3-е издание

Охватывает MySQL, Oracle, PostgreSQL и SQL Server

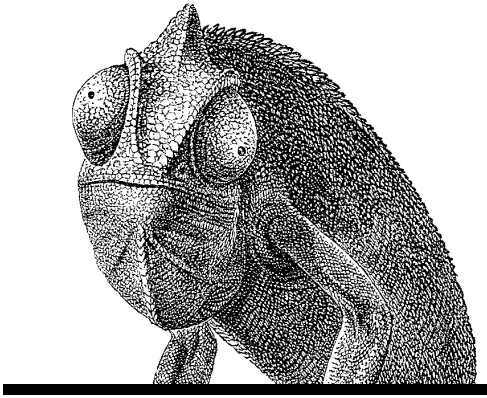
# SQL

## СПРАВОЧНИК

Символ

O'REILLY®

*Кевин Е. Кляйн,*
*Дэниэл Кляйн*
*и Брэнд Хант*



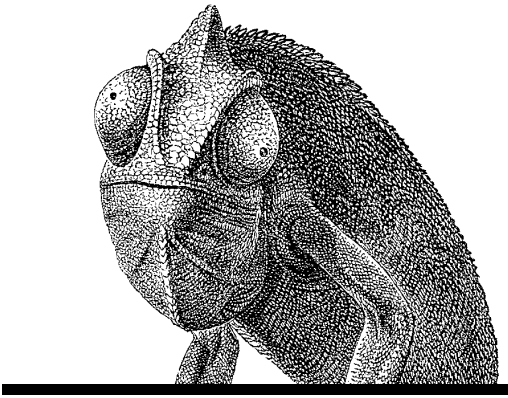
# SQL
# IN A NUTSHELL

*A Desktop Quick Reference*

Third edition

*Kevin E. Kline*
*with Daniel Klien & Brand Hunt*

O'REILLY®



# SQL
# СПРАВОЧНИК

Третье издание

*Кевин Е. Кляйн,*
*Дэниэл Кляйн и Брэнд Хант*

*Санкт-Петербург — Москва*
*2010*

Кевин Кляйн, Дэниэл Кляйн и Брэнд Хант

# SQL. Справочник, 3-е издание

Перевод Е. Демьянова, А. Слинкина


| | |
|---|---|
| Главный редактор | *А. Галунов* |
| Зав. редакцией | *Н. Макарова* |
| Выпускающий редактор | *П. Щеголев* |
| Научный редактор | *А. Рындин* |
| Редактор | *Ю. Бочина* |
| Корректор | *С. Минин* |
| Верстка | *Д. Орлова* |

*Кляйн К., Кляйн Д., Хант Б.*
SQL. Справочник, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2010. – 656 с., ил.
ISBN 978-5-93286-165-3



В третьем издании книги «SQL. Справочник» описываются все операторы SQL согласно последнему стандарту ANSI SQL2003, а также особенности реализации этих операторов в наиболее популярных СУБД: Microsoft SQL Server 2008, Oracle 11g, MySQL 5.1 и PostgreSQL 8.3. Издание содержит описание реляционных моделей данных, объяснение основных концепций реляционных СУБД, полное описание синтаксиса SQL, а также описание специфических функций, характерных для каждой СУБД.

Справочник подготовлен профессиональными администраторами и опытными разработчиками, использующими различные диалекты SQL для поддержки сложных корпоративных приложений. Основная задача издания – служить кроссплатформенным руководством для тех, кто, не будучи экспертами, занимается переносом кода (включая пользовательские приложения) между различными СУБД. Независимо от того, является ли читатель новичком в SQL или имеет значительный опыт его использования, он найдет много полезных советов и приемов в этой лаконичной и удобной для работы книге.



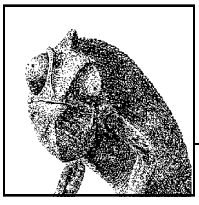
**ISBN 978-5-93286-165-3**
**ISBN 978-0-596-51884-4 (англ)**





Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 380-5007, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.
Подписано в печать 30.09.2009. Формат 70x100$^{1}/_{16}$. Печать офсетная.
Объем 41 печ. л. Тираж 1500 экз. Заказ N
Отпечатано с диапозитивов в ГУП «Типографии «Наука»
199034, Санкт-Петербург, 9 линия, 12.

# Оглавление



# Предисловие

С момента появления в 1970-х годах развитие языка SQL (structured query language, структурированный язык запросов) шло в параллели с информационным бумом, и в результате SQL стал самым широко используемым языком управления базами данных. Множество компаний и отдельных разработчиков, включая разработчиков ПО с открытым кодом (*http://www.opensource.org*), создали свои диалекты SQL в соответствии со своими потребностями. В то же время комитеты, занимающиеся стандартами, разрабатывали растущий список стандартных возможностей.

В этом справочнике описываются все операторы SQL согласно последнему стандарту ANSI SQL2003(SQL3), а также особенности реализации этих операторов в различных СУБД. В этой книге вы найдете описание реляционных моделей данных, объяснение основных концепций реляционных СУБД, полное описание синтаксиса SQL.

Но самое главное, особенно для программиста или разработчика (проектировщика ПО), это то, что эта книга является справочником по двум самым популярным коммерческим СУБД (Microsoft SQL Server и Oracle) и двум самым известным СУБД с открытым кодом (MySQL и PostgreSQL). Внимание, уделяемое в этой книге системам с открытым кодом, отражает возрастающую важность таких систем для компьютерного сообщества.

Синтаксис SQL, описанный в этой книге, включает следующие варианты реализации:

- В соответствии со стандартом ANSI SQL2003 (SQL3)
- MySQL версии 5.1
- Oracle Database 11g
- PostgreSQL версии 8.3
- Microsoft SQL Server 2008

## Зачем нужна эта книга?

Основным источником информации по реляционным СУБД традиционно является документация и файлы справки, поставляемые разработчиками. Хотя доку-

ментация и является необходимым ресурсом, к которому большинство разработчиков и администраторов обращается в первую очередь, такая документация имеет определенные ограничения:

- Реализация языка SQL описывается без указания того, насколько хорошо эта реализация согласуется с ANSI стандартом
- Описывается только один конкретный продукт, но не дается никакого описания возможных проблем при переводе, миграции или интеграции
- Описание приводится в множестве маленьких разрозненных файлов
- Описание отдельных команд зачастую настолько детально, что сбивает с толку, скрывая простые и очевидные способы применения этих команд в повседневной работе программиста или администратора.

Другими словами, поставляемая с СУБД документация дает исчерпывающее объяснение каждого аспекта конкретной версии продукта. И это нормально: в конце концов, документация и нужна для того, чтобы дать всю информацию о продукте. В документации приводится синтаксис команд (со всеми мыслимыми вариантами), и в общих чертах описывается использование. Тем не менее, если вы работаете с разными СУБД и хотите работать продуктивно, то вы вряд ли будете использовать эти специфические особенности; наоборот, вы будете использовать те возможности, которые необходимы в большинстве реальных ситуаций.

Эта книга начинается там, где заканчивается документация. В книге использован опыт профессиональных администраторов баз данных и разработчиков, использующих различные диалекты SQL в своей каждодневной работе по поддержке сложных корпоративных приложений. Вам предлагается воспользоваться их опытом, изложенным в компактном и легко используемом формате. Независимо от того, являетесь ли вы новичком в SQL или используете его на протяжении нескольких лет, вы найдете множество полезных советов и приемов. А при переходе от одной СУБД к другой важно быть в курсе основных возможных проблем, которые могут возникнуть, если вы не будете осведомлены и осторожны.

## Для кого предназначена эта книга?

Этот справочник будет полезен разным читателям. Это и программисты, которым нужен ясный и удобный справочник по SQL; и разработчики, занимающиеся миграцией с одного диалекта SQL на другой; и администраторы баз данных, которым нужно выполнять тысячи команд для поддержки корпоративных информационных систем в рабочем состоянии, а также управлять объектами баз данных, такими как таблицы, индексы, представления.

Эта книга является справочником, а не учебником. Мы, к примеру, не объясняем концепцию элементарного цикла; опытному программисту это известно, и ему «мясо» подавай. Поэтому мы объясняем, как должен работать курсор по стандарту ANSI, и как курсор работает на описываемых платформах, особенности курсоров на каждой платформе, а также основные потенциальные проблемы при использовании курсоров и методы решения этих проблем.

Хотя эта книга не учебник по SQL или проектированию баз данных, мы даем некоторую вводную информацию и надеемся, что она окажется вам полезной. Главы 1 и 2 дают краткое введение в SQL, описывая истоки возникновения этого

языка, основные структуры и базовые операции. Если вы новичок в SQL, то эти главы помогут вам начать работать.

## Как организована эта книга

Книга содержит четыре главы и приложение:

Глава 1. *История и реализации SQL*

Обсуждаются реляционная модель данных, описываются текущий и предыдущие стандарты SQL, дается введение в реализации SQL, описываемые в этой книге.

Глава 2. *Основные концепции*

Описываются концепции, необходимые для понимания реляционных баз данных и SQL команд.

Глава 3. *Справочник операторов SQL*

Приводится алфавитный справочник операторов SQL. В этой главе описывается последний стандарт ANSI (SQL3) и детали реализации каждой команды в MySQL, Oracle, PostgreSQL и SQL Server.

Глава 4. *Функции SQL*

Описываются все функции SQL по стандарту ANSI SQL3, а также особенности реализации этих функций производителями. Также приводятся все специфические функции, реализованные в каждой платформе.

Приложение. *Ключевые слова: общие и платформо-зависимые*

Приводится список ключевых слов, зарезервированных стандартом SQL3 и фирмами-производителями. Не используйте слова из этого списка для именования объектов и переменных.

## Как использовать эту книгу

Книга в первую очередь является справочником команд. Следовательно, вы будете использовать ее для получения информации по командам и функциям SQL. Однако при наличии описания по стандарту и по четырем платформам описание каждой команды может быть очень большим.

Для того чтобы избежать излишнего многословия при описании, мы сравниваем реализацию каждой платформы со стандартом SQL3. Если реализация согласуется со стандартом, мы не повторяем описание еще раз.

Обобщенные и переносимые между платформами примеры приводятся в описании каждой команды по стандарту SQL3. Так как стандарт развивается быстрее большинства реализаций, то для тех команд, которые не поддерживаются ни одной из рассматриваемых в книге платформ, примеры не приводятся. С другой стороны, приводятся примеры для пояснения многочисленных расширений и улучшений команд, реализованных на каждой конкретной платформе.

