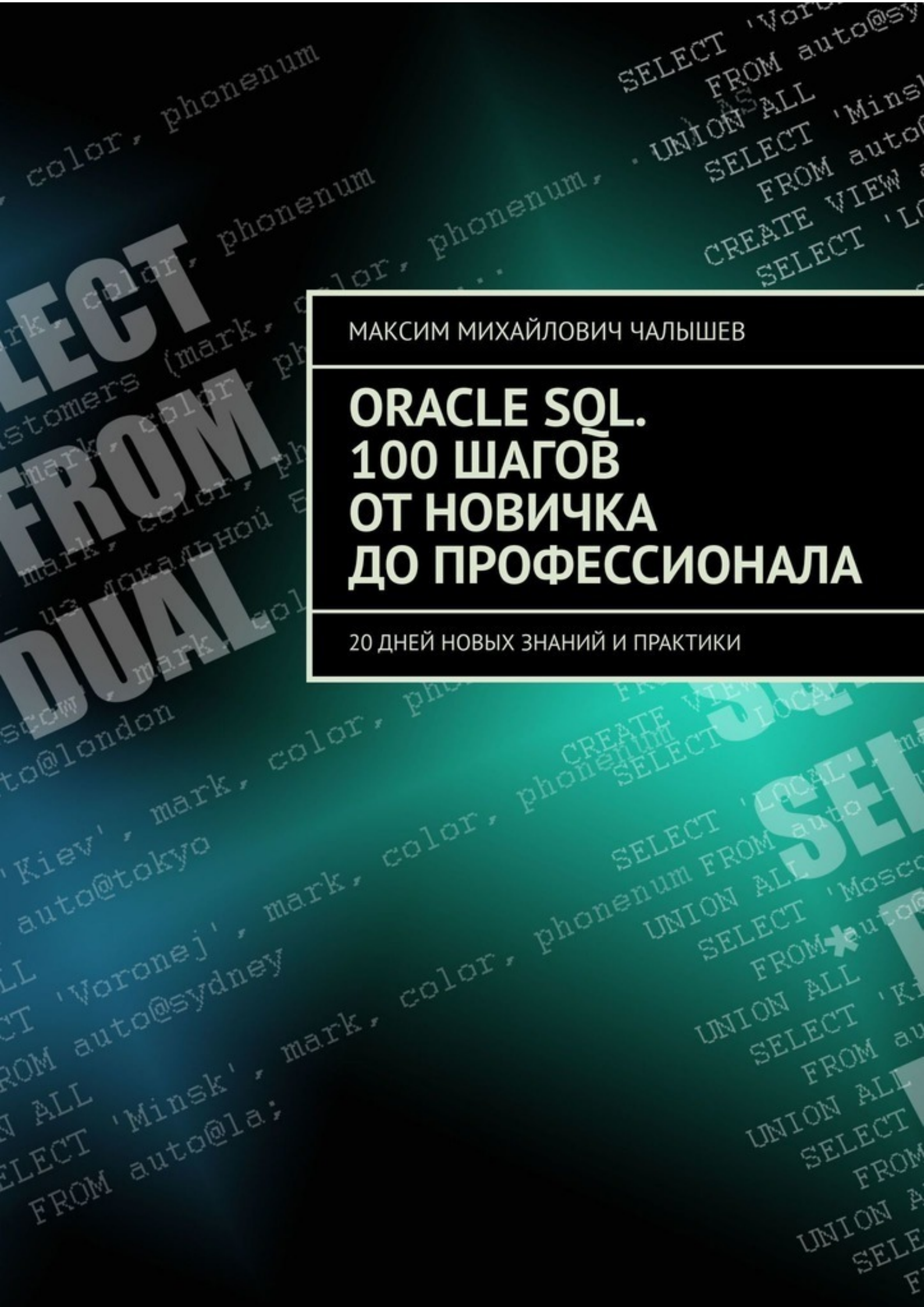


МАКСИМ МИХАЙЛОВИЧ ЧАЛЫШЕВ

ORACLE SQL. 100 ШАГОВ ОТ НОВИЧКА ДО ПРОФЕССИОНАЛА

20 ДНЕЙ НОВЫХ ЗНАНИЙ И ПРАКТИКИ

Максим Чалышев

Oracle SQL. 100 шагов от новичка до профессионала. 20 дней новых знаний и практики

«Издательские решения»

Чалышев М. М.

Oracle SQL. 100 шагов от новичка до профессионала. 20 дней новых знаний и практики / М. М. Чалышев — «Издательские решения»,

ISBN 978-5-00-500618-9

Более 300 ответов на вопросы.Более 500 практических заданий.Более 1000 разобранных примеров.Учебник справочник по языку SQL.

ISBN 978-5-00-500618-9

© Чалышев М. М.
© Издательские решения

Содержание

Введение 9
День первый 11
Шаг 1. Что такое SQL, назначение языка 12
- Что такое базы данных, назначение баз данных 13
- Веб-технологии 14
- Мобильные устройства 15
- Игры 16
- Крупные корпорации 17
- Назначение языка SQL, необходимость изучения этого языка 18
- Вопросы учеников 19

Шаг 2. Теория и практика. Учебная схема данных. Организация работы 20
- Общая схема процесса обучения, или Как читать данную книгу 21
- Несколько терминов 22
- Учебная схема 23
- Вопросы учеников 26

Шаг 3. Подготовка к работе. Процесс обучения. Описание интерфейса ORACLE APEX 27
- Подготовка к работе 27
- Процесс обучения 28
- Составьте карточки 29
- Составляйте свой список вопросов 30
- Создайте свое задание 31
- Интерфейс ORACLE APEX 32
- Просмотр структуры таблиц 33
- Просмотр кода процедур 34
- Вопросы учеников 35
- Контрольные вопросы и задания для самостоятельного выполнения 36

Шаг 4. Таблицы в базе данных 37
- Теория 37
- Таблицы нашей учебной схемы 38
- Вопросы учеников 40
- Контрольные вопросы и задания для самостоятельного выполнения 41

Шаг 5. Типы данных 42
- Теория и практика 43
- Важные замечания 46
- Вопросы учеников 47
- Контрольные вопросы и задания для самостоятельного выполнения 48

День второй 49
Шаг 6. Создание таблиц 50
- Важные замечания 51
- Теория и практика 52
- Вопросы учеников 54

Примеры 55
Примеры 56
Контрольные вопросы и задания для самостоятельного выполнения 57
Шаг 7. Структура таблицы 58
Введение 58
Теория и практика 59
Удаление колонки 60
Важные замечания 61
Вопросы учеников 62
Контрольные вопросы и задания для самостоятельного выполнения 63
Шаг 8. Первичные ключи. Вторичные ключи 64
Введение 64
Теория и практика 65
Важные замечания 66
Вопросы учеников 67
Контрольные вопросы и задания для самостоятельного выполнения 68
Шаг 9. Ограничения 69
Введение 69
Теория и практика 70
1. Ограничения на вставку пустых значений NOT NULL 71
Ограничения на уникальность 72
Ограничения на вторичный ключ 73
Важные замечания 74
Вопросы учеников 75
Контрольные вопросы и задания для самостоятельного выполнения 76
Шаг 10. Индексы 77
Введение 77
Теория и практика 78
Важные замечания 79
Вопросы учеников 80
Контрольные вопросы и задания для самостоятельного выполнения 81
День третий 82
Шаг 11. Простые запросы 83
Введение 83
Теория и практика 84
Важные замечания 90
Вопросы учеников 91
Контрольные вопросы и задания для самостоятельного выполнения 92
Шаг 12. Более сложные условия. Знакомимся с логикой выбора строк 93
Введение 93
Теория и практика 94
Важные замечания 98
Вопросы учеников 99

Контрольные вопросы и задания для самостоятельного выполнения 100
Шаг 13. Сортировка результатов запросов 101
Введение 101
Теория и практика 102
Важные замечания 105
Вопросы учеников 106
Контрольные вопросы и задания для самостоятельного выполнения 107
Шаг 14. Ограничение на количество выбранных строк ROWNUM, TOP (n) 108
Введение 108
Теория и практика 109
Синтаксис 110
Важные замечания 111
Вопросы учеников 112
Контрольные вопросы и задания для самостоятельного выполнения 113
Шаг 15. Вставка данных в таблицу – INSERT 114
Введение 114
Теория и практика 115
Важные замечания 117
Вопросы учеников 118
Контрольные вопросы и задания для самостоятельного выполнения 119
День четвертый 120
Шаг 16. Обновление данных – UPDATE 121
Введение 121
Теория и практика 122
Синтаксис команды 123
Важные замечания 125
Вопросы учеников 126
Контрольные вопросы и задания для самостоятельного выполнения 127
Шаг 17. Удаление данных – DELETE 128
Введение 128
Теория и практика 129
Синтаксис 130
Важные замечания 131
Вопросы учеников 132
Контрольные вопросы и задания для самостоятельного выполнения 133
Шаг 18. Псевдонимы 134
Введение 134
Теория и практика 135
Важные замечания 137
Вопросы учеников 138
Контрольные вопросы и задания для самостоятельного выполнения 139

