Client)

При регистрации будущего клиента компании DreamHome в базу данных вносятся такие сведения, как номер клиента, имя (включая имя и фамилию), номер телефона, а также некоторая информация об искомом объекте недвижимости, включая предпочитаемый тип объекта

недвижимости и максимальную арендную плату, которую клиент готов платить.

Осмотр объектов недвижимости (таблица Viewing)

Клиент может потребовать, чтобы ему разрешили осмотреть объект недвижимости (в том числе повторно). По результатам каждого осмотра в базу данных вносятся такие сведения, как номер клиента, номер объекта недвижимости, дата осмотра клиентом объекта недвижимости, а также все комментарии, сделанные клиентом по поводу пригодности для него этого объекта недвижимости. Клиент не может осматривать один и тот же объект недвижимости в определенную дату больше одного раза.

Договора аренды (таблица Lease)

После того как клиент находит подходящий для него объект недвижимости, заключается договор аренды. О каждом договоре аренды хранится такая информация, как номер договора аренды, номер клиента, номер объекта недвижимости, ежемесячная арендная плата, метод оплаты, залог (который составляет удвоенное значение ежемесячной арендной платы), отметка о внесении залога, дата начала и окончания периода аренды, а также продолжительность договора аренды в месяцах. Номер каждого договора аренды является уникальным во всех отделениях компании DreamHome. Клиент может заключить договор аренды любого объекта недвижимости на срок, который должен составлять не меньше трех месяцев и не превышать одного года. Дата окончания договора аренды остается незаполненной в течение всего времени действия договора (договор может быть продлен, увеличивая продолжительность договора). Поле заполняется только в момент фактического окончания договора.

4. Задания по лабораторным работам

В процессе изучения курса студенты должны выполнить 4 лабораторных задания. Список заданий лабораторных работ приведен ниже.

По каждой лабораторной работе студент подготавливает отчет, который содержит:

- Условия задания.
- Решение задания с комментариями
- Выводы.

4.1. Задание к лабораторной работе № 1 (Создание БД)

1) Создать базу данных DreamHome по ее концептуальной модели (см. рис. 3.1.)
2) Заполнить базу данных с учетом следующих требований:
    a) Создать данные о 5-ти отделениях компании DreamHome.
    b) Одно из этих отделений – большое, в нем работают:
        - Директор
        - Менеджер
        - 2 инспектора
        - У каждого инспектора 3-4 ассистента
    c) Остальные отделения – не большие: менеджер, 1-2 инспектора, 3-4 ассистента.
    d) Создать данные о 8 владельцах недвижимости, которые владеют 14-ю объектами недвижимости.
    e) Создать данные о 12 – ти клиентах.
    f) Создать данные об осмотрах, всего их – 23.
    g) Создать данные о договорах – всего 9, из них 2 – завершены.

h) Закрепить объекты за сотрудниками (не равномерно, есть сотрудник, за которым закреплено 5 объектов).
i) Закрепить договора за объектами и клиентами – неравномерно: есть клиенты, у которых по 3 договора, есть объекты, по которым по 3 договора.
j) В БД должны быть:
    - Инспектор без ассистентов
    - Клиент, у которого все договора завершены
    - Объекты без договоров
    - Объекты, условия аренды которых подходят одновременно нескольким клиентам.

3) Написать или сгенерировать скрипт для создания и заполнения базы

4.2. Задание к лабораторной работе № 2 (11 запросов)

На заполненной базе данных реализовать следующие запросы:

1) Сколько всего отделений у компании DreamHome.
2) Сколько всего сотрудников у компании DreamHome.
3) Сколько всего объектов недвижимости предлагает в аренду компания DreamHome.
4) Сколько всего владельцев объектов недвижимости работают с компанией DreamHome
5) Сколько всего клиентов зарегистрировано в компании DreamHome
6) Сколько всего договоров заключено в компании DreamHome в 2006-2007 годах
7) Сколько всего осмотров выполнено сотрудниками компании DreamHome в 2006-2007 годах
8) Сколько сотрудников в каждом отделении компании DreamHome

9) Сколько всего объектов предлагает в аренду каждое отделение компании DreamHome
10) Сколько ассистентов находится в подчинении каждого инспектора компании DreamHome
11) Индивидуальный запрос.

