Оглавление

Введение

Одной из составляющих успешной экономической деятельности любого хозяйствующего субъекта является эффективное и профессиональное информационное обеспечение. Эта составляющая во многом способствует результативности и достижению целевых показателей деятельности, включая получение прибыли. Эти задачи являются комплексными и зависящими от многих факторов. Среди них важное место занимает квалификация сотрудников, имеющих дело с организацией, обработкой, анализом информации предметных областей, имеющих отношение к сфере деятельности предприятия, коммерческой фирмы или учреждения. Эффективному решению обозначенных производственных задач способствуют информационные системы (ИС), спроектированные и реализованные на основе баз данных (БД).

Развитию теории информации способствовало рождение и бурное развитие теории баз данных. Благодаря этой теории, человечество перешло от эпохи файловых систем к системам на основе особым образом структурированных файлов, организация и автоматизированная обработка которых позволила гораздо более технологично управлять данными и обрабатывать их.

Сегодня информационные системы строятся на основе разных моделей данных. Весьма большая часть из них построена на основе реляционной модели. Как известно, все началось в 1970 году. Сотрудник исследовательской лаборатории *IBM* Э. Ф. Кодд (E. F. Codd) опубликовал статью, в которой предложил новую концепцию организации данных. В последующие годы появилось множество экспериментальных систем управления базами данных (СУБД), основанных на идее Кодда. Так, в рамках проекта *System R* (1976 год, Сан-Хосе, Калифорния) был предложен язык структурированных запросов *SQL — Structured Query Language*, который в дальнейшем стал стандартом языков запросов для всех реляционных СУБД. Через некоторое время модель Кодда была мощно усилена предложением П. Чена (P. Cheung) — моделью «сущность-связь» (*ER-модель*), которая сразу была признана

компьютерным сообществом. Она долгие годы считалась самой распространенной технологией проектирования баз данных и основой методологии логического проектирования реляционных БД.

Известно, что в состав любой информационной системы структурно входят *программные* и *языковые средства*.

Программные средства (ПС) ИС — это, собственно, компьютерные программы, которые должен выполнить компьютер с целью реализации заложенных в них алгоритмов решения конкретных задач обработки данных.

Неотъемлемой частью ИС являются *языковые средства* (ЯС), призванные для описания структур данных, обеспечивающие доступ к ним и действий над ними. Один из популярных представителей языковых средств общения с базами данных — язык SQL. Он стал воплощением всех лучших идей реляционной модели. В первую очередь это относится к реализации механизмов обработки данных. Разработчикам СУБД удалось эффективно встроить в программное обеспечение математические аппараты реляционной алгебры и реляционного исчисления. В результате язык оказался лаконичным, выразительным, вполне доступным для изучения.

Благодаря своим качествам язык SQL на долгие годы стал стандартом среди языковых средств обработки реляционных баз данных. Как стандарт он поддерживается всеми ведущими мировыми производителями программного обеспечения в сфере технологий баз данных. Одно из несомненных достоинств SQL — обеспечение независимости разрабатываемых прикладных систем от типа СУБД. Другое не менее значимое преимущество — обеспечение унификации инструментальных средств разработки приложений баз данных.

Заметим, что этот язык не стоит на месте. Жизнь ставит перед программистами новые задачи, которые, в свою очередь, привносят новые идеи в SQL, а их осмысление приводит к изменениям и дополнениям в стандарт этого языка.

В связи с языком SQL начинающему программисту следует дать один важный совет. При первом знакомстве SQL может показаться простым и незатейливым языком.

Мягко говоря, это заблуждение. В реальности — это богатый и сложный язык. Он позволяет изящно, лаконично, выразительно записывать решения самых разных задач — от простых учебных до нетривиальных производственных задач. Поэтому начинающему программисту надо запастись терпением, чтобы постепенно погрузиться в мир идей и возможностей SQL и почувствовать всю мощь этого инструмента при решении реальных задач на ЭВМ.

Для кого предназначена эта книга

Проблемы цифровой трансформации образовательной деятельности — серьезный вызов нашего времени. В последние годы эти проблемы заметно обострились, в том числе в связи с известной пандемией коронавируса COVID-19 2019–2022 годов. Сегодня идет поиск решений, направленных на эффективную поддержку образовательной сферы, которые позволили бы в равной степени поддержать новые очные и удаленные формы учебного процесса, сделать его современным, интересным, результативным.