Шаг 19. BETWEEN 140
Введение 140
Теория и практика 141
Важные замечания 143
Вопросы учеников 144
Контрольные вопросы и задания для самостоятельного выполнения 146
Шаг 20. DISTINCT, дубликаты значений 147
Введение 147
Теория и практика 148
Важные замечания 150
Вопросы учеников 151
Контрольные вопросы и задания для самостоятельного выполнения 152
День пятый 153
Шаг 21. Математика в запросах 154
Введение 154
Теория и практика 155
Важные замечания 158
Вопросы учеников 159
Контрольные вопросы и задания для самостоятельного выполнения 160
Шаг 22. Запрос к результату выражения и специальная таблица DUAL 161
Введение 161
Теория и практика 162
Важные замечания 164
Вопросы учеников 165
Контрольные вопросы и задания для самостоятельного выполнения 166
Конец ознакомительного фрагмента. 167

Oracle SQL. 100 шагов от новичка до профессионала 20 дней новых знаний и практики

Максим Михайлович Чалышев

SQL.100 шагов от новичка до профессионала

© Максим Михайлович Чалышев, 2019

ISBN 978-5-0050-0618-9
Создано в интеллектуальной издательской системе Ridero

20 дней новых знаний и практики.
Более 300 ответов на вопросы.
Более 500 практических заданий.
Более 1000 разобранных примеров.

Данную книгу я посвящаю своим друзьям:
Кузнецову Алексею – профессионалу управления в сфере ИТ,
Коршакову Артему – будущему высококлассному IT-специалисту.

Чалышев Максим

Введение

Приветствую. Сначала как автор этой книги расскажу немного о своем профессиональном опыте. На данный момент я работаю в сфере информационных технологий уже почти 20 лет.

Основной моей специализацией в ИТ были и остаются базы данных и, прежде всего, СУБД ORACLE.

В первый раз я познакомился с данной СУБД в институте, один из моих преподавателей проходил стажировку в США. Он рассказывал студентам о базе данных ORACLE, о применении на производстве, в финансовых организациях, в крупных государственных учреждениях.

Во время изучения я был очарован четкой и понятной организацией структур информации в СУБД ORACLE, обширностью средств работы с данными, ее мощностью и вместе с тем неповторимой гибкостью, а также уникальными возможностями данной СУБД.

Далее, после института и защиты диплома, в течение нескольких лет я работал на крупном производственном предприятии, информационная система которого была построена преимущественно на использовании СУБД ORACLE.

Потоки данных представлялись грандиозными для того времени, каждый день сервер ORACLE рассчитывал сотни тысяч производственных компонентов, технологических маршрутов, спецификаций изделий.

Стоит понимать, что на дворе был конец XX века и объемы, которые тогда казались нам огромными, сейчас вызывают улыбку, технологические мощности ушли далеко вперед, и сейчас даже средний ноутбук сопоставим по быстродействию с многопроцессорными серверами тех лет.

Поле этого я перешел в сектор телекоммуникаций, и именно здесь в своей работе я столкнулся с быстрыми транзакциями, а также обработкой огромных объемов данных в сотни и сотни миллионов записей.

Далее была длительная работа в процессинге крупного банка, где также использовалась СУБД ORACLE, интернет-сатрапе средней компании-разработчика ПО, на этом этапе я познакомился с СУБД других производителей, таких как MS SQL PostgreSQL, MySQL; также я работал на большом DWH-проекте, где объемы в миллиарды записей передавались за считаные минуты.

И каждый раз, сталкиваясь с новой для себя сферой, я открывал дополнительные возможности и новые уникальные приемы разработки, которыми и хотел бы поделиться в этой книге.

Именно после многих лет работы начинаешь осознавать, что практика и теория отличаются. Те примеры, которые описаны в документации, могут работать по-разному в разных условиях, а также иметь множество нюансов использования в конкретной ситуации в конкретной сфере.

Основой моей работы была именно СУБД ORACLE, хотя я также неизбежно сталкивался вплотную и с другими технологиями, такими как Java, SAS, Python, веб-разработка JavaScript, Node JS.

Важно осознавать, что ORACLE сейчас представляет собой целый конгломерат производственных решений, куда входит, например, ORACLE Siebel CRM, JaVA, ORACLE Service Bus, но все же основным продуктом данной корпорации была и остается именно СУБД ORACLE.

Что же касается настоящего момента, то сейчас я провожу специальные курсы по SQL в базовой и расширенной версиях, на которые каждый из вас может записаться.

Адрес курсов www.sqladv.ru

Вопросы учеников, часть практических примеров взяты непосредственно с этих курсов.

День первый

Шаг 1. Что такое SQL, назначение языка

Приветствую вас, уважаемый читатель. Позволю написать несколько слов о себе.

На текущий момент вот уже более 20 лет я работаю IT-специалистом. Я занимал должности архитектора, администратора баз данных, разработчика баз данных.

У меня также есть своя IT-школа sqladv.ru, где один из курсов, который я веду сам, посвящен разработке баз данных и языку SQL.

Эта книга представляет собой также своеобразный курс обучения: с помощью данной книги, упорно занимаясь, вы освоите язык SQL от начального уровня до уровня ведущего разработчика.

В данной книге рассмотрен диалект языка ORACLE SQL как один из наиболее распространенных на сегодняшний день. Также специалисты ORACLE обычно имеют более высокую зарплату по сравнению с другими разработчиками.

Когда я преподавал в своей IT-школе sqladv.ru, то убедился, что программирование – это, прежде всего, практика, поэтому в каждой главе данной книги разбираются актуальные примеры и всегда присутствует несколько практических заданий, обязательных для выполнения.

В каждой главе книги вы сможете найти наиболее интересные и актуальные вопросы моих учеников, разумеется, с моими ответами.

Итак, перейдем непосредственно к теоретической части.

Что такое базы данных, назначение баз данных

Трудно себе представить, что раньше вся информация размещалась на бумажных носителях и архивы документов занимали подчас целые здания.

Сейчас же пришел новый век, новая информационная эпоха, и теперь для хранения и обработки информации используются в основном электронные системы.

Огромное количество таких систем работает под управлением баз данных. Спектр применения систем управления базами данных на сегодняшний день практически необъятен – базы данных используются в интернете, в производстве, в промышленности, в маркетинге, в мобильных устройствах, в финансовой и банковской сферах, на телевидении, в телекоммуникациях и рекламе.

Какова применимость баз данных, то есть где используются базы данных?
Вот лишь некоторые области, где базы данных нашли применение.

Веб-технологии

Веб-технологии проникли в нашу жизнь, и без них уже сложно представить современный мир. Социальные сети, почта, поисковые системы, онлайн-сервисы погоды и навигации – всем этим большинство из нас пользуется ежедневно.

Почти все онлайн-ресурсы работают с базами данных. Практически каждый сайт, поисковая система, социальная сеть построены на основе баз данных и используют язык SQL.

Поэтому, если ваш бизнес или ваша профессия каким-либо образом связаны с интернет-проектами, если вы веб-дизайнер или веб-программист (и не важно, на каком языке вы специализируетесь), знание баз данных и языка SQL вам обязательно пригодится в работе.

Мобильные устройства

Большинство мобильных приложений, приложений для планшетов также использует базы данных в своей работе. Система управления базами данных для мобильных устройств называется SQLite. SQLite имеет ряд особенностей, связанных с характеристиками мобильных устройств, но в целом использует такой же синтаксис SQL, как и другие базы данных.

Игры

Современные компьютерные игры также невозможны без использования баз данных. Игры – это множество объектов, игроков, карт, вооружений, стратегий, юнитов… С этой информацией надо активно работать. Без систем управления базами данных тут не обойтись. И если вы планируете связать свою жизнь с интереснейшей профессией игрового дизайнера или программиста компьютерных игр, знание баз данных будет для вас очень и очень желательным.