Мы понимаем, что при таком подходе может потребоваться неоднократно перепрыгивать от описания платформенной реализации команды обратно к описанию

команды в стандарте SQL3. Однако мы верим, что это все же лучше, чем забивать книгу сотнями страниц повторяющейся информации.

## Ресурсы

На следующих веб-сайтах можно найти дополнительную информацию о платформах, рассматриваемых в этой книге:

*MySQL*

Корпоративным ресурсом по MySQL является *http://www.mysql.com*, еще один хороший сайт *http://dev.mysql.com/doc/refman/5.1/en/*. Хорошим ресурсом для разработчиков с множеством полезных примеров является Devshed.com Ищите информацию по MySQL по ссылке *http://www.devshed.com/c/b/MySQL*.

*PostgreSQL*

Домашняя страница этой СУБД с открытым кодом расположена на *http://www.postgresql.org*. Помимо огромного количества информации для скачивания на сайте поддерживается рассылка для пользователей PostgreSQL. Другой достойный внимания сайт, предлагающий также поддержку коммерческим пользователям, – *http://www.pgsql.com*.

*Oracle*

Портал Oracle расположен на *http://www.oracle.com*. Огромное количество информации для постоянных пользователей Oracle находится по адресу *http://www.oracle.com/technology/*. Вся документация доступна по ссылке *http://www.oracle.com/technology/documentation/index.html*. Для полезной информации об Oracle от независимых источников посетите сайт Independent Oracle User Group *http://www.ioug.org*.

*SQL Server*

Официальный сайт Microsoft SQL Server *http://www.microsoft.com/sql/*. Другой хороший ресурс расположен на сайте Профессиональной ассоциации пользователей SQL Server (PASS) *http://www.sqlpass.org*.

## Изменения в третьем издании

Основной причиной для выпуска нового издания книги о технологии обычно является прогресс этой технологии. С момента выпуска второго издания этой книги был создан новый стандарт ANSI, а большинство производителей СУБД выпустили как минимум по одному новому релизу. В результате наши читатели получат свежую информацию о последних версиях языка SQL, представленных на рынке.

Вот основные изменения в третьем издании:

*Уменьшенный охват*

Читателям второго издания пришлось по душе представленное в нем описание всех основных СУБД. Но поддержка такого большого количества информации в актуальном состоянии оказалась сложным делом. Поэтому по результатам большого опроса читателей мы решили исключить из этого изда-

ния обзор двух наименее популярных платформ: Sybase Adaptive Server и IBM DB2 UDB.

*Улучшенная организация*

При подготовке второго издания была проделана громадная работа по описанию всего, что читатель может захотеть узнать о командах и функциях SQL в основных СУБД. Но необходимую информацию не всегда было легко найти. Мы улучшили указатели, содержание и колонтитулы страниц, чтобы поиск был быстрее и эффективнее.

*Больше примеров*

Примеров слишком много не бывает. Мы пополнили и так немалое число базовых примеров, добавив примеры кода, поясняющие уникальные и мощные возможности стандартного SQL и расширений, предлагаемых платформами по отдельности.

## Обозначения, принятые в данной книге

В этой книге мы придерживаемся следующих обозначений:

`Моноширинный шрифт`

Применяется для описания синтаксиса, фрагментов кода и примеров.

*`Моноширинный шрифт с курсивом`*

Используется для указания переменных в коде, вместо имен которых пользователь должен ввести собственные значения.

**`Жирный моноширинный шрифт`**

Используется для выделения отдельных фрагментов кода.

*Курсив*

Используется при введении новых терминов, для привлечения внимания, для указания команд или вводимых пользователем имен файлов и папок, а также для указания переменных в тексте.

**Жирный шрифт**

Используется для имен объектов базы данных, таких как таблицы, столбцы или хранимые процедуры.

*КУРСИВ В ВЕРХНЕМ РЕГИСТРЕ*

Используется для ключевых слов SQL.

Так оформляется дополнительная информация (например, ссылка на материалы по теме или более подробное объяснение).

Такой текст содержит предупреждение или предостережение.

## Использование примеров кода

Эта книга должна служить подспорьем в вашей работе. В целом вы можете свободно использовать приведенный на ее страницах код в своих приложениях или документации. Если объем заимствованного вами кода невелик, нет необходимости обращаться к нам за разрешением. К примеру, вставка в вашу программу нескольких строчек кода из данной книги разрешения не требует. В то же время продажа или иное распространение CD-дисков с записью примеров, взятых из книг издательства O'Reilly, требует разрешения. Для цитирования материалов этой книги при ответе на вопрос специального разрешения не нужно, но при включении в составляемую вами документацию значительного объема кода из данной книги получение нашего разрешения потребуется.

При использовании наших материалов мы не требуем обязательной ссылки на источник, однако будем вам за нее благодарны. Ссылка на источник, как правило, включает в себя название, имя автора, издательство и ISBN книги, например: «*SQL in a Nutshell*, Third Edition, by Kevin E. Kline with Daniel Kline and Brand Hunt. Copyright 2009 O'Reilly Media, Inc., 978-0-596-51884-4».

Если вы чувствуете, что при использовании кода из книги вы вышли за рамки свободного распространения, оговоренные выше, пожалуйста, свяжитесь с нами по адресу *permissions@oreilly.com*.

## Нам интересно ваше мнение

Мы, конечно, тестировали и выверяли информацию, приведенную в данной книге, максимально внимательно, но, возможно, вы обнаружите, что в какие-то особенности были внесены изменения (или мы что-то пропустили!). Мы будем вам признательны за подобные сведения, особенно, если это поможет сделать книгу лучше. Пожалуйста, присылайте свои вопросы и комментарии по поводу этой книги издателю по следующему адресу:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (для звонков из США или Канады)
(707) 829-0515 (для международных или местных звонков)
(707) 829-0104 (факс)

На сайте издательства есть посвященная этой книге страница, где приводятся примеры, список опечаток и другие дополнительные сведения. Адрес страницы:

*www.oreilly.com/catalog/9780596518844/*

Пожалуйста, укажите нам на наши опечатки и синтаксические ошибки. (Вы можете представить, насколько трудно откорректировать книгу, описывающую стандарт ANSI и четыре разные реализации.) Адрес электронной почты для комментариев и вопросов технического характера, связанных с этой книгой:

*booksquestions@oreilly.com*

Чтобы получить информацию о наших книгах, конференциях, ресурсных центрах и сети O'Reilly Network, посетите наш веб-сайт, расположенный по адресу:

*www.oreilly.com*

## Safari® Books Online


Если на обложке технической книги есть пиктограмма «Safari® Books Online», это означает, что книга доступна в Сети через O'Reilly Network Safari Bookshelf.

Safari предлагает намного лучшее решение, чем электронные книги. Это виртуальная библиотека, позволяющая без труда находить тысячи лучших технических книг, вырезать и вставлять примеры кода, загружать главы и находить быстрые ответы, когда требуется наиболее верная и свежая информация. Она свободно доступна по адресу *http://safari.oreilly.com*.

## Благодарности

Мы хотели бы поблагодарить нескольких человек из издательства O'Reilly Media. В первую очередь, мы испытываем огромную благодарность к Джулии Стил (Julie Steel), редактору третьего издания. Джулия не давала нам расслабиться, и благодаря ей мы закончили то, что начали. Благодаря ее мягкому и предупредительному стилю с Джулией было очень приятно работать. Спасибо тебе за все, что ты для нас сделала!

Также мы благодарим наших замечательных технических рецензентов: Питера Гулутзана (Peter Gulutzan), который отвечал за стандарт SQL и работал с нами еще со второго издания; Томаса Локхарта (Thomas Lockhart), специалиста по PostgreSQL; Рональда Брэдфорда (Ronald Bradford) – Oracle/MySQL и Ричарда Соннена (Richard Sonnen) – Oracle. Сердечное спасибо! Вы внесли существенный вклад в точность, удобочитаемость и ценность этой книги. Без вас наши разделы о расширениях языка были бы неполными. В дополнение снимаем шляпу перед Питером Гулутзаном и Труди Пельтцер (Trudy Pelzer) за их книгу «SQL-99 Complete, Really!», которая помогла нам разобраться в стандарте SQL3.

### Благодарности Кевина Кляйна

В создании толстой книги, которую вы держите в руках, нам помогали многие люди. Мы выражаем признательность тем, кто помог этой книге стать реальностью.

Во-первых, большое спасибо двум моим фантастическим соавторам – Дэну и Брэнду. Вы удивительные ребята, и с вами было приятно работать. Крепко обнимаю Джулию Стил, нашего редактора из O'Reilly Media.

Большое спасибо моим коллегам из Quest Software за поддержку и ободрение. Кристиан Хаскер (Christian Hasker), Энди Грант (Andy Grant), Хизер Эйкман (Heather Eichman), Дэвид Гугик (David Gugick), Билли Босворс (Billy Bosworth), Дуглас Кристалл (Douglas Chrystall), Дэвид Свонсон (David Swanson), Джейсон

Холл (Jason Hall), Эриэл Вэйл (Ariel Weil) и многие другие мои друзья из Quest Software, – спасибо, что сделали эти последние шесть лет в Quest Software такими яркими!

Посвящаю эту книгу своим любимым Дилану, Эмили, Анне и Кэти. Вы были моей надеждой, дыханием и светом в моменты, когда казалось, что в мире уже не оставалось ни надежды, ни дыхания, ни света. Спасибо, что любите меня так сильно и бескорыстно. И наконец, спасибо моей драгоценнейшей Рэйчел – твоя любовь сохранила мое сердце и веру.

## Благодарности Дэниэла Кляйна

Я хотел бы поблагодарить моего брата Кевина за его постоянную готовность работать вместе со мной; моих коллег из Университета Анкоридж штата Аляска за их пожелания; и читателей первых двух изданий за их честные отзывы и здравую критику. Мы также получили огромное количество отзывов от переводчиков первых двух изданий, и за это им также большое спасибо.

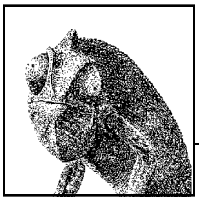
## Благодарности Брэнда Ханта

Спасибо моей жене Мишель: без твоей поддержки и любви я бы не был частью этого проекта. Я ценю каждый момент нашей совместной жизни, и особенно ценю, что ты прощала мне постоянное ночное клацанье по клавиатуре, мешавшее тебе спать.

Спасибо моим родителям Рексу и Джэки, тем людям, в наибольшей степени благодаря которым мне удается доводить до конца свои начинания, особенно те, которые требуют многократных усилий (например, написание книги!).