Идея данной книги появилась совсем недавно — в связи с ограничительными мерами в мире из-за пандемии коронавируса. Оказалось, что удаленные формы учебной работы существенно увеличили нагрузку на всех участников образовательного процесса, проявили сложности психологического характера, создали определенные проблемы общения и другие неприятности. Возникла идея создать систему для организации лабораторных занятий и управления ими с использованием элементов цифровизации образовательного процесса и технологий удаленного доступа к данным. Предполагалось, что система должна одинаково хорошо работать как в очном варианте проведения занятий, так и в удаленном. Одновременно с реализацией компьютерной системы понадобилось подготовить лабораторные работы по практикуму «Базы данных», которые образовали бы основу Хранилища компьютерных заданий для студентов по указанной дисциплине. Очевидно, что электронное Хранилище постоянно находится в развитии и способствует накоплению методического материала и опыта по преподаваемой дисциплине.

В связи с вышесказанным, книга предназначена для преподавателей и студентов вузов и техникумов, изучающих дисциплину «Базы данных». Такая дисциплина обязательно должна включать разделы, посвященные языку SQL.

С точки зрения теории реляционных баз данных выделяют четыре подмножества языка SQL:

– *DCL* (Data Control Language);
– *DDL* (Data Definition Language);
– *DML* (Data Manipulation Language);
– *TCL* (Transaction Control Language).

Очевидно, в различных учебных курсах эти подмножества рассматриваются с разной степенью детализации. Однако с точки зрения практики использования SQL специалистами наиболее востребовано подмножество *DML*. Действительно, наиболее часто используемые программистами в работе средства этого языка — команды для решения повседневных задач, которые возникают на производстве. Эта часть языка не оставит равнодушным читателя, который заинтересовался базами данных как современной формой организации данных и инструментами их обработки.

Как известно, самый лучший способ подготовки — практическое решение задач на ЭВМ. В книгу включена подборка лабораторных работ, предназначенных для реализации на компьютере. Структурно каждая лабораторная работа состоит из заданий, объединенных одной темой. Лабораторные работы сгруппированы по правилу «от простого — к сложному». Предполагается, что читателю, работающему с этой книгой, будут доступны выгрузки баз данных, использованные при подготовке авторских лабораторных работ. Особенность предложенных для решения задач баз данных состоит в том, что их предметные области просты и интуитивно понятны, структуры таблиц легко читаются, а сами таблицы не содержат больших объемов исходных данных.

Структура книги

Книга состоит из трех частей.

В *первой части* «*Язык SQL и СУБД MySQL. Некоторые предварительные сведения*» рассматривается материал, по-

священный выборки данных из таблиц, описывается механизм работы команды *SELECT*, особенности взаимодействия разделов этой команды. Отдельное внимание уделено предикатам, применяемым в команде *SELECT*, а также встроенным функциям СУБД *MySQL* и агрегатным функциям *SQL*. Функции группированы по их назначению: обработка числовых данных, данных типа «дата» и «время», символьных данных. С практической точки зрения особую ценность имеют вложенные запросы. В связи с этим рассматриваются разновидности таких запросов, понятия «предикат», «модифицированная операция сравнения», «коррелированный подзапрос», «некоррелированный подзапрос», «квантифицированный предикатный подзапрос» и другие. Приводятся примеры применения кванторов *ANY*(), *ALL*(), предиката *IN* при решении некоторых простых задач на данную тему.

Во *второй части* — «*Лабораторный практикум по решению задач на ЭВМ*» — собраны задания, предназначенные для их решения на компьютере. Лабораторные работы включают несколько заданий, имеют тематическую группировку и следуют по возрастанию сложности. По отдельным темам лабораторных работ даются дополнительные комментарии, например, по темам «Вложенные SQL-запросы», «Операция *JOIN* команды *SELECT*», «Оператор *UNION* языка SQL», «Квантор *Exists*()». Предполагается, что при выполнении заданий пользователь будет использовать авторские выгрузки баз данных. Почти для всех заданий даны ответы. При разработке заданий автором за основу была взята СУБД *MySQL*. Решения заданий проверены с помощью web-сервера *Denwer*. Последний раздел этой части — «Обсуждение примеров» — посвящен обсуждению некоторых типичных ошибок и рассмотрению вариантов решения отдельных задач.

Одна из идей книги состоит в том, что она предлагает два совершенно разных варианта практической работы на ЭВМ. Первый вариант — самостоятельная работа читателя на своем (домашнем) компьютере. Второй вариант — организованное групповое проведение занятий с использованием информационной системы *SQL-Start*, возможности которой кратко описаны в *третьей части* книги. Система

предназначена для проведения лабораторных занятий по дисциплине «Базы данных» при изучении языка SQL. Описание системы состоит из двух частей — описания *Рабочего места преподавателя* и *Рабочего места студента* [1, 2].