Крупные корпорации

Во всех без исключения крупных компаниях используются базы данных. Системы управления базами данных управляют банкоматами и пунктами выдачи наличных, на них реализованы бухгалтерия, учет и управление кадрами. Системы управления базами данных считают телефонные звонки по тарифам, вычисляют стоимость интернет-подключений и трафика.

Если вы работаете в крупной компании или собираетесь связать свою карьеру с работой на корпорацию, вам необходимо понимать, что такое база данных, принципы ее устройства, знать и понимать язык для работы с базами – SQL.

Назначение языка SQL, необходимость изучения этого языка

Structured Query Language (SQL) – язык структурированных запросов.

Язык запросов SQL – универсальный язык для работы с данными базы. Язык запросов SQL используется для управления массивами данных в БД, множествами.

Язык SQL предоставляет возможность для вывода структурированной заданной информации из базы. SQL также применяется для изменения данных, добавления данных из базы.

Язык SQL относится к функциональным языкам программирования. Он отличается от алгоритмических языков. Основу языка составляет не алгоритм как таковой, а совокупность команд, определяющих взаимоотношения информационных множеств и подмножеств.

Следует отметить, что системы управления базами данных – СУБД – имеют различные реализации, такие как ORACLE, MS SQL, MY SQL.

Язык SQL в разных СУБД имеет небольшие отличия, например в детальном синтаксисе описания операторов.

Такие отличия присутствуют в специальных функциях, относящихся к той или иной СУБД, но все же в основном язык – это общий синтаксис, практически идентичный для любой СУБД.

В данном курсе мы будем рассматривать общепринятый синтаксис SQL ORACLE.

Данная книга, как я ранее писал, обучает диалекту ORACLE SQL как наиболее востребованному и сложному, но на страницах книги вы также сможете найти некоторые примеры на других диалектах.

Вопросы учеников

Я программирую на PHP, пригодятся ли мне знания из данной книги?

Да, язык PHP используется для доступа к данным команды SQL, поэтому если вы намерены повышать свой профессиональный уровень, вам необходимо изучить материалы этой книги.

Какой уровень знаний у меня будет после прочтения данной книги и выполнения всех практических заданий?

Вы будете знать SQL на профессиональном уровне, вполне достаточном для разработки сложных баз данных.

В данный момент получили широкое распространение NoSQL базы данных, какой язык используется для работы с такими СУБД?

Для каждой NoSQL СУБД используется свой язык программирования, отличный от SQL, разумеется.

С каким уровнем знаний можно приступить к чтению этой книги?

С начальным уровнем знаний. Вполне достаточно небольшого уровня компьютерной грамотности.

Шаг 2. Теория и практика. Учебная схема данных. Организация работы

Научиться языку SQL на профессиональном уровне по данной книге вполне реально, более того – я не вижу причин, по которым это может не получиться. Разумеется, многое зависит от вашего личного упорства, методичности обучения, системности выполнения практических заданий. Выделите каждый день по три—четыре часа вашего времени, и уже через три недели вы сможете писать SQL-запросы любой сложности.

Книга называется «100 шагов», и это соответствует действительности: это 100 шагов, которые вам необходимо пройти, для того чтобы овладеть SQL и базами данных на профессиональном уровне. Каждый шаг представляет собой отдельную главу книги. Глава книги – обзор определенного вопроса или темы по предмету: базы данных или язык SQL.

Каждая глава поделена, в свою очередь, на следующие разделы:

1. **Введение** – в этом разделе рассказывается, собственно, о предмете или теме, которой посвящается данная глава книги.
2. **Теория и практика** – читателю даются теоретические обоснования темы, разбирается синтаксис рассматриваемых в главе операторов. Приводятся понятные и доступные примеры.
3. **Важные замечания** – в любой теме есть свои особенности, свои нюансы, эти нюансы и обозреваются в разделе.
4. **Вопросы учеников** – здесь я отвечаю на наиболее популярные и интересные вопросы, которые задавали мне слушатели моих курсов.
5. **Контрольные вопросы и задания для самостоятельного выполнения** – вопросы по уроку, вопросы и задания, которые вам необходимо решить самим.

Общая схема процесса обучения, или Как читать данную книгу

Книгу следует рассматривать как учебное пособие, и я не стану скрывать, что в книге используются материалы моих курсов по обучению базам данных и языку SQL школы sqladv.ru.

Внимательно изучите теорию, запомните, проанализируйте синтаксис команд.

Все примеры необходимо прорешивать, самостоятельно писать запросы, создавать схемы, размышлять над теоретическим обоснованием примера, то есть стараться понять, почему получается так, а не иначе.

Практические задания также необходимо решать самостоятельно, и при решении данных заданий следует обратиться к материалам из теории.

Особое внимание следует уделить вопросам, рассматриваемым в разделах «Важные замечания» и «Вопросы учеников». Именно в этих разделах, как правило, содержится наиболее важная и полезная информация.

Несколько терминов

Иногда в тексте книги могут встретиться некоторые сокращения и специальные термины; расшифруем их значение.

СУБД – система управления базами данных – совокупность программных средств, обеспечивающих управление созданием и использованием баз данных. Наиболее распространенные: ORACLE, MS SQL, mySQL, PostgreeSQL.

БД – база данных – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

Учебная схема

Для занятий нам понадобится учебная схема данных.

И есть хорошая новость.

Многие примеры начального курса могут быть выполнены онлайн, для этого вам потребуется просто перейти по заданной ссылке и заполнить некоторые поля.

Итак, большинство примеров и практических заданий вы сможете выполнить на онлайн-сервисах.

Первый сервис предоставляет компания ORACLE для разработки, я уже зарегистрировал там аккаунт, и сейчас там уже есть все таблицы, которые нам потребуются в учебе.

Также там есть все необходимые заполненные данные для выполнения учебных задач и запросов.

Вам достаточно пройти по ссылке

https://apex.oracle.com/pls/apex/

и ЗАПОЛНИТЬ регистрационную информацию.

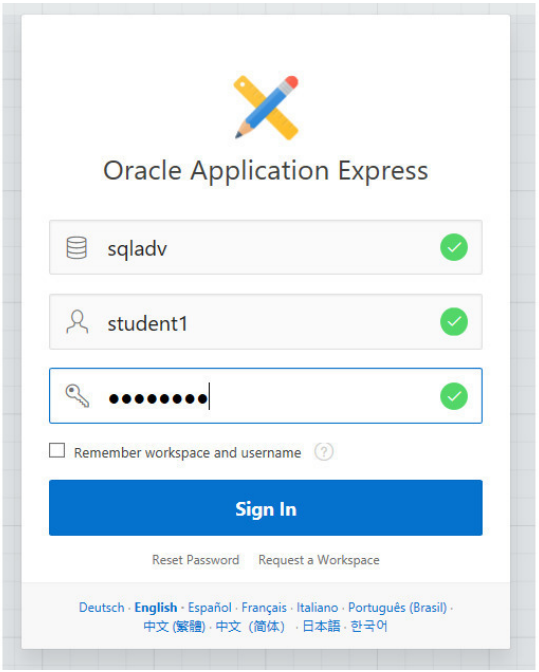


Рисунок 1. Форма авторизации в APPEX

Первое поле сверху мы заполняем SQLADV, во второе поле мы вносим имя пользователя student1 и заполняем пароль – также student1.

Также будут работать учетные записи: student2/, student2, student3/, student3… student11/, student11).

Перед вами откроется среда разработки.

Выберите пункт меню SQL Workshop, а дальше SQL ComMANd.

Пред вами откроется среда выполнения SQL-запросов.

Напишите следующий учебный запрос:

SELECT * FROM AUTO;

Нажмите кнопку RUN SQL, и далее в нижней части экрана должен появиться результат выполнения запроса.

Это и будет тот сервис, которым вы в основном сможете пользоваться для выполнения практических заданий.

Данный ресурс предоставлен компанией ORACLE в рекламных маркетинговых целях.

Если не устраивает данный сервис или почему-то этот сервис у вас не работает, тогда существует второй способ.

Вы можете воспользоваться сервисом SQL-Фидель.

Учебная схема, по которой необходимо заниматься, состоит из нескольких таблиц, которые заполнены соответствующими данными.

Введите в поле браузера ссылку
http://sqlfIDdle.com/.
Выберите тип базы данных ORACLE 11 g r2.

Далее вам потребуется загрузить учебную схему, по которой мы будем заниматься. Скрипт учебной схемы находится по следующей ссылке в текстовом документе:
http://sqladv.ru/dev/sql.txt.

Скопируйте содержимое скрипта в поле в левой части экрана и нажмите кнопку BuildSchema.