Спасибо членам нашей команды – Кевину, Дэниэлу и Джонатану – за то, что дали мне участвовать в этом проекте и потратили силы на обучение человека, которому впервые довелось писать книгу для O'Reilly. Ваш профессионализм, этика рабочих отношений и умение превратить самую нудную задачу в удовольствие так восхищают, что я намерен украсть эти умения и владеть ими в дальнейшем!

Спасибо моим друзьям и коллегам из Rogue Wave Software, ProWorks, NewCode Technology и Systems Research and Development за предоставление мне песочницы, в которой я мог оттачивать свои навыки SQL, баз данных, бизнеса, разработки программного обеспечения, письма и дружбы. Это Гус Уотрес (Gus Waters), Грэг Коупер (Greg Koerper), Марк Мэнли (Marc Manley), Вэнди Минн (Wendi Minne), Эрин Фоли (Erin Foley), Элэйн Кулл (Elaine Cull), Рэндал Робинсон (Randall Robinson), Дэйв Риттер (Dave Ritter), Эдин Дзулиц (Edin Zulic), Дэвид Нур (David Noor), Джим Шур (Jim Shur), Крис Мосбрукер (Chris Mosbrucker), Дэн Робин (Dan Robin), Майк Фокс (Mike Faux), Джейсон Протеро (Jason Prothero), Тим Романовски (Tim Romanowski), Энди Мосбрукер (Andy Mosbrucker), Джефф Джонас (Jeff Jonas), Джефф Бутчер (Jeff Butcher), Чарли Барбур (Charlie Barbour), Стив Данхэм (Steve Dunham), Брайэн Мэйси (Brian Macy) и Зив Мейлер (Ze'ev Mehler).



# 1

# История и реализации SQL

В начале 1970-х работы сотрудника IBM профессора Э. Ф. Кодда послужили основой при разработке продукта для работы с реляционными данными, получившего название SEQUEL (Structured English Query Language). SEQUEL позднее превратился в SQL (Structured Query Language).

IBM, равно как и другие производители СУБД, был очень заинтересован в стандартизации методов доступа и манипуляции данными в реляционных базах данных. Хотя в IBM была разработана теория реляционных баз данных, компания Oracle первой вывела технологию на рынок. Со временем SQL стал достаточно популярным, чтобы привлечь к себе внимание Американского национального института по стандартам (ANSI), выпустившего стандарты SQL в 1986, 1989, 1992, 1999, 2003 и 2006 годах. В этом издании используется стандарт 2003 года, так как стандарт SQL2006 описывает элементы SQL, не входящие в поле рассмотрения данной книги. (По существу, SQL2006 описывает использование XML в SQL.)

С 1986 года было создано много различных языков, позволявших программистам и разработчикам работать с реляционными данными. Но лишь немногие из них были столь же просты для изучения и стали настолько общеприняты, как SQL. Программисты и администраторы получили возможность изучения одного языка, который с небольшими поправками применим к широкому спектру платформ СУБД, приложений и продуктов.

В этом справочнике приводятся описания синтаксиса пяти реализаций SQL2003:

- Синтаксис по стандарту ANSI SQL
- MySQL версии 5.1
- Oracle Database 11g
- PosgreSQL версии 8.3
- Microsoft SQL Server 2008

## Реляционная модель и ANSI SQL

*Реляционные системы управления базами данных*, такие как описываемые в этой книге, являются двигателем множества информационных систем в мире, в особенности веб-приложений и распределенных клиент-серверных вычислительных

систем. Реляционные СУБД позволяют множеству пользователей быстро и одновременно получать доступ, создавать, редактировать данные, не влияя на работу других пользователей. Реляционные СУБД позволяют разработчикам создавать полезные приложения для доступа к данным, а администраторам дают возможность развивать, защищать, оптимизировать информационные ресурсы.

Реляционная СУБД – это система, пользователи которой получают доступ к данным, представленным в виде набора связанных таблиц. Данные хранятся в *таблицах*, состоящих из *строк* и *столбцов*. Таблицы, хранящие независимые данные, могут быть *связаны* между собой, если в них есть столбцы, значения в которых уникально идентифицируют строки. Наборы таких столбцов называют ключами. Кодд впервые описал реляционную теорию в своей исторической работе «Реляционная модель данных для больших разделяемых банков данных», опубликованной в журнале Communications of the ACM (Association for Computing Machinery) в 1970 году. Согласно новой реляционной модели данных Кодда данные должны быть *структурированными* (в таблицы из строк и столбцов), *управляемыми* с помощью операторов, таких как выборка, проекция, объединение, и *целостными* в результате применения правил контроля целостности, таких как первичные и внешние ключи. Кодд также изложил правила, которые должны применяться при проектировании реляционных баз данных. Процедура применения этих правил получила название *нормализации*.

## Правила Кодда для реляционных баз данных

Кодд применил строгие математические теории (в основном, теорию множеств) к управлению данными и составил список критериев, которому должна удовлетворять база данных, чтобы считаться реляционной. Основной концепцией реляционной теории является хранение данных в таблицах. Эта концепция сейчас настолько привычна, что кажется тривиальной; однако еще не так давно создание системы, поддерживающей реляционную модель, считалось сложной задачей с малополезным результатом.

Рассмотрим *двенадцать принципов реляционных баз данных*:

1. Логически информация должна быть представлена в виде таблиц.
2. Каждый элемент данных должен быть доступен по комбинации имени таблицы, значения первичного ключа и названия столбца.
3. Значения NULL должны однозначно трактоваться как «отсутствующая информация», а не как пустые строки, пробелы или нули.
4. Метаданные (данные, описывающие базу данных) должны храниться в базе данных – так же, как и обычные данные.
5. Один и тот же язык должен использоваться для создания данных, представлений, ограничений целостности, авторизации, транзакций и работы с данными.
6. Представления должны отражать изменения данных в таблицах, на основе которых они созданы (и наоборот).
7. Должно быть достаточно одной команды для выполнения любой из следующих операций: извлечение данных, вставка данных, изменения данных и удаление данных.

8. Пакетные и пользовательские команды должны быть логически отделены от структур физического хранения и методов доступа.
9. Пакетные и пользовательские команды должны иметь возможность изменения модели данных без пересоздания этих данных или приложений, использующих базу данных.
10. Ограничения целостности должны храниться в метаданных, а не в приложениях.
11. Язык манипуляции данными не должен зависеть от того, являются ли данные централизованными или распределенными.
12. И пакетная, и построчная обработка должны подчиняться одним и тем же ограничениям и правилам целостности.

Эти принципы продолжают оставаться «лакмусовой бумажкой» для определения реляционной СУБД: платформа, не удовлетворяющая всем критериям, не может считаться реляционной. Эти правила относятся не к процессу разработки приложений, а относятся только к самому «движку», к СУБД. На сегодняшний момент большинство коммерческих СУБД проходят тест Кодда. Среди СУБД, рассматриваемых в третьем издании этого справочника, только MySQL не удовлетворял всем принципам, и то только в релизе, который предшествовал рассматриваемому здесь.

Понимание принципов Кодда помогает программистам и разработчикам в правильном проектировании и создании реляционных баз данных. В следующих разделах мы покажем, как SQL удовлетворяет этим принципам.

## Структуры данных (правила 1, 2 и 8)

Правила Кодда 1 и 2 формулируются следующим образом: «логически информация должна быть представлена в виде таблиц» и «каждый элемент данных должен быть доступен по комбинации имени таблицы, значения первичного ключа и названия столбца». Таким образом, при создании таблицы не требуется указание того, как базе данных взаимодействовать с лежащими в основе физическими структурами данных. Более того, SQL логически изолирует процесс доступа к данным и физическое хранение данных, как того требует правило 8: «пакетные и пользовательские команды должны быть логически отделены от структур физического хранения и методов доступа».

В реляционной модели данные представляются в виде двухмерной *таблицы*, описывающей отдельную сущность (например, расходы компании). Теоретики называют таблицы *сущностями*. Таблицы состоят из *строк* или *записей* (ученые называют их *кортежами*) и *столбцов* (теоретики называют их *атрибутами*, так как столбец описывает какой-либо атрибут сущности). Пересечение записи и столбца образует *значение*. Один или несколько столбцов, чьи значения позволяют идентифицировать запись, могут выступать в качестве *первичного ключа*. В наши дни это кажется элементарным, но когда эти концепции были предложены, они явились действительно инновационными.

SQL3 определяет сложную иерархическую структуру объектов помимо простых таблиц, являющихся базовыми структурами данных. Реляционный подход состоит в работе с данными на уровне таблиц, а не на уровне отдельных строк. За ориентацией на обработку таблиц стоит работа с множествами. Как следствие,

почти все SQL-команды эффективнее работают с множествами строк, а не с отдельными строками. Другими словами, для эффективного использования SQL необходимо мыслить в терминах наборов данных, а не в терминах отдельных строк.

На рис. 1.1 представлена терминология SQL3, используемая для описания иерархической структуры реляционной базы данных: *кластер* содержит множество каталогов; *каталог* содержит множество схем; *схема* содержит множество объектов, таких как таблицы или представления; *таблицы* состоят из строк и столбцов.