Как пользоваться этой книгой

В первой части книги приведены некоторые сведения теоретического и справочного характера по языку *SQL* и реляционной СУБД *MySQL*. Во время работы с лабораторным практикумом этот материал находится всегда «под рукой» и может оказаться полезным как справочный. Кроме того, эта часть книги может оказаться нужной для начинающего программиста, который заинтересовался темой обработки информации, организованной в виде базы данных.

Наиболее практически значимая часть этой книги — вторая — «Лабораторный практикум по решению задач на ЭВМ». Для работы с ней читателю понадобится локальный web-сервер типа *Denwer* и выгрузки таблиц базы данных, подготовленные автором. Для использования базы данных ее нужно предварительно импортировать на свой компьютер. Все лабораторные работы имеют тематическую направленность. Предполагается, что читатель знаком с основами реляционных баз данных и понимает идеи и назначение языка SQL. Несмотря на эти знания, в первой части книги имеются полезные сведения, имеющие прямое отношение к изучаемому материалу. Первая особенность лабораторного практикума состоит в том, что для каждого задания лабораторной работы дан правильный ответ. Вторая — в том, что читатель может написать, очевидно, различные решения одной и той же задачи. Однако ему предлагается учесть требования (указания), предъявляемые к будущему решению задачи. Их выполнение важно с методической точки зрения, так как при отработке очередной лабораторной работы важно учесть ее тематическую направленность, необходимость в изучении определенных инструментов языка *SQL* и получении определенного способа решения задачи.

При работе с лабораторными заданиями читателю потребуется один раз импортировать предлагаемую базу данных и всякий раз проделывать следующие действия:

– ознакомиться с лабораторным заданием;

– изучить предметную область на основе сведений, представленных в таблицах базы данных;

– приступить к решению задачи;

– попытаться получить решение лабораторного задания с учетом предъявленных к нему требований (указаний).

После лабораторных работ дан полезный материал под названием «Обсуждение примеров». По своему объему он невелик, однако приведенные в нем примеры заставляют задуматься начинающего программиста над многообразием возможностей языка *SQL* и существованием различных подходов и приемов к решению одной и той же задачи. В этом же материале показаны примеры некоторых типичных ошибок, которые могут возникнуть при решении самых разных задач.

Замечание. Некоторые версии СУБД типа *MySQL* требуют записи имен таблиц SQL-запросах *строчными буквами*.

Пример 1. Имя таблицы записано заглавными и строчными буквами.

```
select Kategor Категория, Tovar Товар, Date_P ДатаПоступления
from DataTable1
order by Категория,Товар
```

Пример 2. Имя таблицы записано строчными буквами.

```
select Kategor Категория, Tovar Товар, Date_P ДатаПоступления
from datatable1
order by Категория,Товар
```

Часть I
Язык SQL и СУБД MySQL. Некоторые предварительные сведения

1.1. Оператор SELECT языка SQL

SQL — типичный представитель так называемых *полных языков баз данных*. По этой причине он обязан включать средства для описания схем баз данных и их сопровождения, управления таблицами данных, поддержку целостности баз данных, реализацию правил ограничения целостности и ряд других возможностей. Однако большинство пользователей, работающих с базами данных, рассматривают этот инструмент как язык запросов, который позволяет формулировать произвольно сложные и декларативно точные запросы к базам данных. Решение повседневных производственных задач с использованием средств выборки данных — наиболее распространенный во всем мире вид деятельности программистов, имеющих дело с реальными базами данных. Вопросы теории и практики баз данных рассмотрены во многих книгах и публикациях, например, в книгах [3–5].

Один из важных практических аспектов *SQL* связан с понятием *скалярного выражения*. Предполагается, что оно вырабатывает результат некоторого типа, который специфицирован в стандарте языка *SQL*. Такие выражения образуют основу *SQL*, поскольку элементы списков выборки, условия и некоторые другие конструкции языка базируются именно на таких выражениях.

Назовем наиболее важных представителей скалярных выражений в *SQL*:

1) числовые выражения;
2) выражения из строк символов;
3) выражения из значений даты-времени;
4) выражения из значений временных интервалов;
5) булевские выражения.

Как известно, для выборки данных предназначен специальный оператор *SELECT*. Результатом его выполнения является таблица из набора строк одинаковой структуры.