После чего уже в правой части экрана SQLFIDlle вы сможете писать необходимые запросы и выполнять учебные задания.

Учебные запросы в SQLFIDdle пишутся в текстовом поле в правой части экрана.

После создания схемы напишите следующий учебный запрос:

SELECT * FROM MAN

Нажмите RUN SQL, в нижней части экрана должен появиться результат выполнения запроса.

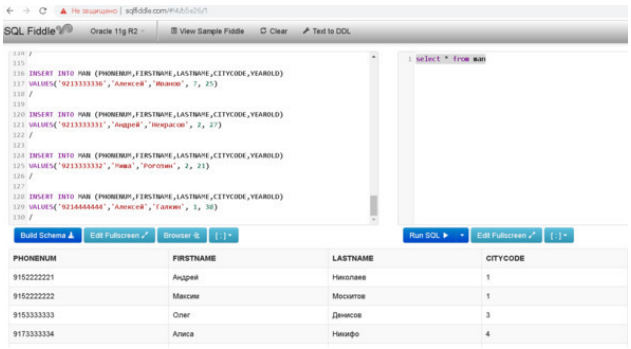


Рисунок 2. Пробный запрос SQLFIDlle

Если все получилось, то вы можете приступать к учебе.

Вопросы учеников

Так все-таки в каком из сервисов лучше выполнять практические задания?

В любом, лично мне больше нравится сервис APEX.

Я пытаюсь залогиниться в сервис APEX, но появляется сообщение об ошибке. В чем может быть дело?

Проверьте правильность ввода имени пользователя и пароля. Самое верхнее поле должно содержать значение SQLADV.

Во время работы с сервисом SQLFIDlle при создании схемы возникает ошибка.

Проверьте, пожалуйста, правильно ли установлен переключатель СУБД – значение выпадающего списка должно быть ORACLE 11g. Также проверьте, скопирован ли скрипт при создании схемы.

Шаг 3. Подготовка к работе. Процесс обучения. Описание интерфейса ORACLE APEX

Подготовка к работе

До шага 51 все занятия и практические упражнения можно выполнить с использованием онлайн-сервисов, в шаге 51 подробно описано, какое программное обеспечение следует установить дополнительно и как это сделать.

Процесс обучения

Как я уже говорил, самое важное в процессе обучения языку SQL – это именно практика, при этом не важно, как много вы знаете. Главное – научиться использовать свои знания в работе.

Тысяча практических задач из данной книги, прорешенных вами, даст гораздо больший эффект, чем изучение одной лишь теории.

Данная книга ориентирована исключительно на практическую работу. Теория здесь поясняется практическими примерами.

Изучив теорию и синтаксис выражения, сразу приступайте к практике, чтобы понять, как работает данный синтаксис, данное выражение.

Практические задания находятся сразу после пояснения теоретических положений. Примерный текст практического задания выглядит следующим образом:

> Выберите из таблицы автомобилей (AUTO) машины синего и зеленого цветов.

Попробуйте сначала самостоятельно написать запрос.

Сравните свое решение с решением, приведенным в книге.

Если сразу не получается, тогда внимательно изучите решение, приведенное в книге, и постарайтесь понять сам принцип, как решается данная задача, что является главным в решении.

После решения каждого из заданий переходите к заданиям для самостоятельного выполнения.

В данной книге рассматривается более тысячи практических примеров и упражнений по SQL.

Кроме того, в книге есть не только практические задачи – после примеров в некоторых шагах встречаются контрольные вопросы по теоретическим материалам.

Такие вопросы необходимы для более четкого понимания изучаемой темы.

Внимательно изучите вопрос и постарайтесь дать на каждый вопрос развернутый и подробный ответ.

Некоторые названия таблиц заменены их смысловыми обозначениями. Вам следует самостоятельно понять, исходя из смысла слова, в какой таблице находятся соответствующие данные.

На самом деле основных таблиц всего три, так что это будет несложно. Например, если речь идет об авто, тогда это таблица AUTO, если речь идет о людях, покупателях, тогда имеется в виду таблица MAN, если о городах – CITY.

Вот несколько практических советов, как улучшить процесс обучения SQL.

Составьте карточки

Наиболее сложные, трудно запоминаемые теоретические вопросы и ответы лучше записывать на карточки, чтобы в конце занятий повторять сложные темы.

Также рекомендуется повторять эти карточки через каждые 10 глав книги.

Данная карточка может выглядеть следующим образом. С одной стороны пишется ключевой вопрос: «*Какой оператор в SQL-запросах отвечает за группировку данных?*»

С другой стороны ответ: «*Оператор GROUP BY*».

И далее несколько примеров запросов с использованием данного оператора.

Составляйте свой список вопросов

Для себя составьте дополнительные вопросы и ответьте на них.

Например, мы изучили агрегатные функции SUM, MAX, MIN, а как работает агрегатная функция COUNT? Не бойтесь задавать себе сложные вопросы и изучать новую информацию.

Создайте свое задание

Придумайте свои практические задания и порешайте их.

В дополнение к практическим заданиям из книги продумайте свои собственные и попробуйте их решить.

Например, у вас есть практическое задание: выбрать из таблицы AUTO все автомобили марки BMW синего цвета.

Выбрать из таблицы AUTO все автомобили марки BMW синего и зеленого цветов.

Интерфейс ORACLE APEX

Онлайн-сервис ORACLE APEX обладает рядом дополнительных возможностей, которые будут нам помогать в процессе обучения. Поэтому, если вы используете для выполнения практических задач APEX, рекомендую ознакомиться с данным материалом.

К дополнительным возможностям сервиса APEX относится просмотр объектов схемы данных.

После входа в сервис APEX следует воспользоваться пунктом меню SQL Workshop, выберите подпункт меню OBJECT BROWSER.

В левой части экрана располагается список объектов, где через выпадающий список указывается, объекты какого типа отражаются в списке.

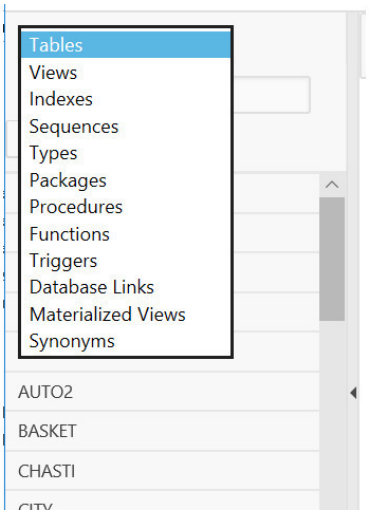


Рисунок 4. Список объектов

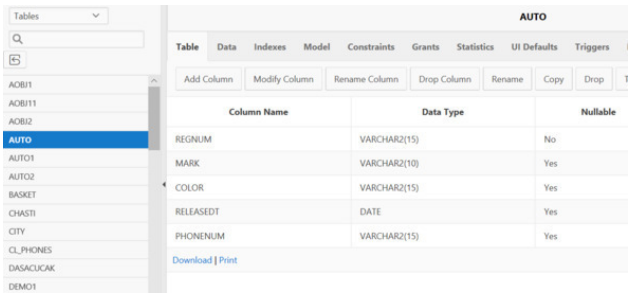
Выбор соответствующего типа объекта покажет список объектов заданного типа. Щелчок по заданному объекту позволяет отобразить структуру и свойства заданного объекта.

Просмотр структуры таблиц

Выберите в выпадающем списке в левой верхней части формы TABLEs. Список отразит все таблицы, которые вам доступны.

Выберите любую из таблиц.

В правой части страницы отразится структура выбранной таблицы.



Также доступны следующие вкладки, которые показывают сведения о таблице:

- TABLE – структура избранной таблицы;
- Data – данные избранной таблицы;
- INdexes – сведения об индексах заданной таблицы;
- ConstraINts – ограничения заданной таблицы;
- Grants – права базы данных по заданной таблице;
- SQL – SQL-код таблицы. Если вам необходимо посмотреть SQL-код таблицы, тогда следует обратиться к этой вкладке.

Просмотр кода процедур

В некоторых шагах мы обращаемся к исходному коду процедур и функций.

Выберите в выпадающем списке одно из следующих наименований: SEQUENCEs, Function, Procedures, Packages.

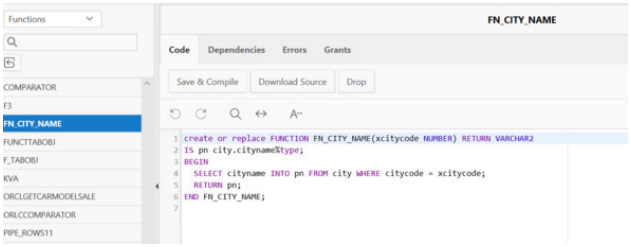


Рисунок 5. Исходный код функции Fn_CITY_NAME

В правой части веб-страницы будет отображен исходный код выбранного объекта.