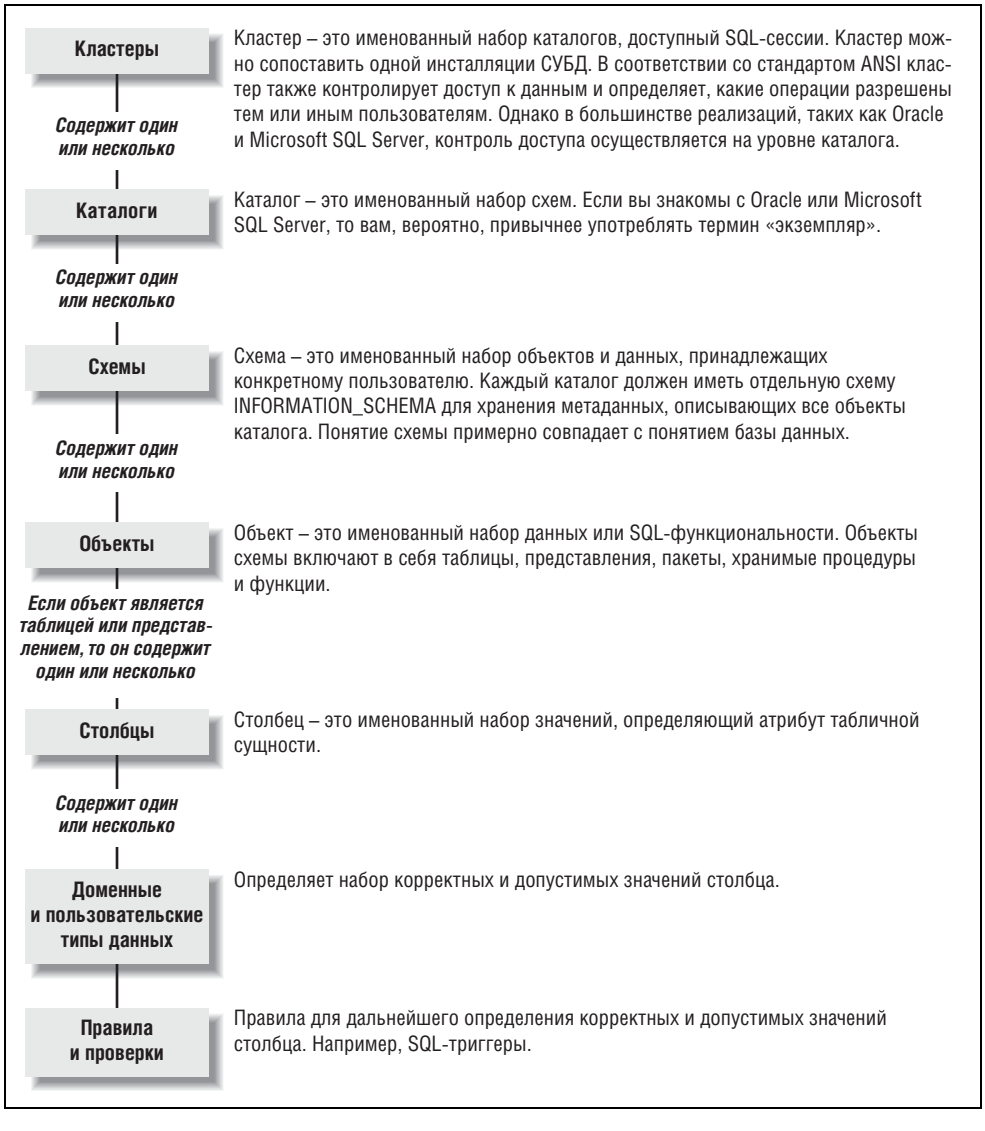



***Рис. 1.1.** Иерархия данных в SQL3*

Например, в таблице **Business_Expense** столбец с названием **Expense_Date** мог бы означать дату, в которую были понесены расходы. Каждая запись в таблице описывает отдельную сущность, в нашем случае – все, что описывает расходы (когда произведен, на какую сумму, кем, для чего и т. д.) Каждый атрибут затрат – другими словами, каждый столбец – должен быть *атомарным*, что означает, что столбец записи должен содержать одно и только одно значение. Если таблица создана так, что на пересечении строки и столбца может содержаться несколько значений, то это нарушает одно из базовых положений SQL-проектирования. (Некоторые СУБД, рассматриваемые в этой книге, позволяют хранить в столбце больше одного значения с помощью типов *VARRAY* и *TABLE*.)

Для значений столбцов определяются правила поведения. В первую очередь, все значения столбца должны относиться к одному и тому же *домену* или, по-другому, к одному *типу данных*. К примеру, если столбец **Expense_Date** имеет тип *DATE*, то значение *ELMER* в этот столбец сохранить нельзя, так как это значение является не датой, а строкой, а в столбце **Expense_Date** допустимы только даты. Кроме того, SQL3 позволяет контролировать значения столбцов с помощью *ограничений* (обсуждаются детально в главе 2) и *проверок* (assertion). SQL-ограничение может, к примеру, проверять, что в столбце **Expense_Date** содержатся даты не более чем годичной давности. Доступ к данным всех пользователей и процессов контролируется с помощью *AuthorizationID* или *user*. Доступ или модификация определенных наборов данных может разрешаться или запрещаться на уровне отдельных пользователей.

В реляционных базах также используются понятия *кодировки* и *схем упорядочивания*. Кодировка – это «символы» или «алфавит», используемые «языком» данных. Например, кодировка для американского английского не содержит буквы *ñ* из испанской кодировки. Схема упорядочивания – это правила сортировки для кодировки. Схема упорядочивания определяет, как при манипуляции будут отсортированы данные. Например, для символов американского английского могут быть применены сортировки *алфавитная*, *регистронезависимая* или *алфавитная*, *регистрозависимая*.

Стандарт ANSI не описывает, как сортировки должны выполняться, но говорит о необходимости поддержки основных схем упорядочивания отдельного языка.

При написании SQL-кода для конкретной платформы СУБД вы должны понимать, какая используется схема упорядочивания, потому что это оказывает большое влияние на поведение запросов, в особенности, на фразы *WHERE* и *ORDER BY*. Например, запрос, использующий бинарную схему упорядочивания, вернет записи в другом порядке, нежели запрос, сортирующий данные по алфавиту.

## Неопределенные значения (правило 3)

В большинстве СУБД любой тип данных может иметь неопределенное значение (NULL). Неопытные SQL-программисты и разработчики часто принимают NULL за ноль или пробел. Но NULL – это ни то ни другое. В SQL3 NULL буквально означает, что значение неизвестно или не определено. (Вопрос о том, считать ли NULL все-таки неопределенным или же неизвестным значением, является пред-

метом постоянных научных дебатов). Такое разделение позволяет проектировщику базы данных отличать данные, сознательно введенные как ноль, и данные, значение которых не зафиксировано в системе или намеренно проставлено как неопределенное. Как иллюстрацию семантического различия, представим систему работы с платежами. Если в качестве суммы платежа стоит NULL, то это не значит, что товар был куплен бесплатно. Это означает, что количество товара неизвестно или еще не определено.

Существует много отличий в том, как каждая СУБД работает с неопределенными значениями. И это является одним из главных источников проблем при переносе кода между платформами. Например, пустая строка в Oracle представляется значением NULL. Все остальные СУБД, рассматриваемые в этой книге, позволяют явно сохранить пустую строку в столбце с типом *CHAR* или *VARCHAR*.

Побочным эффектом такой неопределенной природы значения NULL является то, что NULL не используется в вычислениях и сравнениях. Вот несколько коротких, но очень важных правил из стандарта ANSI, которые необходимо помнить при работе с неопределенными значениями в выражениях SQL:

- Значение NULL нельзя вставить в столбец, определенный как NOT NULL.
- Значения NULL не равны между собой. Ожидать, что два значения NULL окажутся равны при сравнении двух столбцов – это очень частая ошибка. (Правильным способом определения NULL в логическом выражении или во фразе *WHERE* является использование специальных фраз «IS NULL» и «NOT IS NULL»).
- Значения NULL игнорируются при вычислении по столбцу агрегированных значений, таких как *AVG*, *SUM* или *MAX*.
- Если столбец, содержащий NULL, используется в запросе во фразе *GROUP BY*, то результат запроса содержит одну строку для всех значений NULL. Стандарт ANSI предписывает, что все найденные неопределенные значения должны попасть в одну группу.
- Фразы *DISTINCT* и *ORDER BY*, как и *GROUP BY*, также не различают значения NULL друг от друга. При сортировке с помощью фразу *ORDER BY* производители СУБД могут устанавливать произвольным образом порядок сортировки по умолчанию для NULL.

## Метаданные (правила 4 и 10)

Четвертое из правил Кодда для реляционных баз данных требует, чтобы описание базы данных хранилось в таких же таблицах, что и обычные данные. Данные, описывающие саму базу данных, называются *метаданными*. Например, каждый раз, когда вы создаете в базе данных таблицу или представление, создаются и сохраняются записи, описывающие эти новые таблицы. Дополнительные записи требуются для описания столбцов, ключей или ограничений таблицы. Эта технология реализована в большинстве коммерческих и открытых реляционных СУБД. Например, в SQL Server есть так называемые системные таблицы, в которых отслеживается информация о базах данных и всех их объектах. Кроме того, SQL Server использует «системные базы данных» для хранения инфор-

мации о сервере, на котором установлено и сконфигурировано программное обеспечение СУБД.

## Язык (правила 5 и 11)

Правила Кодда *не требуют*, чтобы при работе с базами данных использовался именно язык SQL. Эти правила, в частности 5 и 11, только определяют, как должен вести себя язык при работе с реляционными базами данных. Одно время SQL состязался с другими языками (такими как RDO и Fox/PRO), которые также удовлетворяли критериям реляционности, и SQL победил, по трем причинам. Во-первых, SQL относительно простой, интуитивно понятный и похожий на естественный английский, язык, но при этом охватывает большинство аспектов манипуляции данными. Во-вторых, SQL достаточно высокоуровневый язык. Программист или администратор баз данных не должен тратить время и силы на сохранение данных в нужные регистры оперативной памяти или на кэширование данных на диск; все эти задачи СУБД решает автоматически. Наконец, SQL не принадлежит какому-то одному производителю и поэтому успешно был адаптирован к множеству различных платформ.

## Представления (правило 6)

*Представление* – это виртуальная таблица, не хранящая данные физически, а создаваемая на лету из SQL-запроса каждый раз, когда происходит обращение к представлению. Представления позволяют создавать несколько разных образов одних и тех же исходных данных для разных пользователей без необходимости модифицировать способ хранения данных.

Некоторые производители СУБД поддерживают объекты, называемые материализованными представлениями. Но не попадитесь на схожесть названий: материализованные представления не подчиняются тем же правилам, что и стандартные ANSI-представления.

## Операции над множествами (правила 7 и 12)

Многие языки для манипуляции данными, такие как многоуважаемый Xbase, выполняют операции совсем не так, как SQL. В этих языках необходимо явно указывать, что делать с данными, – с каждой строкой. Программа в цикле перебирает все строки, применяя логику обработки к каждой строке, и поэтому такой стиль программирования часто называется *построчной обработкой*, или *процедурным программированием*.

Программы на SQL логически оперируют *множествами* данных. Теория множеств применяется почти во всех операторах SQL, включая операторы *SELECT*, *INSERT*, *UPDATE* и *DELETE*. Данные выбираются из множества, называемого таблицей. В отличие от построчной обработки, при работе с множествами программист говорит, что ему требуется получить, а не как обрабатывать каждый кусочек данных. Иногда работу с множествами называют *декларативной обработкой*, потому что программист декларирует желаемый результат (например, «Мне нужны все сотрудники из южного региона с зарплатой более $70 000 в год»), а не описывает всю процедуру извлечения данных.

Теория множеств была разработана математиком Георгом Кантором в конце XIX века. В то время она была воспринята достаточно противоречиво. Сегодня теория множеств настолько плотно вошла в нашу жизнь, что преподается в школе. Простыми и общеизвестными применениями теории множеств являются карточные каталоги, десятичная система классификации Дьюи, алфавитные телефонные справочники.