Синтаксис этого оператора в его стандартной форме имеет вид [3]:

```
<Оператор Select> ::=
SELECT [ ALL | DISTINCT] select_item_commalist
        FROM table_reference_commalist
    [ WHERE conditional_expression ]
    [ GROUP BY column_name_commalist ]
    [ HAVING conditional_expression ]
    [ ORDER BY order_item_commalist ]
```

Заметим, что оператор изображен в нотации *БНФ* (Бэкуса — Наура форма).

Напомним, что в рамках данной нотации:

<Понятие> — метапеременная (определяемое понятие) *БНФ*;

[Конструкция] — возможное отсутствие конструкции;

::= — символ метаприсваивания;

| — черта альтернативы (выбор).

Обсудим *семантику* данного оператора. Выполнение запроса состоит из нескольких шагов, соответствующих разделам оператора [3]. Фактически, данный оператор определяет некий конвейер обработки данных. В связи с этим весьма важно понимание работы этого конвейера, так как в противном случае это приводит не только к написанию неверных команд, приводящих к неверному результату, но и к непониманию существа ошибок.

Итак, на первом шаге выполняется раздел FROM. Список select_item_commalist — перечисление имен таблиц X_1, X_2, ..., X_n. В результате его выполнения порождается новая таблица (например, T). С точки зрения теории реляционных баз данных она представляет собой расширенное декартово произведение таблиц X_1, X_2, ..., X_n. Если в разделе FROM указана одна таблица, то она же и будет результатом этого раздела. Кроме стандартного способа ссылки на столбцы результирующей таблицы, имеется альтернативный способ, основанный на использовании так называемых квалифицированных имен столбцов или псевдонимов. Способ основан на использовании псевдонима таблицы из раздела FROM.

К примеру, если с некоторой таблицей X связан псевдоним Y, то на некоторый столбец Q таблицы X можно сослаться с помощью псевдонима Y. Q.

На втором шаге выполняется раздел WHERE. Конструкция conditional_expression — условное выражение раздела, которое применяется к каждой строке таблицы T. Результат такой выборки вновь является таблица. Обозначим ее через T_1. В эту таблицу включаются те строки таблицы T, для которых условное выражение conditional_expression дает значение true. Заметим, что заголовки таблиц T и T_1 должны совпадать. Если этот раздел отсутствует, то он понимается как WHERE true, т. е. таблица T_1 включает те же строки, что и исходная для нее таблица T.

Если в операторе выборки имеется раздел GROUP BY, то он выполняется на третьем шаге. Каждый элемент списка имен столбцов column_name_commalist должен быть одним из имен столбцов ранее полученной таблицы T_1. В результате выполнения раздела GROUP BY порождается новая сгруппированная таблица — T_2. В ней строки таблицы T_1 размещены в минимальное число групп. Во всех строках одной такой группы значения столбцов, указанных в списке column_name_commalist, одинаковы. Сгруппированные таблицы не могут рассматриваться в качестве окончательного результата оператора выборки. Они существуют только на концептуальном уровне на стадии выполнения запроса, содержащего раздел GROUP BY.

Если в операторе выборки присутствует раздел HAVING, то на следующем шаге будет выполняться он. Конструкция conditional_expression — условное выражение раздела, применяемое к каждой группе строк таблицы T_2. Обозначим результат этой обработки данных через T_3. Эта таблица будет содержать только те группы строк таблицы T_2, для которых результатом вычисления условного выражения conditional_expression является значение true. Условное выражение conditional_expression раздела HAVING строится по тем же правилам, что и обычные условные выражения. Однако имеет место важное отличие этого условия от других условий: выражение conditional_expression применяется не к отдельным строкам, а к *группам строк*. По этой причине

предикаты, из которых строится условное выражение conditional_expression раздела HAVING, должны быть *предикатами на всю группу строк в целом*. В них могут использоваться имена столбцов группировки (или инварианты групп) и специальные функции — так называемые *агрегатные функции*, примененные к другим столбцам.

Напомним, что *предикат — основа логического выражения*. Он позволяет специфицировать условие, результатом вычисления которого могут быть значения true, false или unknown.

Здесь важно обсудить ситуацию, когда в запросе имеется раздел HAVING, но нет раздела GROUP BY. Тогда таблица T_1 рассматривается как сгруппированная таблица, состоящая из одной группы строк и без столбцов группирования. В этом случае логическое выражение conditional_expression раздела HAVING может состоять только из предикатов с агрегатными функциями. Заметим, что результат вычисления этого раздела может оказаться пустым.