Вопросы учеников

Сколько времени следует уделять занятиям?

Рекомендую уделять занятиям не менее четырех часов в день, тогда за 20 дней вы успеете пройти все шаги.

Если я все же не могу найти ответ на вопрос или не могу решить задание, что мне делать?

На сайте sqladv.ru есть ссылка на нашу группу в «Фейсбуке», там вы наверняка найдете ответ и решение задачи, с которой испытываете трудности.

В SQLFIDele есть такие же возможности по просмотру и редактированию таблиц, как в ORACLE APEX?

Нет, *SQLFIDele* – это менее сложный инструмент, тем не менее его возможностей достаточно, чтобы выполнить большинство практических заданий из этой книги.

Сколько примерно времени в пропорции уделять теории, а сколько посвятить практике?

Лучше всего из четырех часов рабочего времени следует один час уделить теории, а три часа – практике. Таким образом, примерно 80 процентов вашего учебного времени должно занимать выполнение практических задач.

Контрольные вопросы и задания для самостоятельного выполнения

1. Сколько рекомендуется тратить времени на занятия?
2. Как сделать карточки с наиболее сложными для понимания вопросами?
3. Как в APEX посмотреть все таблицы схемы?
4. Как в APEX посмотреть структуру заданной таблицы?

Шаг 4. Таблицы в базе данных

Теория

База данных – это, прежде всего, таблицы. Таблицы базы данных можно представить как таблицы в WORD или EXCEL, где в каждой ячейке содержатся определенные данные, но также есть и некоторые отличия.

Дело в том, что таблицы в базах данных создаются по некоторым правилам, и вот основные правила для таблиц в базе данных.

Так как мы изучаем SQL-диалект ORACLE СУБД, то данные правила справедливы именно для СУБД ORACLE:

• любая таблица в базе имеет уникальное наименование в рамках схемы данных;

• у каждой таблицы всегда есть заданное количество колонок: больше нуля и меньше 1024;

• каждая колонка также должна иметь уникальное наименование, но уже в рамках данной таблицы;

• в таблице в базе данных может быть практически неограниченное количество строк, здесь ограничения касаются только объема диска базы данных;

• для данных в таблице можно создавать ограничения. Ограничения касаются всех данных в колонке, на которую установлено ограничение;

• имена таблиц, имена колонок имеют ограничения по количеству символов и не могут называться зарезервированным словом, например командой из языка SQL или PL SQL. Также наименование колонки таблицы не должно начинаться с цифр;

• имя колонки в рамках таблицы также должно быть уникальным.

Создание таблиц по указанным правилам – это первый шаг в разработке базы данных.

Таблицы в базе, состав колонок таблицы должны производиться в соответствии со стандартами проектирования реляционной базы данных.

Работа со структурой таблиц, данными в таблицах осуществляется с помощью языка запросов SQL.

Одна или несколько колонок в таблице могут быть обозначены как первичный ключ.

Первичным ключом обозначаются колонки таблицы, содержащие набор уникальных значений, по которым мы можем однозначно идентифицировать строчку в рамках этой таблицы. Первичный ключ не может содержать пустые значения, так как всегда имеет ограничение NOT NULL.

Вторичный ключ – так обозначается колонка таблицы, в которой есть данные, используемые для связи с другой таблицей.

Таблицы нашей учебной схемы

Наша учебная схема очень проста и состоит всего лишь из четырех таблиц.

Первая таблица MAN содержит сведения о людях, которые приобрели машины.

Колонки таблицы MAN:

• PHONEnum – уникальный телефонный номер человека, первичный ключ для таблицы MAN, содержит текстовые данные;

• CITYCode – код города, вторичный ключ для связи с таблицей CITY;

• FirstName – имя человека (текстовые данные);

• LAStName – фамилия человека (текстовые данные);

• YearOld – возраст человека (числовые данные).

Таблица CITY – справочник городов, состоит из трех колонок:

• CITYCODE – уникальный код города, ключевое поле для таблицы CITY (числовые данные);

• CITYNAME – наименование города (текстовые данные);

• PEOPLES – население города, количество человек, которые проживают в городе (числовые данные).

Таблица AUTO – сведения об автомобилях автосалона.

Колонки таблицы AUTO:

• REGnum – уникальный регистрационный номер автомобиля (содержит текстовые данные);

• PHONEnum – телефонный номер покупателя, вторичный ключ для связи с таблицей MAN;

• MARK – марка авто (текстовые данные);

• COLOR – цвет авто (текстовые данные);

• ReleASeDT – дата создания авто, дата/время (специальный тип данных).

Таблица AUTO1 является копией таблицы AUTO и имеет те же колонки, что и таблица AUTO, и достаточно похожие данные, эта таблица используется в нескольких учебных заданиях (так же, как CITY1, MAN1).

Следующее изображение показывает основные таблицы в учебной базе данных в виде схемы:

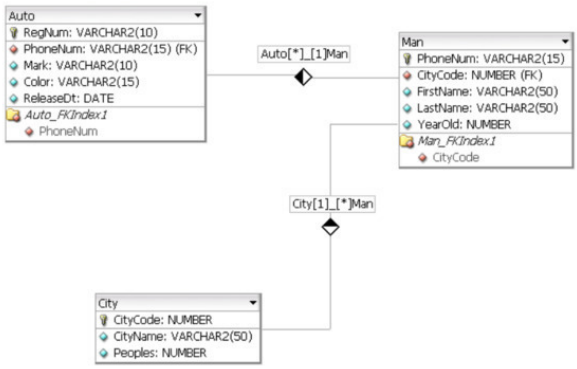


Рисунок 6. Учебная схема базы данных

Вопросы учеников

Вы сказали, что наименования таблиц должны быть уникальны в рамках одной схемы. Что такое схема?

В СУБД есть понятие схемы – это особая логическая область, ассоциированная с заданной учетной записью, которая объединяет несколько объектов базы данных.

Почему телефонный номер покупателя PHONEnum – текстовое поле, разве оно не должно быть числовым?

Иногда телефонный номер заполняют со скобками. Чтобы разрешить это противоречие, я использовал текстовый (VARCHAR2) тип данных для этой колонки; кроме того, так сделать правильно, так как это упрощает поиск нужных нам номеров по префиксу.

Какие команды SQL позволяют изменять структуру таблицы, добавлять новые колонки, например?

Это команда ALTER TABLE, с которой мы познакомимся чуть позже.

Контрольные вопросы и задания для самостоятельного выполнения

1. Могут ли колонки разных таблиц называться одинаково?
2. Что такое первичный ключ, какие колонки (первичные ключи) есть в нашей учебной схеме?
3. Какие данные находятся в учебной таблице CITY?
4. Какая колонка в таблице MAN нашей учебной схемы содержит данные о возрасте человека?

Шаг 5. Типы данных

Для удобства в SQL все данные разделены на различные типы: например, есть строковый тип, к которому относятся только строки и текст; есть целочисленный тип, к нему можно отнести только целые числа; определен специальный тип данных для чисел с плавающей точкой.

Каждой колонке в таблице назначается свой определенный тип данных, то есть каждая колонка таблицы может сохранять данные строго заданного типа и никаких данных другого типа, отличного от этого. Например, в одной из колонок могут находиться только строки и текст, а в другой – только числа.

Типов данных в SQL ORACLE-диалекте множество, мы же рассмотрим самые основные из них.

Теория и практика

Ниже приведена таблица основных типов данных, используемая в SQL ORACLE. В таблице колонка-размер означает, какой максимальный объем информации сможет вместить этот тип данных. Например, тип данных VARCHAR2 может вместить в строку длиной не более 4000 символов.