Связь теории множеств и реляционных баз данных детально описывается в следующих разделах.

## Правила Кодда в действии: простой пример SELECT

До сих пор в этой главе мы рассматривали отдельные аспекты теории реляционных баз данных, определенных Коддом и реализованных в стандарте ANSI SQL. В этом разделе приводится общий обзор наиболее важного SQL-оператора, *SELECT*, и основных его применений – а именно, выполнение реляционных операций проекции, выборки и соединения.

*Проекция*

Выбираются конкретные столбцы данных

*Выборка*

Выбираются конкретные строки

*Соединение*

Одним результирующим множеством возвращаются строки столбцы из нескольких таблиц

Хотя на первый взгляд может показаться, что оператор *SELECT* выполняет только реляционную операцию выборки, но на самом деле он выполняет все три операции.

Следующий оператор осуществляет проекцию, выбирая имя и фамилию автора из таблицы **authors**:

```
SELECT au_fname, au_lname, state
FROM   authors
```

Результат этого оператора *SELECT* представляется в виде другой таблицы данных:

```
au_fname          au_lname            state
----------------- ------------------- -----
Johnson           White               CA
Marjorie          Green               CA
Cheryl            Carson              CA
Michael           O'Leary             CA
Meander           Smith               KS
Morningstar       Greene              TN
Reginald          Blotchet-Halls      OR
Innes             del Castillo        MI
```

Результат выполнения запроса иногда называют результирующим множеством (result set), рабочей таблицей (work table) или производной таблицей (derived table) в отличие от базовой таблицы, по отношению к которой выполняется оператор *SELECT*.

Важно заметить, что в операторе *SELECT* фраза *SELECT* (т. е. ключевое слово *SELECT* со списком извлекаемых столбцов и выражений) соответствует реляционной операции проекции. Выборка – операция извлечения конкретных строк – реализуется фразой *WHERE* оператора *SELECT*. *WHERE* отфильтровывает ненужные строки результата, оставляя только необходимые. Дополним предыдущий пример условием отбора только авторов не из Калифорнии:

```
SELECT au_fname, au_lname, state
FROM   authors
WHERE  state <> 'CA'
```

В то время как предыдущий запрос возвращал всех авторов, этот запрос возвращает намного меньшее подмножество строк:

```
au_fname        au_lname            state
--------------- ------------------- -----
Meander         Smith               KS
Morningstar     Greene              TN
Reginald        Blotchet-Halls      OR
Innes           del Castillo        MI
```

Комбинируя операции проекции и выборки в одном запросе, вы можете с помощью SQL выбрать только те строки и столбцы, которые вам нужны.

Последняя реляционная операция, которую мы обсудим в этом разделе, – это соединение. Соединение связывает две таблицы и возвращает результирующее множество, состоящее из данных из обеих таблиц.

Разные разработчики СУБД позволяют соединить в одном запросе разное число таблиц. Oracle не накладывает ограничений на число соединяемых таблиц, SQL Server позволяет соединять не больше 256 таблиц.

Описанный в стандарте ANSI способ соединения таблиц заключается в использовании фразы *JOIN* оператора *SELECT*. Более старый способ, известный как тэта-соединения, для задания соединения использует фразу *WHERE*. Следующий пример демонстрирует оба подхода. Каждый оператор выбирает информацию о сотрудниках из таблицы **employee** и название должности из таблицы **jobs**. Первый оператор *SELECT* использует новый синтаксис соединения при помощи *JOIN*, а второй использует тэта-соединение:

```
-- ANSI style
SELECT a.au_fname, a.au_lname, t.title_id
FROM   authors AS a
JOIN   titleauthor AS t ON a.au_id = t.au_id
WHERE  a.state <> 'CA'

-- Theta style
SELECT a.au_fname, a.au_lname, t.title_id
FROM   authors AS a,
```

```
        titleauthor AS t
WHERE   a.au_id = t.au_id
  AND   a.state <> 'CA'
```

Для дополнительной информации о соединениях читайте раздел «Фраза JOIN» в главе 3.

# История стандарта SQL

В ответ на быстрый рост числа различных SQL-диалектов ANSI в 1986 опубликовал первый стандарт SQL для внесения большей согласованности в действия разработчиков СУБД. Затем, в 1989 году, последовал второй, широко принятый стандарт. Стандарт SQL был также одобрен Международной организацией по стандартам (ISO). ANSI в 1992 году выпустил следующую версию, известную как SQL92 или SQL2, а затем еще одну в 1999, называемую SQL99 или SQL3. Версия, выпущенная в 2003 году, также называется SQL3 (или SQL2003). В этой книге под термином SQL3 мы подразумеваем именно стандарт 2003 года.

В каждой новой версии стандарта ANSI добавляет в язык новые команды и возможности. К примеру, в SQL99 были добавлены возможности работы с объектными типами данных.

## Новое в SQL2006

Комитет ANSI, отвечающий за SQL, выпустил новую версию стандарта в 2006 году, при этом в нем сохранились и были расширены все основные усовершенствования стандарта SQL3. Стандарт SQL2006 можно назвать эволюционным по отношению к стандарту SQL3, он не внес существенных изменений в команды и функции языка. Вместо этого стандарт SQL2006 описал новую функциональную область применения стандарта SQL. Если говорить коротко, то SQL2006 описывает взаимодействие SQL и XML. Например, стандарт SQL2006 описывает, как в реляционную базу импортировать и хранить XML данные, как манипулировать этими данными, и как их представлять в виде XML или обычных SQL данных в XML обертке. Стандарт SQL2006 предоставляет возможности интеграции SQL кода с кодом XQuery, языком запросов к XML, стандартизируемым консорциумом W3C. Так как XML и XQuery являются вполне самостоятельными дисциплинами, они не входят в рамки этой книги и не рассматриваются в ней.

## Новое в SQL2003 (SQL3)

SQL99 состоял из двух частей – *Foundation:1999* и *Bindings:1999*. Секция Foundation в SQL3 содержит обе части из SQL99, а также новую часть под названием *Schemata*.

Базовые требования в SQL3 не изменились по сравнению с базовыми требованиями SQL99, поэтому СУБД, удовлетворяющие SQL99, автоматически удовлетворяют SQL3. Хотя базовая часть SQL3 не содержит дополнений (за исключением нескольких новых зарезервированных слов), некоторые отдельные операторы были изменены или дополнены. Так как эти изменения описываются в синтаксисе конкретных операторов в главе 3, то здесь мы говорить о них не будем.

Некоторые элементы базовой части SQL99 в SQL3 были удалены:

- Типы данных *BIT* и *BIT VARYING*
- Фраза *UNION JOIN*
- Оператор *UPDATE ... SET ROW*

Некоторое число других, достаточно экзотических, возможностей было добавлено, удалено или переименовано. Многие новые возможности SQL3 интересны пока только с академической точки зрения, так как пока они не поддерживаются ни одной платформой. Тем не менее, мы не можем пройти мимо следующих возможностей:

*Элементарные OLAP-функции*

SQL3 вводит дополнительную поддержку OLAP (Online Analytical Processing, аналитическая обработка в реальном времени), включая оконные функции для широко используемых вычислений, таких как скользящее среднее или нарастающий итог. Оконные функции вычисляют агрегированные значения по окнам данных: *ROW_NUMBER*, *RANK*, *DENSE_RANK*, *PERCENT_RANK* и *CUME_DIST*. OLAP-функции подробно описаны в части T611 стандарта. Некоторые платформы СУБД уже имеют поддержку OLAP функций. За подробностями обращайтесь к главе 4.

*Выборки (sampling)*

SQL3 добавляет во фразу *FROM* фразу *TABLESAMPLE*. Эта возможность будет полезной для статистических запросов к большим базам данных, таким как хранилища данных.

*Улучшенные числовые функции*

SQL3 вводит большое количество новых числовых функций. В этом случае стандарт пытался успеть за развитием рынка, так как некоторые платформы СУБД уже поддерживают эти функции. Детали в главе 4.

## Уровни соответствия стандарту

В стандарте SQL92 впервые было введено понятие уровня соответствия стандарту. Было введено три уровня: *начальный*, *средний* и *полный*. Разработчикам СУБД требовалось достичь как минимум начального уровня соответствия для того, чтобы говорить о совместимости своей СУБД с ANSI SQL. Американский национальный институт по стандартам и технологиям (NIST) добавил *переходный* уровень между начальным и средним, таким образом, уровни соответствия по NIST были: начальный, переходный, средний и полный. Стандарт ANSI имел только три уровня: начальный, средний и полный. Каждый уровень соответствия является надмножеством нижестоящего уровня, т. е. каждый уровень содержит все требования нижестоящего уровня.

Позже стандарт SQL99 изменил уровни соответствия, убрав понятия начального, среднего и полного уровня. В SQL99 говорится о необходимости выполнения разработчиками СУБД всех базовых требований стандарта (Core SQL99) для заявления, что продукт разработчика готов к использованию с SQL99. Core SQL99 включает в себя все требования начального уровня соответствия из SQL92, требования из других уровней соответствия SQL92 и некоторые совершенно новые

требования. При желании разработчик может реализовать дополнительные пакеты возможностей, описанные в SQL99.

## Дополнительные пакеты возможностей в стандарте SQL3

Мало кто из разработчиков на текущий момент полностью удовлетворяет или даже превосходит требования стандарта. Базовые требования стандарта это как ограничение скорости на шоссе: кто-то едет чуть быстрее, кто-то чуть медленнее и почти никто не едет точно с допустимой скоростью. Так и реализация стандарта разными разработчиками может отличаться.

Два комитета – один в ANSI, другой в ISO, включающие представителей практически всех разработчиков, – составили описание дополнительных возможностей, о котором рассказывается в этом разделе. В такой объединенной и отчасти политизированной обстановке разработчики пришли к соглашению относительно того, какие возможности и особенности реализации должны войти в стандарт.

Нововведения в стандарте ANSI часто появляются из уже существующих продуктов или являются результатом новых исследований и разработок в научном сообществе. Поэтому поддержка функциональности разработчиками может быть неоднородной. Относительно новым добавлением в стандарт SQL3 является SQL/XML (значительно расширенным в SQL2006). Части стандарта SQL99 остались в SQL3 практически без изменений, хотя иногда немного менялись названия или порядок следования.