В операторе выборки может присутствовать раздел GROUP BY, но отсутствовать раздел HAVING. Эта ситуация трактуется как наличие раздела в виде HAVING true, т. е. таблица T_3 содержит только те группы строк, которые содержаться в таблице T_2.

После выполнения раздела (в случае отсутствия разделов GROUP BY и HAVING — случай (a)) явно или неявно, или неявно заданного раздела HAVING случай (b) выполняется раздел SELECT. При выполнении этого раздела на основе таблицы T_1 (случай (a)) или на основе сгруппированной таблицы T_3 (случай (b)) строится новая таблица T_4, содержащая столько строк, сколько строк или групп строк содержится в таблицах T_1 или T_3 соответственно. Число столбцов в таблице T_4 зависит от числа элементов в списке элементов выборки select_item_commalist и от вида элементов.

Выполнение раздела ORDER BY производится следующим образом. Сначала выбирается первый элемент списка сортировки order_item_commalist, и строки таблицы T_4 расставляются в порядке возрастания (*ASC*) или в порядке убывания (*DESC*) в соответствие со значениями выражения, содержащегося в данном элементе. Такие значения

вычисляются для каждой строки таблицы T_4. Далее выбирается второй элемент списка сортировки order_item_commalist, и в соответствии со значениями заданного в нем выражения и порядка сортировки расставляются строки, которые после перового шага сортировки уже образовали группы с одинаковым значением выражения перового элемента списка сортировки. Операция продолжается до обработки всего списка сортировки order_item_commalist. Результирующий отсортированный список и является окончательным результатом запроса.

Далее важно обратить внимание на *разновидности ссылок на таблицы* в разделе FROM команды Select [3]. *БНФ* этой конструкции имеет вид (упрощенный вариант):

```
table_reference ::= table_primary
table_primary ::= table_or_query_name
[[AS]correlation_name [(derived_column_list)]] |
derived_table [AS] correlation_name
[(derived_column_list)]
table_or_query_name ::= {table_name | query_name}
derived_table ::= (query_expression)
```

Классический случай ссылки на таблицу — указание имени так называемой базовой таблицы.

Базовая таблица — родовая структура модели *SQL*. Она обладает *заголовком* (имена столбцов) и *телом* (строки с данными, которые соответствуют заголовку). Другой вариант ссылки — derived_table — производная таблица, порождаемая запросом, заключенным в круглые скобки — (query_expression).

Конструкция correlation_name представляет собой *псевдоним*. Он дает возможность присвоить таблице на момент выполнения запроса некоторое временное имя. Можно считать, что при обработке раздела FROM выражение запроса вычисляется и сохраняется во временной таблице.

1.2. Предикаты раздела WHERE оператора Select

Обсудим логические выражения раздела WHERE оператора Select. Синтаксически логическое выражение раздела

WHERE определяется как булевское выражение. Основой логического выражения являются *предикаты*. Рассмотрим основные разновидности предикатов раздела WHERE.

1. *Предикат сравнения* (=, <, <=, >, >=, <>).

Предназначен для сравнения строк, чисел, битовых строк, величин типа «дата-время».

Пример:

```
WHERE Price* Quantity > 80000.00
```

2. *Предикат BETWEEN*.

Позволяет специфицировать условие вхождения некоторой величины в диапазон значений.

Пример:

```
WHERE Price*Quantity BETWEEN 50000.00 AND 80000.00
```

3. *Предикат NULL*.

Позволяет проверить, является ли некоторое значение неопределенным.

Примеры:

1) WHERE ProductId IS NULL

2) WHERE ProductId IS NOT NULL

4. *Предикат IN*.

Позволяет специфицировать условие вхождения некоторого значения в указанное множество значений.

Примеры:

1) WHERE ProductId IN (11, 12, 15, 19)

2) WHERE ProductId (DEPT_NO IN (11, 12, 15, 19))

Заметим, что первый из примеров эквивалентен следующей конструкции:

```
WHERE (ProductId = 11) OR (ProductId = 12) OR
      (ProductId = 15) OR (ProductId = 19)
```

5. *Предикат LIKE*.

Позволяет установить соответствие некоторого значения заданному шаблону значений.

Пример:

```
WHERE ProductName LIKE '%Виноград%'
```

6. *Предикат EXISTS*.

Значением условия EXISTS(query_expression) является true в том случае, когда мощность таблицы-результата выражения запроса больше нуля, и false — в противном случае.