Типы данных	Размер	Описание
char(размер)	Максимальный размер 2000 байт.	Для текстовых данных не превышающих 2000 символов. Статический выделение памяти под текст. Если сохраняемое значение короче, строка дополняется пробелами до 2000 символов. В скобках указывается количество символов фиксированной длины
nvarchar2(размер)	Максимальный размер 4000 байт.	Специальный тип для сохранения текстовых данных в формате UNICODE. В скобках указывается – количество сохраняемых символов в кодировке Unicode переменной длины.
varchar2(размер)	Максимальный размер 4000 байт.	Для строк и текста не превышающих 4000 символов. В скобках указывается количество сохраняемых символов переменной длины.
CLOB	Максимальный размер 2GB.	Для сохранения больших тестов. Символьные данные переменной длины.
raw	Максимальный размер 2000 байт.	Для двоичных данные переменной длины
number(точность,масштаб)	Точность может быть в диапазоне от 1 до 38. Масштаб может быть в диапазоне от -84 до 127.	Например, number (14,5) представляет собой число, которое имеет 9 знаков до запятой и 5 знаков после запятой.
numeric(точность,масштаб)	Точность может быть в диапазоне от 1 до 38.	Например, numeric(14,5) представляет собой число, которое имеет 9 знаков до запятой и 5 знаков после запятой.
decimal(точность,масштаб)	Точность может быть в диапазоне от 1	Например, decimal (5,2) — это число, которое имеет 3 знака перед запятой

таблица. Типы данных

	до 38.	и 2 знака после .
date	date может принимать значения от 1 января 4712 года до н.э. до 31 декабря 9999 года нашей эры.	Используется для хранения информации дата время.

Таблица. Типы данных

Важные замечания

Для удобства (во избежание излишней путаницы) в учебных примерах для этой книги рассматриваются в основном только три типа данных, однако нам достаточно этих типов для решения большинства учебных задач и понимания учебного материала.

Основные типы данных, используемые в книге в практических задачах:

• VARCHAR2 (n) – тип для хранения текстовой информации, в скобках указывается максимальное количество символов в строке. Данный тип используется при работе со строковыми данными разной длины, память под такие данные выделяется динамически;

• NUMBER – тип данных для хранения числовой информации, причем можно использовать как для целых чисел, так и для чисел с плавающей точкой;

• DATE – специальный тип данных для сохранения специальной информации – дата-время, например дата и время создания записи, дата и время электронной подписи документа, дата и время заключения сделки.

Эти типы данных достаточно часто используются на практике и применяются в работе.

Вопросы учеников

Вы рассказали про тип данных VARCHAR2 для хранения строк, но в виде строки можно записать и числа, и даты тоже. Зачем так много разных типов, может, они не нужны?

Да, вы можете сохранять числа как строки, но в дальнейшем вам будет сложно работать с этими данными, обработка будет затруднена. Например, со строками нельзя производить математических вычислений, и вам придется применять операции преобразования, что негативно скажется на качестве и скорости работы вашей программы.

Типы данных в других SQL СУБД, отличных от ORACLE, также различаются?

Есть незначительные различия, например в MS SQL используется VARCHAR, а не VARCHAR2, или вместо CLOB используется тип TEXT. Необходимо обратиться к соответствующему разделу документации выбранной СУБД, чтобы понять, какие именно типы данных различаются.

Почему в ORACLE SQL используется именно VARCHAR2, а не просто VARCHAR, как в MS SQL-сервер?

В ORACLE SQL тоже существует тип VARCHAR, но исторически сложилось так, что в ORACLE SQL между этими двумя типами существует разница:

• VARCHAR может хранить до 2000 символов, а VARCHAR2 может хранить до 4000 символов;

• если мы объявим тип данных как VARCHAR, то будет зарезервировано место для пустых NULL VALUES.

Поэтому чаще всего на практике в ORACLE используется VARCHAR2.

Контрольные вопросы и задания для самостоятельного выполнения

1. Для чего используется тип DATE?
2. Нам необходимо в одной из колонок таблиц сохранять целые числа, какой тип следует использовать?
3. Какой тип правильно использовать для хранения данных о цвете автомобиля?
4. Какой тип данных необходимо использовать для хранения сведений о росте человека?

День второй

Шаг 6. Создание таблиц

Как мы уже поняли, основная информация, с которой нам предстоит работать в базе данных, находится в таблицах. Из шага 4 (таблицы в базе данных) про таблицы мы узнали следующие важные сведения:

- каждая таблица обладает уникальным наименованием;
- у таблицы обязательно должны быть колонки, каждая из которых обладает уникальным наименованием в рамках этой таблицы;
- для каждой колонки таблицы задается свой тип данных (про типы данных подробно рассказывается в предыдущем шаге).

Важные замечания

1. Имена таблиц, имена колонок в SQL имеют ограничение по количеству символов и не могут называться зарезервированным словом, таким как команда из языка SQL.

Например, нельзя назвать таблицу или колонку GROUP, это зарезервированное слово, часть команды GROUP BY, или же недопустимо называть таблицу или колонку таблицы FROM, SELECT, INSERT, KEY.

2. Имена таблиц и имена колонок не могут начинаться с цифр; также принято использовать при именовании таблиц и колонок латинские буквы, хотя, впрочем, вполне возможно назвать таблицы и колонки таблиц на русском, китайском и даже хинди (я видел и такое), но подобные выкрутасы не приветствуются.

3. Очень желательно, чтобы наименования таблиц, а также наименования колонок таблиц отражали смыл относительно данных, которые содержатся в этих таблицах, например: MANS, CARS, STAFF – люди, машины, персонал, GOODS – товары, ITEMS – элементы.

Теория и практика

Теперь, когда мы разобрались с теоретической частью, самое время заняться практическими упражнениями.

Разберемся на практике, как создавать таблицы в базе, используя язык SQL.

Для создания таблиц используется специальная команда SQL CREATE TABLE.

Синтаксис упрощенный.

```
CREATE TABLE имя таблицы (
Column_NAME1 column_type (NUMBER, или VARCHAR2 (n) или DATE) primary KEY,
Column_NAME 2 column_type (NUMBER, или VARCHAR2 (n) или DATE),
Column_NAMEn column_type (NUMBER или VARCHAR2 (n) или DATE)
);
```

Для простоты на начальном этапе в наших практических примерах мы будем использовать три основных типа данных.

(NUMBER, VARCHAR2 (n), DATE), соответственно, для хранения чисел, строковых данных и данных календарного типа (дата/время).

Для текстового типа VARCHAR2 (n) после VARCHAR2 в скобках указывается количество символов для данной колонки.

Итак, сначала идет команда создания таблицы CREATE TABLE, далее – наименование таблицы: MANS, GOODS, ITEMS или любое другое.

Далее в скобках через запятую перечисляются наименования колонок и тип колонок.

Вот несколько примеров, как создавать таблицы в языке SQL:

1. Создать таблицу «Мебель»:

- артикул;
- наименование;
- количество;
- номер партии.

```
CREATE TABLE furnit (artikl VARCHAR2 (50) PRIMARY KEY, NAME VARCHAR2 (50),partCOUNT NUMBER, partnum NUMBER);
```

2. Создать таблицу «Корзина для веб-магазина»:

- артикул;
- наименование товара;
- имя покупателя;
- количество;
- дата покупки.

```
CREATE TABLE shopINgcart (
article VARCHAR2 (50) PRIMARY KEY
,itemNAME VARCHAR2 (50)
,buyerNAME VARCHAR2 (50)
,itemCOUNT NUMBER
,dtbuy DATE
);
```

Создать таблицу «Предприятие»:
• название бригады;
• номер бригады;
• количество человек;
• дата создания;
• направление деятельности.

```
CREATE TABLE plant
(
NAMEteam VARCHAR2 (15),
numteam NUMBER PRIMARY KEY,
MANCOUNT NUMBER,
crDATE DATE,
dirToDo VARCHAR2 (30)
);
```

Вопросы учеников

Можно ли использовать заглавные буквы в языке SQL и когда это допустимо?

Язык SQL не зависит от регистра, то есть при составлении команд можно писать и заглавными, и строчными буквами.

Примеры

Create TABLE Tab1 (TABno INteger PRIMARY KEY, NAME VARCHAR2 (10));
Create TABLE Tab1 (TABno INteger PRIMARY KEY, NAME VARCHAR2 (10));
CREATE TABLE Tab1 (TABNo INTEGER PRIMARY KEY, NAME VARCHAR2 (10));

Как переносить команды SQL на другую строку, если в одну строчку не помещается, существуют ли какие-то специальные правила?

Язык SQL допускает достаточно вольный перенос строк, главное, не разделять этим переносом осмысленные команды, а также соблюдать последовательность команд.

Примеры

Можно написать так:

```
CREATE TABLE TAB1 (TABno INteger PRIMARY KEY, NAME
VARCHAR2 (10));
```

А можно и так:

```
CREATE TABLE
TAB1 (
TABno INteger PRIMARY KEY,
NAME VARCHAR2 (10));
```

А вот такая запись уже неверна:

```
CREATE TABLE TAB1 (TABno INteger PRIMARY
KEY, NAME VARCHAR2
(10));
```

Еще один пример неверной записи:

```
CREATE TABLE
PRIMARY KEY
TAB1 (TABno INteger,
NAME VARCHAR2 (10));
```

Контрольные вопросы и задания для самостоятельного выполнения

1. Найдите ошибку в скрипте создания таблицы.