Девять дополнительных пакетов возможностей, представляющих различные наборы команд, являются опциональными для реализации. Какие-то элементы стандарта могут поддерживаться различными пакетами, в то же время есть возможности, не поддерживаемые ни одним пакетом. Вот список этих пакетов и их функциональностей:

*Часть 1. SQL/Framework*

Содержит общие определения и концепции, используемые в стандарте. Определяет структуру стандарта и как соотносятся друг с другом различные части. В этой части также описываются установленные комитетом уровни соответствия стандарту.

*Часть 2. SQL/Foundation*

Содержит базовые требования (расширенный раздел из SQL99 Core). Самая большая и наиболее важная часть стандарта.

*Часть 3. SQL/CLI (Call-Level Interface)*

Определяет интерфейс уровня вызовов для динамического использования SQL из внешних приложений. Также включает спецификацию более чем 60 процедур и функций, что способствует написанию действительно переносимых приложений-оберток.

*Часть 4. SQL/PSM (Persistent Stored Modules)*

Стандартизирует процедурные конструкции языка, похожие на некоторые диалекты SQL, такие как PL/SQL или Transact-SQL.

*Часть 9. SQL/MED (Management of External Data)*

Определяет порядок работы с данными, расположенными вне базы данных, с помощью указателей и интерфейсов-оберток

*Часть 10. SQL/OBJ (Object Language Binding)*

Описывает использование SQL из программ, написанных на Java. Тесно связано с JDBC, но имеет несколько преимуществ. Кроме того, серьезно отличается от описанного в предыдущем стандарте способа использования SQL из другого языка.

*Часть 11. SQL/Schemata*

Определяет 85 представлений (на 3 больше, чем в SQL99), используемых для хранения метаданных базы данных и хранящихся в специальной схеме с названием *INFORMATION_SCHEMA*. Несколько представлений изменены по сравнению с SQL99.

*Часть 12. SQL/JRT (Java Routines and Types)*

Определяет функции и типы языка Java. Теперь поддерживаются такие возможности Java, как статические методы и классы.

*Часть 14. SQL/XML*

Вводит новый тип с названием XML, четыре новых оператора (*XMLPARSE*, *XMLSERIALIZE*, *XMLROOT* и *XMLCONCAT*), несколько новых функций (описанных в главе 4) и новый предикат *IS DOCUMENT*. Также включает правила для отображения элементов SQL **(идентификаторов, схем и объектов)** на элементы XML.

Обратите внимание, что частей с номерами 5, 6, 7 и 8 в стандарте нет.

Важно понимать, что разработчик может заявлять об ANSI-совместимости своей СУБД, удовлетворяя только базовым требованиям, поэтому нужно читать то, что написано «мелким шрифтом», – подробный перечень поддерживаемых возможностей. Зная, какие возможности описываются в девяти дополнительных пакетах стандарта, вы можете понять возможности конкретной СУБД, а также особенности переноса SQL-кода на другую платформу.

Стандарт ANSI, описывающий извлечение, манипуляцию и управление данными с помощью команд SELECT, JOIN, ALTER TABLE, DROP и других, формализует поведение команд и синтаксические структуры для множества платформ. Этот стандарт стал еще более важным с появлением СУБД с открытым исходным кодом (таких как MySQL или PostgerSQL), которые становятся достаточно популярными и разрабатываются не крупными компаниями, а виртуальными командами разработчиков.

В этом справочнике описывается реализация SQL четырьмя популярными реляционными СУБД. Эти платформы не поддерживают стандарт SQL3 полностью; фактически разработчики постоянно играют с комитетом по стандартам в догонялки. Часто, как только разработчики приближаются к стандарту, комитет обновляет, уточняет или еще как-то меняет стандарт. С другой стороны, разработчики часто реализуют в своих СУБД возможности, еще не описанные в стандарте, но значительно увеличивающие эффективность работы пользователей.

## Классы операторов SQL3

Способ разбиения операторов SQL на классы существенно различается в SQL92 и SQL3. Однако старые термины продолжают использоваться, поэтому читателю необходимо иметь о них представление. SQL92 делит операторы на три большие категории:

*Язык манипулирования данными (Data Manipulation Language, DML)*

Содержит команды для манипулирования с данными, такие как *SELECT*, *INSERT*, *UPDATE* и *DELETE*.

*Язык определения данных (Data Definition Language, DDL)*

Содержит команды для работы с объектами базы данных, такие как *CREATE* и *DROP*.

*Язык управления данными (Data Control Language, DCL)*

Содержит команды для управления привилегиями, такие как *GRANT* и *REVOKE*.

SQL3 использует 7 категорий, называемых теперь *классами*, в качестве общего подхода к делению на типы всех команд SQL. Эти «классы» операторов немного отличаются от категорий операторов в SQL92, так как более аккуратно и логично определяют перечень операторов, попадающих в каждый класс, а также позволяют дальнейшее расширение возможностей и списка классов. Также в новые классы попадают некоторые операторы, которые в SQL92 не вписывались достаточно точно ни в одну из категорий.

В табл. 1.1 приводится перечень классов операторов SQL3, и для каждого класса приводятся примеры операторов, рассматриваемых позднее более подробно. В данный момент достаточно только запомнить названия классов.

*Таблица 1.1. Классы операторов SQL3*

| Класс | Описание | Примеры команд |
|---|---|---|
| Операторы подключения | Создают и закрывают клиентское подключение | *CONNECT*, *DISCONNECT* |
| Операторы управления выполнением | Управляют выполнением набора SQL операторов | *CALL*, *RETURN* |
| Операторы работы с данными | Извлекают и изменяют данные | *SELECT*, *INSERT*, *UPDATE*, *DELETE* |
| Диагностические операторы | Обеспечивают диагностической информацией, сообщают об исключениях и ошибках | *GET DAGNOSTINCS* |
| Операторы управления схемой | Управляют объектами в схеме и внутри схемы базы данных | *ALTER*, *CREATE*, *DROP* |
| Операторы управления сессиями | Управляют поведением по умолчанию и другими параметрами сессий | *SET CONSTRAINT* |
| Операторы управления транзакциями | Начинают и заканчивают транзакции | *COMMIT*, *ROLLBACK* |

Постоянно работающим с SQL необходимо быть в курсе как старых (SQL92), так и новых (SQL3) классов, так как оба варианта еще используются при обращении к SQL-операторам.

## Диалекты SQL

Постоянно эволюционирующий SQL породил множество диалектов, используемых на разнообразных платформах. Причиной появления диалектов является то, что пользователям СУБД необходимы новые функции раньше, чем ANSI выпустит новый стандарт. Иногда научное и исследовательское сообщество добавляет в стандарт новые возможности под давлением со стороны конкурирующих на рынке технологий. Например, многие разработчики расширяют возможности программирования в своих СУБД при помощи Java (как в случае с DB2, Oracle и Sybase) или VBScript (в случае Microsoft). В будущем разработчики будут использовать эти языки совместно с SQL для разработки SQL-программ.

Многие диалекты содержат условные операторы (такие как *IF...THEN*), возможности управления потоком выполнения (циклы *WHILE*), переменные и возможности обработки ошибок. Так как всех этих возможностей не было в стандарте на момент возникновения потребности в них со стороны пользователей, то разработчики предложили свои собственные команды и синтаксис. В реальности, у некоторых разработчиков из ранних 80-х есть различия в синтаксисе даже элементарных команд (например, *SELECT*), так как их реализации предшествовали стандарту. ANSI в данное время уточняет стандарт для устранения этих несоответствий.

В некоторых диалектах представлена функциональность, делающая их более близкими к языкам процедурного программирования. К примеру, процедурные возможности этих диалектов SQL могут содержать команды обработки ошибок, команды управления потоком выполнения, условные команды, переменные, массивы и многое другое. В этой книге мы все эти возможности будем называть диалектами, хотя они, по сути, являются процедурными расширениями SQL. Пакет SQL/PSM из стандарта описывает различные возможности, связанные с хранимыми процедурами, и содержит многие расширения, предлагаемые диалектами.

Вот несколько распространенных диалектов:

*PL/SQL*

Используется в Oracle. Сокращение обозначает Procedural Language/SQL. Диалект во многом похож на язык программирования Ada.

*Transact-SQL*

Используется в Microsoft SQL Server и Sybase Adaptive Server. С тех пор как в начале 1990-х Microsoft и Sybase начали развивать разные платформы, их реализации Transact-SQL также начали отличаться.

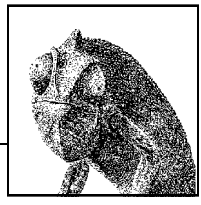
*PL/pgSQL*

Расширение SQL, реализованное в PostgreSQL. Сокращение расшифровывается как Procedural Language/PostgreSQL.

Пользователи, планирующие плотно работать с одной СУБД, должны изучить все особенности диалекта, используемого этой платформой.

# 2

# Основные концепции



SQL является простым и интуитивно понятным способом взаимодействия с базой данных. Хотя в стандарте SQL2003 не дается определение «базы данных», стандарт определяет основные функции и концепции для создания, извлечения, изменения и удаления данных. Важно знать синтаксис, используемый в стандарте, а также специфические особенности конкретных платформ. В этой главе рассматриваются необходимые основы синтаксиса SQL.

## Рассматриваемые СУБД

В этом издании справочника описывается стандарт SQL и его реализация несколькими ведущими поставщиками реляционных СУБД:

*MySQL*

MySQL – популярная СУБД с открытым кодом, которая известна своей молниеносной производительностью. СУБД работает на множестве операционных систем, включая большинство Linux-систем. По причине основного акцента на скорости работы она поддерживает набор возможностей уже, чем у конкурентов. В этой книге рассматривается MySQL версии 5.1.

*Oracle*

Oracle является лидирующей СУБД на рынке. Работает практически на любых операционных системах и серверах. Благодаря своей масштабируемой и надежной архитектуре эта СУБД стала выбором многих пользователей. В этом издании мы рассматриваем Oracle 11g.

*PostgreSQL*

PostgreSQL является системой с самыми богатыми возможностями среди всех СУБД с открытым кодом. В то время как MySQL известен высокой производительностью, PostgreSQL снискал славу за прекрасную поддержку стандарта ANSI, надежную работу с транзакциями, богатый набор поддерживаемых типов данных и объектов. В дополнение к богатым возможностям PostgreSQL также поддерживает широкий список операционных систем и аппаратных платформ. В этой книге рассматривается PostgreSQL 8.2.1.

*SQL Server*

Microsoft SQL Server является популярной СУБД, доступной только под операционную систему Windows. Особенностями СУБД являются простота использования, широчайший набор возможностей, низкая стоимость и высокая производительность. В этой книге мы рассматриваем Microsoft SQL Server 2008.