4.3. Задание к лабораторной работе № 3 (Все о…)

На заполненной базе данных реализовать следующие запросы:

1) Все об отделениях:
    a. Номер,
    b. Полный адрес,
    c. ФИО менеджера,
    d. Количество сотрудников.

2) Все о сотрудниках:
    a. Табельный номер,
    b. ФИО,
    c. Должность,
    d. Номер и полный адрес отделения, где работает сотрудник
    e. Пол,
    f. Дата рождения,
    g. Оклад
    h. Если сотрудник - ассистент, то ФИО его руководителя (инспектора)
    i. Количество объектов, которые он ведет

3) Все об объектах недвижимости:
    a. Регистрационный номер
    b. Полный адрес
    c. Тип
    d. Количество комнат
    e. Сумма арендной платы, назначенная владельцем
    f. ФИО владельца
    g. ФИО сотрудника, который ведет объект

h. Номер и полный адрес отделения, которое предлагает объект в аренду
i. Сколько раз объект сдавался в аренду
j. Признак, арендует ли в данный момент этот объект какой-нибудь клиент

4) Все о клиентах:

a. Регистрационный номер
b. ФИО
c. Номер телефона
d. Тип объекта недвижимости, который он ищет
e. Максимальная сумма арендной платы, которую он может заплатить
f. Признак, арендует ли он в данный момент объект недвижимости
g. Срок, в течение которого он обслуживается в компании

5) Все о владельцах объектов недвижимости:

e. Регистрационный номер,
f. ФИО,
g. Адрес
h. Номер телефона
i. Количество объектов недвижимости, которыми он владеет

6) Выдать данные по регистрационной карточке договора самостоятельно.

7) Создать формы для просмотра каждой карточки.

8) Добавить на карточки фотографии сотрудников и объектов недвижимости.

4.4. Задание к лабораторной работе № 4 (Обновление данных)

Написать запросы на обновление информации в базе данных:

1. Удалить сотрудников-ассистентов, которые в 2005 году не организовали ни одного осмотра по объектам, которые они ведут.

2. Увеличить зарплату на 20 % тем сотрудникам, у которых выполнено максимальное количество осмотров по объектам, которые они ведут.

3. Добавить договора для тех клиентов, для которых есть ровно один подходящий им объект. Номер договора – следующий по базе данных, сумма договора равна сумме, предлагаемой клиентом, плюс 20 %. Проверить, что клиент уже не арендует какой-либо объект, а подходящие объекты выбирать из числа тех, которые не находятся в аренде.

4. Повысить зарплату на 10 % самому старому сотруднику фирмы, при условии, что он ведет хотя бы один объект.

5. Удалить те объекты недвижимости, по которым не было (и нет) ни одного договора.

Список рекомендованной литературы

1. Коннолли, К.Бегг. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание. – М.: Издательский дом «Вильямс», 2003 г.
2. М. Грабер, Введение в SQL. - М., Вр-во «Лори», 1996.
3. М.Грабер. Справочное руководство по SQL. - М., Из-во "ЛОРИ", 1998 г. (ANSI X3H2, Комитет стандартов баз данных).

Навчальне видання

Методичний посібник з курсу
«Бази даних та інформаційні системи»

Петрушина Тетяна Иванівна,

Коломієць Дмитро Васильович

Видано в авторській редакції

Підп. до друку 06.09.2010. Формат 60х84/8.
Гарн. Таймс. Тираж 56 прим.

Редакційно-видавничий Центр
Одеського національного університету
імені І.І. Мечникова,
65082, м. Одеса, вул. Єлісаветинська, 12, Україна
Тел.: (048) 723 28 39