CREATE TABLE ORACLE1 (81NAME VARCHAR2 (20), ITEMS NUMBER);

2. Найдите ошибку в другом скрипте создания таблицы.

CREATE TABLE DELTA (SELECT VARCHAR2 (20), COUNT NUMBER);

3. Можно ли при наименовании таблицы использовать строчные и заглавные символы?

4. Создайте самостоятельно таблицу «Запчасти», задайте имена колонок и название таблицы сами, правильно определите типы данных.

Таблица «Запчасти»:
- номер запчасти;
- марка авто;
- название запчасти;
- количество данных запчастей;
- стоимость запчасти.

Создайте самостоятельно таблицу «Фото», задайте имена колонок и название таблицы сами, правильно определите типы данных.

Таблица «Фото»:
- название фото;
- размеры;
- подпись;
- дата создания.

Создайте самостоятельно таблицу «Уроки» («Занятия»), задайте имена колонок и название таблицы сами:

- название занятия;
- день недели;
- дата начала занятия;
- дата окончания занятия.

Шаг 7. Структура таблицы

Введение

Мы научились создавать таблицы на предыдущем шаге. Таблицы и колонки таблиц, их названия, расположение, последовательность колонок, типы данных колонок называются структурой таблицы.

Структуру таблицы можно менять, то есть добавлять новые колонки в таблицу, удалять колонки из таблицы, менять типы данных у заданной колонки. Также, если таблица нам больше не нужна или просто надоела, существует возможность такую таблицу удалить.

Теория и практика

Существует несколько команд для изменения структуры таблицы, добавления, удаления или изменения типа данных колонки таблицы.

Все эти команды объединяет то, что они начинаются с ключевой команды ALTER TABLE.

Добавление колонки.

Добавляем новую колонку к нашей таблице.

Синтаксис:

ALTER TABLE TABLE_NAME ADD (column_NAME column_type);

TABLE_NAME – наименование таблицы.

Column_NAME – наименование колонки.

Column_type – тип данных колонки (VARCHAR (n) или NUMBER или DATE).

Примеры:

Пусть у нас есть таблица GOODS, необходимо добавить колонку itemprice типа NUMBER, цена изделия.

ALTER TABLE GOODS ADD (itemprice NUMBER);

Пусть у нас есть таблица MANS, необходимо добавить колонку DATEreg типа DATE, дата регистрации, и колонку patronymic – отчество VARCHAR2 (50).

ALTER TABLE MANS ADD (DATEreg DATE);
ALTER TABLE MANS ADD (patronymic VARCHAR2 (50));

Удаление колонки

Также мы можем удалить колонку из заданной таблицы с помощью специальной SQL-команды DROP COLUMN.

Синтаксис:

ALTER TABLE TABLE_NAME DROP COLUMN column_NAME;

Примеры:
Пусть у нас есть таблица GOODS, необходимо удалить колонку COLOR.

ALTER TABLE GOODS DROP COLUMN COLOR;

Пусть у нас есть таблица MANS, необходимо удалить колонку YEAROLD.

ALTER TABLE MANS DROP COLUMN YEAROLD;

Меняем тип данных для колонки таблицы.

Синтаксис изменения типа колонки:

ALTER TABLE TABLE_NAME MODIFY (column_NAME DATA_type);

Column_NAME – наименование колонки.
Data_type – тип данных колонки (VARCHAR (n) или NUMBER или DATE).

Примеры:
– заменить в таблице MANS тип поля NAME на VARCHAR2 (90);

ALTER TABLE MANS MODIFY (NAME VARCHAR2 (90));

– заменить в таблице GOODS тип поля INSERT_DATE на DATE;

ALTER TABLE GOODS MODIFY (INSERT_DATE DATE);

Удаляем таблицу из базы данных.

Синтаксис команды SQL для удаления таблицы:

DROP TABLE TABLE_NAME;

Здесь TABLE_NAME – наименование таблицы.

Примеры:
– удалить таблицу DOC;

DROP TABLE doc;

– удалить таблицу ITEMS;

DROP TABLE ITEMS;

– удалить таблицу BILLING_PERIOD со связанными данными в таблице PERIODS.

DROP TABLE BILLING_PERIODS CASCADE;

Важные замечания

1. При выполнении действий по изменению структуры таблицы следует быть особенно осторожным, следует тщательно взвешивать свои действия: восстановление таблицы в прежнем виде может быть затруднительно или невозможно.

2. Если вы используете команды изменения типов данных и встречаетесь с ошибкой **ORA-01439, модифицируемый столбец при смене типа данных должен быть пуст**. Сохраните данные в столбце и используйте специальные преобразования, о которых будет рассказано в следующих шагах.

3. В некоторых случаях удаление таблицы или колонки таблицы будет запрещено, поскольку могут быть еще таблицы со связанными данными. Требуется сначала удалить данные в связанных таблицах, а уже затем удалять таблицу либо колонку. Или же воспользоваться специальной командой DROP CASCADE.

Вопросы учеников

Можно ли переименовать таблицу?

Да, вполне, и для этого есть две команды:

ALTER TABLE TABLE_NAME RENAME TO new_TABLE_NAME;

или же

RENAME <old_TABLE> TO <new_TABLE>

Универсальный же синтаксис предполагает использование ALTER TABLE.

Примеры:

Переименуем таблицу с названием STAFF в EMP:

ALTER TABLE STAFF RENAME TO emp;

Переименуем таблицу с названием TRADES в TRADE:

ALTER TABLE trades RENAME TO trade;

Можно ли переименовать столбец в таблице?

ALTER TABLE TABLE_NAME RENAME COLUMN old_column_NAME to new_column_NAME;

Пример:

Переименовать колонку с наименованием NAME в таблице STAFF в колонку LASTNAME:

ALTER TABLE STAFF RENAME COLUMN NAME TO LASTNAME;

Контрольные вопросы и задания для самостоятельного выполнения

1. С помощью какой команды можно добавить колонку к заданной таблице?
2. Повторите команды изменения структуры таблицы.
3. К таблице «Запчасти» добавить колонку «Вес запчастей» – напишите SQL-команду.
4. Удалите из таблицы FOTO колонку Fotosize – напишите SQL-команду.
5. У нас есть таблица «Предметы» («Уроки в школе»), необходимо добавить колонку «Преподаватель» – напишите SQL-команду.
6. Удалите из базы данных таблицу FOTO.

Шаг 8. Первичные ключи. Вторичные ключи

Введение

Первичный ключ – это сочетание значений колонок таблицы, уникально определяющее каждое значение таблицы. Такие колонки называются первичным ключом. Первичные ключи таблицы необходимы для поддержания целостности базы данных.

Любая колонка в таблице может быть обозначена как первичный ключ, это уникальные колонки, в которых только уникальные значения, по которым мы можем однозначно идентифицировать строчку в рамках этой таблицы.

Колонку для связи таблицы с другой таблицей называют вторичным ключом, то есть если есть две таблицы связаны по одной или нескольким колонкам, такая колонка во второй связанной таблице называется вторичным ключом.

Вторичный ключ также называют внешним ключом таблицы.

Теория и практика

В нашем примере есть две таблицы (таблица CITY, MAN по колонке CITYCODE), в таблице CITY CITYCODE является первичным ключом.

В таблице MAN CITYCODE будет вторичным ключом.

Синтаксис создания первичного ключа:

```
CREATE TABLE TABLE_NAME
(
column1 DATAtype NULL/NOT NULL,
column2 DATAtype NULL/NOT NULL,
…
CONSTRAINT constraINt_NAME PRIMARY KEY (column1, column2, … column_n)
);
```

Также возможно создание первичного ключа с помощью конструкции ALTER TABLE:

```
ALTER TABLE TABLE_NAME ADD CONSTRAINT constraINt_NAME PRIMARY KEY (column1, column2, … column_n);
```

Синтаксис создания вторичного ключа:

```
CREATE TABLE TABLE_NAME (
column1 DATAtype NULL/NOT NULL,
…
CONSTRAINT fk_column
FOREIGN KEY (column1, column2, … column_n)
REFERENCES parent_TABLE (column1, column2, … column_n));
```

Добавление вторичного ключа с помощью конструкции ALTER TABLE:

```
ALTER TABLE TABLE_NAME
ADD CONSTRAINT constraINt_NAME
FOREIGN KEY (column1, column2, … column_n)
REFERENCES parent_TABLE (column1, column2, … column_n);
```

Важные замечания

Первичный ключ может состоять из одной или нескольких колонок.
Пример:

```
ALTER TABLE TABLE_NAME ADD CONSTRAINT constraINt_NAME PRIMARY KEY (column1);
```

или же

```
ALTER TABLE TABLE_NAME ADD CONSTRAINT constraINt_NAME PRIMARY KEY (column1, columnN);
```

Вопросы учеников

Обязательно ли обозначать внешний ключ? Почему это не будет работать просто так?