# Категории синтаксиса

Чтобы начать использовать SQL, необходимо понять, как пишутся операторы. Синтаксис SQL разбивается на четыре основных категории. Категории перечисляются в следующем списке, а затем детально рассматриваются в дальнейших разделах:

*Идентификаторы*

Пользовательское или системное имя объекта базы данных, например самой базы данных, таблицы, ограничения, столбца, представления и т. п.

*Литералы*

Любое пользовательское или системное значение, не являющееся идентификатором или ключевым словом. Литералы могут быть строковыми (*«привет»*), числовыми (*1234*), датами (*1 января 1980*), или булевыми (*TRUE*).

*Операторы*

Обозначения, описывающие действия, которые необходимо выполнить над одним или несколькими выражениями. Операторы, как правило, используются в командах INSERT, DELETE, SELECT и UPDATE. Также операторы часто используются при создании объектов в базе данных.

*Зарезервированные и ключевые слова*

Имеют особое значение для синтаксического анализатора SQL. *Ключевые слова*, такие как *SELECT*, *GRANT*, *DELETE* или *CREATE*, не могут использоваться в СУБД в качестве идентификаторов. Обычно они являются командами или SQL-операторами. *Зарезервированные слова* – это слова, которые могут в будущем стать ключевыми словами. В дальнейшем при разговоре о любой из этих двух концепций мы будем употреблять только термин *ключевое слово*. Вы можете обойти ограничение на запрет использования ключевых слов в качестве идентификаторов при помощи идентификаторов, взятых в кавычки (обсуждаемых далее). Но так поступать не рекомендуется.

## Идентификаторы

Помните, что реляционные СУБД построены на базе теории множеств. В терминах ANSI *кластеры* содержат множества каталогов, *каталоги* содержат множества схем, *схемы* содержат множества объектов и так далее. Большинство СУБД используют похожие термины: *экземпляры* СУБД содержат одну или несколько баз данных, *базы данных* содержат одну или несколько схем, *схемы* содержат одну или несколько таблиц, представлений, хранимых процедур или привилегий доступа к объектам. На каждом уровне этой иерархии элементам требуются уникальные названия (идентификаторы), чтобы к этим элементам можно было

обращаться из программ или системных процессов. Это значит, что каждый объект (независимо от того, является ли он базой данных, таблицей, представлением, столбцом, индексом, ключом, триггером, хранимой процедурой или ограничением) должен быть идентифицирован. При запуске команды на создание объекта вы должны указать идентификатор (т. е. имя) этого нового объекта.

Опытные программисты при выборе идентификаторов руководствуются двумя наборами правил:

*Правила именования*

Логические правила, используемые проектировщиками баз данных на практике. Постоянное следование этим правилам приводит к более ясной и управляемой структуре базы данных. Это не столько требования SQL, сколько результат жизненного опыта программистов.

*Ограничения на идентификаторы*

Ограничения, наложенные стандартом SQL и различными реализациями. Например, это могут быть ограничения на допустимую длину идентификатора. Подробнее ограничения каждой СУБД рассматриваются далее в этом разделе.

## Правила именования

Правила именования описывают базовые соображения, которыми необходимо руководствоваться при выборе идентификатора объекта. В этом разделе мы приводим правила именования, созданные на базе нашего многолетнего опыта. Стандарт SQL не накладывает никаких ограничений, кроме ограничений на уникальность и длину идентификатора и на набор допустимых к использованию символов. Но вам желательно придерживаться также следующих правил:

*Выбирайте осмысленные и содержательные названия*

Не называйте таблицу **XP05**; вместо этого используйте название **Expences_2005** для указания того, что в таблице хранятся расходы за 2005 год. Помните, что, возможно, спустя долгое время после вашего ухода другие люди будут пользоваться этой базой данных и этой таблицей, и введенные вами имена объектов должны быть понятными с первого взгляда. Все СУБД имеют ограничения на длину идентификатора, но обычно идентификатор можно сделать достаточно длинным и понятным каждому.

*Используйте везде один и тот же регистр символов*

Для именования всех объектов используйте либо только верхний регистр символов, либо только нижний. Некоторые СУБД чувствительны к регистру символов, и идентификаторы со смешанными регистрами символов могут впоследствии стать источником проблем.

*Будьте последовательны в применении аббревиатур*

Однажды выбрав сокращение, используйте его повсеместно. Например, однажды употребив сокращение *EMP* вместо *EMPLOYEE*, используйте его везде; не нужно в одном месте употреблять *EMP*, а в другом *EMPLOYEE*.

*Для лучшей читаемости используйте подчеркивания*

Название столбца *UPPERCASEWITHUNDERSCORES* читается не так легко, как *UPPERCASE_WITH_UNDERSCORES*.

*Не используйте в идентификаторах название компании или продукта*

Компании приобретаются, а продукты меняют названия. Эти вещи слишком изменчивы, чтобы быть включенными в названия объектов баз данных.

*Не используйте слишком очевидные суффиксы и префиксы*

К примеру, не используйте префикс **DB_** для названия базы данных, и не начинайте имя каждого представления с **V_**. Простой запрос к системным таблицам базы данных ответит администратору или программисту на вопрос, какой тип объекта представляется идентификатором.

*Не используйте для названия объекта всю доступную длину*

Если СУБД позволяет задействовать в имени таблицы до 32 символов, то постарайтесь оставить хотя бы несколько свободных символов про запас. Некоторые СУБД добавляют префиксы или суффиксы к имени таблицы, когда оперируют временными наборами данных.

*Не используйте идентификаторы, заключенные в кавычки*

Идентификаторы объектов можно заключать в двойные кавычки (в ANSI это называется *идентификаторами с разделителями*). Такие идентификаторы являются регистрозависимыми. Использование кавычек может привести к созданию идентификаторов, которые окажутся сложными в использовании или впоследствии породят проблемы. Например, кавычки позволят вставить в идентификатор пробелы, специальные символы, символы разных регистров и даже управляющие символы, но многие инструменты сторонних производителей (и даже инструменты самих разработчиков СУБД) не могут корректно работать с такими именами объектов. Поэтому не следует использовать идентификаторы, заключенные в кавычки.

В некоторых СУБД для идентификаторов с разделителями можно использовать не только кавычки. Например, в SQL Server для тех же целей можно также использовать квадратные скобки (*[ ]*).

Приведем основные преимущества следования правилам именования. Во-первых, ваш SQL-код становится в некоторой мере самодокументирующимся, потому что используемые имена являются осмысленными и понятными другим пользователям. Во-вторых, ваш SQL-код и базу данных с однотипно названными объектами будет легче поддерживать – особенно тем людям, которые будут делать это после вас. В-третьих, следование правилам именования повышает производительность. Если базу данных вдруг придется переводить на другую платформу, то согласованные и осмысленные названия сэкономят силы и время. Потратив в начале работы несколько минут на размышления о выборе имен, вы избежите проблем в будущем.

## Ограничения на идентификаторы

Ограничения на идентификаторы – это ограничения, накладываемые конкретной платформой СУБД. Эти ограничения касаются только обычных идентификаторов, а не заключенных в кавычки. Ограничения стандарта SQL2003 отличаются от тех, что реализованы разработчиками. В табл. 2.1 проводится сравнение

ограничений, накладываемых стандартом и конкретными реализациями СУБД, описываемых в книге.

*Таблица 2.1. Ограничения на идентификаторы объектов (за исключением идентификаторов, заключенных в двойные кавычки)*

| Характеристика | Платформа | Описание |
|---|---|---|
| Длина идентификатора | SQL3 | 128 символов |
| | MySQL | 64 символа, псевдонимы до 128 символов |
| | Oracle | 30 байт (число символов зависит от кодовой страницы), имена баз данных – до 8 байт, имена db-link – до 128 байт |
| | PostgreSQL | 63 символа (свойство *NAMEDATALEN*-1) |
| | SQL Server | 128 символов, временные таблицы – до 116 символов |
| Идентификатор может содержать | SQL3 | Любую цифру, букву и подчеркивания (_) |
| | MySQL | Любую цифру, букву, символ. Не может состоять только из цифр. |
| | Oracle | Любую цифру, букву, подчеркивание, символ решетку (#), знак доллара ($), хотя два последних символа использовать не рекомендуется. Имя db-link может также содержать точку. |
| | PostgreSQL | Любую цифру, букву и подчеркивания (_) |
| | SQL Server | Любую цифру, букву, символ at (@), символ решетку (#) и знак доллара ($). |
| Идентификатор должен начинаться | SQL3 | С буквы. |
| | MySQL | С буквы или цифры. Не может состоять только из цифр. |
| | Oracle | С буквы. |
| | PostgreSQL | С буквы или подчеркивания (_). |
| | SQL Server | С буквы, подчеркивания (_), символа at (@), символа решетки (#). |
| Идентификатор не может содержать | SQL3 | Пробелов и специальных символов. |
| | MySQL | Точку (.), косую черту (/), и символы ASCII(0) и ASCII(255). Одинарные (‘ ’) и двойные (" ") кавычки допустимы только в идентификаторах с ограничителями, идентификатор не может заканчиваться на пробел. |
| | Oracle | Пробелы, двойные кавычки (" ") и специальные символы. |
| | PostgreSQL | Двойные кавычки (" "). |
| | SQL Server | Пробелы и специальные символы. |

| Характеристика | Платформа | Описание |
|---|---|---|
| Поддержка идентификаторов с разделителями | SQL3 | Есть. |
| | MySQL | Есть. |
| | Oracle | Есть. |
| | PostgreSQL | Есть. |
| | SQL Server | Есть. |
| Разделитель, используемый с идентификаторами | SQL3 | Двойные кавычки (" "). |
| | MySQL | Одинарные кавычки (‘ ’) или двойные кавычки (" ") в режиме ANSI совместимости. |
| | Oracle | Двойные кавычки (" "). |
| | PostgreSQL | Двойные кавычки (" "). |
| | SQL Server | Двойные кавычки (" ") или квадратные скобки ([]); скобки предпочтительнее. |
| В качестве идентификатора допустимо ключевое слово | SQL3 | Нет, если только идентификатор не заключен в кавычки. |
| | MySQL | Нет, если только идентификатор не заключен в кавычки. |
| | Oracle | Нет, если только идентификатор не заключен в кавычки. |
| | PostgreSQL | Нет, если только идентификатор не заключен в кавычки. |
| | SQL Server | Нет, если только идентификатор не заключен в кавычки. |
| Схема адресации | SQL3 | *Каталог.Схема.Объект* |
| | MySQL | *База данных.Объект* |
| | Oracle | *Схема.Объект* |
| | PostgreSQL | *База данных.Схема.Объект* |
| | SQL Server | *Сервер.База данных.Схема.Объект* |
| Идентификатор должен быть уникальным | SQL3 | Да. |
| | MySQL | Да. |
| | Oracle | Да. |
| | PostgreSQL | Да. |
| | SQL Server | Да. |
| Регистрозависимость | SQL3 | Нет. |
| | MySQL | Только если регистрозависима операционная система (например, MacOS или Unix). Триггеры, группы журнальных файлов и табличные пространства всегда регистрозависимы. |

*Таблица 2.1 (продолжение)*

| Характеристика | Платформа | Описание |
|---|---|---|
| | Oracle | По умолчанию нет, но можно поменять. |
| | PostgreSQL | Нет. |
| | SQL Server | По умолчанию нет, но можно поменять. |
| Другие ограничения | SQL3 | Нет. |
| | MySQL | Не может состоять только из цифр. |
| | Oracle | Идентификаторы db-link не могут быть длиннее 128 байт и заключаться в кавычки. |
| | PostgreSQL | Нет. |
| | SQL Server | Для идентификаторов с разделителями Microsoft предпочитает использовать квадратные скобки, а не двойные кавычки. |

Идентификаторы должны быть уникальны в пределах своей области видимости. Если вспомнить рассмотренную ранее иерархию объектов базы данных, то имена баз данных должны быть уникальны в пределах экземпляра СУБД, в то время как имена таблиц, представлений, функций, триггеров и хранимых процедур должны быть уникальны в пределах конкретной схемы. С другой стороны, таблица и хранимая процедура *могут* иметь одно и то же имя, так как являются объектами разного типа. Имена столбцов, ключей и индексов должны быть уникальны в пределах таблицы или представления и так далее. За более подробной информацией обратитесь к документации по вашей СУБД – некоторые платформы требуют уникальности идентификаторов в определенных ситуациях, другие – не требуют. Например, в Oracle требуется уникальность имен индексов в пределах базы данных, в то время как другие СУБД (например, SQL Server) требуют уникальности имени индекса только в пределах таблицы, по которой этот индекс построен.

Помните, что идентификаторы, заключенные в кавычки, могут нарушать некоторые из описанных ранее правил. Например, такие идентификаторы являются чувствительными к регистру – идентификатор «foo» не равен идентификатору «FOO». Более того, с кавычками в качестве идентификаторов можно использовать зарезервированные слова или запрещенные для использования буквы и символы. К примеру, использование знака процента (%), как правило, запрещено. Тем не менее, в случае особой необходимости, всегда можно использовать знак процента, заключив идентификатор в двойные кавычки. То есть, чтобы назвать таблицу **expense%%ratios**, надо заключить название в двойные кавычки: «**expense%%rates**». Еще раз напомним, что в SQL3 такие идентификаторы называются идентификаторами с разделителями.

Если объект создан с именем, заключенным в кавычки, то мы рекомендуем всегда обращаться к этому объекту с помощью идентификатора с разделителями.

## Литералы

В SQL литералом называется любое явно заданное числовое значение, символьная строка, временное значение (дата или время) или булево значение, не являющееся идентификатором или ключевым словом. Реляционные СУБД допускают множество различных литералов в SQL программах. Литералы допустимы для большинства числовых, символьных, булевых и временных типов данных. К примеру, в SQL Server числовые типы данных включают (в числе прочих) типы *INTEGER*, *REAL* и *MONEY*. *Числовые литералы* могут выглядеть следующим образом:

```
30
-117
+883.3338
-6.66
$70000
2E5
7E-3
```

Как показывает этот пример, в SQL Server допускается использование знаковых и беззнаковых чисел в обычной и экспоненциальной записи. А так как в SQL Server есть тип данных *MONEY*, то можно даже использовать знак доллара (*$*). SQL Server *не* допускает использования любых других символов в числовых литералах (кроме 0 1 2 3 4 5 6 7 8 9 + - $ . E e), поэтому не пытайтесь использовать запятую (запятая используется в некоторых европейских странах для отделения десятичных разрядов от целой части). Большинство СУБД интерпретируют запятую в числовых литералах как *разделитель списка значений*. Таким образом, литерал вида 3,000 вероятнее всего будет воспринят как два отдельных значения: 3 и 000.

Булевы, символьные и временные литералы могут выглядеть следующим образом:

```
TRUE
'Hello world!'
'OCT-28-1966 22:14:30:00'
```

Символьные литералы всегда должны быть заключены в одинарные кавычки (‘ ’). Это стандартный ограничитель для символьных литералов. В символьных литералах можно использовать не только буквы алфавита. На самом деле, в литерале можно использовать любой символ из таблицы символов. Все это примеры символьных литералов:

```
'1998'
'70,000 + 14000'
'There once was a man from Nantucket,'
'Oct 28, 1966'
```

и все эти значения совместимы с типом данных *CHAR*. Не путайте символьный литерал ‘1998’ с числовым литералом 1998. Так как символьные литералы связаны с символьным типом данных, то неправильно использовать их в арифметических выражениях без явного преобразования типов данных. Некоторые (но не

все) СУБД выполняют автоматическое приведение типов данных к DATE или NUMBER для символьных литералов, содержащих цифры.

Удваивая ограничитель, вы можете при необходимости представить одинарную кавычку в символьном литерале. То есть вы можете использовать две одинарные кавычки каждый раз, когда вам требуется одинарная кавычка в строке. Вот пример для SQL Server, демонстрирующий описанную идею:

```
SELECT 'So he said ''Who''s Le Petomaine?'''
```

Результат выполнения запроса следующий:

```
-----------------
So he said 'Who's Le Petomaine?'
```

## Операторы

Оператором называется символ, указывающий какой действие нужно выполнить с одним или несколькими *выражениями*. Операторы чаще всего используются в командах *DELETE*, *INSERT*, *SELECT* и *UPDATE*, но часто встречаются также при создании объектов базы данных, таких как хранимые процедуры, функции, триггеры и представления.

Обычно операторы делят на следующие категории:

*Арифметические операторы*

Поддерживаются всеми СУБД.

*Операторы присваивания*

Поддерживаются всеми СУБД.

*Битовые операторы*

Поддерживаются MySQL и SQL Server.

*Операторы сравнения*

Поддерживаются всеми СУБД.

*Логические операторы*

Поддерживаются всеми СУБД.

*Унарные операторы*

Поддерживаются MySQL, Oracle и SQL Server.

### Арифметические операторы

*Арифметические операторы* выполняют математические действия над двумя числовыми выражениями. В табл. 2.2 представлен список арифметических операторов.

В MySQL, Oracle и SQL Server операторы + и - могут использоваться для выполнения арифметических операций с датами и временем. На различных платформах есть свои, уникальные методы работы с датами.

*Таблица 2.2. Арифметические операторы*

| Арифметический оператор | Действие |
|---|---|
| + | Сложение |
| - | Вычитание |
| * | Умножение |
| / | Деление |
| % | Остаток при целочисленном делении (только в SQL Server) |

## Операторы присваивания

За исключением Oracle, использующего :=, *оператор присваивания* (=) присваивает значение переменной или псевдониму столбца. Во всех СУБД, рассматриваемых в этой книге, ключевое слово *AS* используется для присваивания псевдонима таблице или столбцу.

## Битовые операторы

И в Microsoft SQL Server, и в MySQL имеются *битовые операторы* (табл. 2.3) для битовых манипуляций над целочисленными выражениями. Битовые операторы работают с типами данных *BINARY*, *BIT*, *INT*, *SMALLINT*, *TINYINT* и *VARBINARY*. PostgreSQL поддерживает типы данных *BIT* и *BIT VARYING*; а также битовые операторы *AND*, *OR*, *XOR*, конкатенацию, *NOT* и сдвиги влево и вправо.

*Таблица 2.3. Битовые операторы*

| Битовый оператор | Значение |
|---|---|
| & | Битовое *И* (два операнда) |
| \| | Битовое *ИЛИ* (два операнда) |
| ^ | Битовое исключающее *ИЛИ* (два операнда) |

## Операторы сравнения

С помощью *операторов сравнения* проверяется равенство или неравенство двух выражений. Результатом операторов сравнения является булево выражение: *TRUE*, *FALSE* или *UNKNOWN*. Обратите внимание, что согласно стандарту ANSI оператор сравнения возвращает NULL, если один из операндов также равен NULL. К примеру, выражение *23+NULL* равно NULL, так же как и выражение *23 февраля 2002 + NULL*. В табл. 2.4 приведен список операторов сравнения.

Булевы операторы сравнения используются чаще всего во фразе *WHERE* для отбора строк, удовлетворяющих условиям поиска. В следующем примере используется оператор «больше или равно»:

```
SELECT *
   FROM Products
   WHERE ProductID >= 347
```

*Таблица 2.4. Операторы сравнения*

| Оператор сравнения | Значение |
|---|---|
| = | Равно |
| > | Больше |
| < | Меньше |
| >= | Больше или равно |
| <= | Меньше или равно |
| <> | Не равно |
| *!=* | Не равно (не по стандарту ANSI) |
| *!<* | Не меньше (не по стандарту ANSI) |
| *!>* | Не больше (не по стандарту ANSI) |

## Логические операторы

*Логические операторы* обычно используются во фразе *WHERE* для проверки истинности различных условий. Они возвращают либо булево значение *TRUE*, либо *FALSE*. В табл. 2.5 приводится список булевых операторов. Следует помнить, что не все СУБД поддерживают полный перечень этих операторов.

*Таблица 2.5. Логические операторы*

| Логический оператор | Значение |
|---|---|
| *ALL* | TRUE, если истинно каждое условие из множества |
| *AND* | TRUE, если оба булевых выражения истинны |
| *ANY* | TRUE, если истинно хотя бы одно условие из множества |
| *BETWEEN* | TRUE, если операнд попадает в заданный интервал |
| *EXISTS* | TRUE, если подзапрос возвращает хотя бы одну строку |
| *IN* | TRUE, если операнд равен хотя бы одному выражению из списка или строке из результата подзапроса |
| *LIKE* | TRUE, если операнд удовлетворяет шаблону |
| *NOT* | Заменяет булево значение на противоположное |
| *OR* | TRUE, если хотя бы одно из двух булевых выражений истинно |
| *SOME* | TRUE, если истинны какие-либо условия из множества |

## Унарные операторы

*Унарные операторы* используются с одним операндом, являющимся выражением любого числового типа. Исключением является битовый унарный оператор (~), используемый только с целочисленными выражениями (табл. 2.6).