Будет работать, но для поддержания ссылочной целостности необходимо использование конструкций SQL для первичных и вторичных ключей.

Контрольные вопросы и задания для самостоятельного выполнения

1. Повторите материалы данного шага.
2. Поясните, в чем отличие первичного и вторичного ключей.

Шаг 9. Ограничения

Введение

Для колонок таблицы в базе данных можно создавать ограничения.

Ограничения – это специальные синтаксические конструкции уровня колонок таблицы, которые предназначены для поддержания ссылочной целостности данных или для вставки правильных данных согласно бизнес-логике приложения.

То есть ограничения допускают вставку в ячейку таблицы только определенных данных, ограниченных заданными правилами.

Теория и практика

Все ограничения, которые используются в ORACLE SQL, можно разделить на следующие группы:

1. Ограничения на вставку пустых значений NOT NULL

Подобный вид ограничений создается, чтобы огранить вставку пустых значений в базу данных.

Снять данное ограничение можно с помощью команды изменения поля таблицы MODIFY.

Синтаксис

Добавлять ограничения на вставку пустых значений можно при создании таблицы.

```
CREATE TABLE TABLENAME (
Column1 NOT NULL, ColumnN NOT NULL
);
```

Это стандартный синтаксис создания таблицы, к имени колонки добавляется синтаксис ограничения NOT NULL.

Или изменять значения для уже готовой таблицы с помощью команды ALTER TABLE.

```
ALTER TABLE TABLENAME MODIFY ColumnName NOT NULL
```

Примеры

Создание таблицы «Корзина» с ограничением на вставку пустых значений в колонки itemNAME, itemCOUNT.

```
CREATE TABLE shopINgcart (
article VARCHAR2 (50) PRIMARY KEY
,itemNAME VARCHAR2 (50) NOT NULL
,itemCOUNT NUMBER NOT NULL
);
```

Запрет добавления пустого значения в FirstName в таблицу MAN.

```
ALTER TABLE MAN MODIFY FirstName NOT NULL
```

Запрет добавления пустого значения в LAStName.

```
ALTER TABLE MAN MODIFY LAStName NOT NULL
```

То есть значение в колонке LAStName MAN обязательно должно быть заполнено.

Снимаем ограничение на вставку пустых значений:

```
ALTER TABLE MAN MODIFY LAStName NULL
```

После выполнения команды значение в колонке LAStName MAN обязательно должно быть заполнено.

Ограничения на уникальность

Данный вид ограничений позволяет указать, что в заданные колонки необходимо добавлять только уникальные неповторяющиеся значения.

Синтаксис

ALTTER TABLE TABLENAME ADD CONSTRAINT cINs_NAME UNIQUE (columnNAME);

Пример:

ALTTER TABLE CITY ADD CONSTRAINT CITY_uniq UNIQUE (CITYNAME);

Названия городов, только уникальные значения.

Ограничения на вторичный ключ

Подобный вид ограничений мы уже рассматривали, когда изучали первичные и вторичные ключи. Смысл данного ограничения в том, что в колонке некоторой таблицы (вторичный ключ) могут находится только значения, которые есть в другой, основной таблице в колонке первичного ключа.

Синтаксис

```
ALTER TABLE for_TABLE
ADD CONSTRAINT fk_const_NAME
FOREIGN KEY (fk_column)
REFERENCES primary_talbe (pk_column);
```

Здесь for_TABLE, fk_column – таблица, колонка, куда устанавливается ограничение. Проверка значений происходит в таблице primary_talbe и колонке pk_column.

Пример

Здесь для таблицы MAN колонки CITYCODE устанавливается ссылочное ограничение по колонке CITYCODE с таблицей CITY, где CITY является главной таблицей.

```
ALTER TABLE MAN
ADD CONSTRAINT fk_MAN_CITY_CODE
FOREIGN KEY (CITYCODE)
REFERENCES CITY (CITYCODE);
```

Ограничение CHECK на вставку и изменение данных

– вычитание;

Синтаксис

```
ALTER TABLE TABLENAME ADD CONSTRAINT CHECK_NAME
CHECK (condition);
```

Здесь condition – условие, CHECK_NAME – наименование ограничения, TABLENAME – имя таблицы.

Пример

Ограничение в таблице MAN на возраст (YEAROLD) больше 16 лет:

```
ALTER TABLE MAN ADD CONSTRAINT
CHECK_YEAROLD_MAN
CHECK (YEAROLD> 16);
```

Важные замечания

Ограничение уникальности можно также создавать для нескольких колонок таблицы, это делается следующим образом:

```
ALTTER TABLE CITY ADD CONSTRAINT CITY_uniq
UNIQUE (CITYNAME, CITYCODE);
```

При этом отдельно необходимо контролировать вставку пустых значений для соответствующих полей таблицы.

Существуют дополнительные опции для создания ограничений ссылочной целостности:
- On delete cAScade – автоматическое удаление связанных строк по внешнему ключу;
- On delete NULL – значение внешнего ключа устанавливается в NULL.

При создании множества ограничений CHECK необходимо, чтобы между ними не было конфликтов, то есть чтобы правила не противоречили друг другу.

Условие в ограничении CHECK может ссылаться на любой столбец таблицы, но не может ссылаться на столбцы другой таблицы.

Вопросы учеников

Чем ограничение на уникальность UNIQUE отличается от первичного ключа (Primary KEY)?

Первичный ключ в таблице может быть только один, ограничений на уникальность может быть много. В таблице в поле с ограничением уникальности допускается вставка пустых значений. Также для первичного ключа создается специальный индекс (pk).

Если колонка в таблице содержит пустые значения, а мы добавляем к этой колонке ограничение NOT NULL, что произойдет?

Вы получите сообщение об ошибке, и ограничение не будет добавлено.

Можно ли создавать ограничения для колонок BLOB, CLOB?

Нет, некоторые виды ограничений можно создавать только для простых типов полей.

Контрольные вопросы и задания для самостоятельного выполнения

1. Добавьте ограничение для уникальных значений на колонки PHONENUM таблицы AUTO.

2. Добавьте ограничение на вставку пустых значений для колонки YEAROLD в таблице MAN.

3. Добавьте ограничение CHECK для колонки YEAROLD в таблице MAN, чтобы YEAROLD было меньше 100.

4. Добавьте ограничение ссылочной целостности для двух таблиц – AUTO и MAN.

Шаг 10. Индексы

Введение

Индексы – это специальные ссылочные массивы в базах данных. Назначение индексов – ускорение поиска данных, процессов сортировки данных. Обычно индексы увеличивают производительность запросов к базе данных.

Теория и практика

Индексы работают по принципу b-tree, то есть сбалансированной древовидной структуры.

> Для примера: у нас в одной из колонок таблицы есть уникальные значения от 1 до 500 000.
>
> Нам необходимо найти значение 255 000. По ссылочному индексу мы определяем, больше или меньше искомое значение 250 000, то есть половины всех значений, далее больше или меньше искомое значение 260 000, далее мы переходим к нашему значению 255 000.
>
> Мы не просматривали каждую запись таблицы, а нашли наше значение за несколько итераций.

Индексы создаются для определенной колонки (колонок) таблицы.

Процесс пересоздания индексов может занимать значительное время, это необходимо учитывать при операциях вставки и обновления, удаления данных.

Синтаксис

CREATE INDEX IDx_NAME ON TABLE_NAME (column_NAME);

Idx_NAME – наименование индекса;
TABLE_NAME – наименование таблицы, где создается индекс;
column_NAME – наименование колонки, для которой создается индекс.
Пример создания индекса для колонки MARK таблицы AUTO:

CREATE INDEX IDx_AUTO_MARK ON AUTO (MARK);

Реверсивный индекс

Если нам необходимо более часто читать записи, отсортированные в обратном порядке, тогда имеет смысл использовать реверсивные индексы. Например, есть таблица валют, в своих расчетах мы чаще используем данные с более поздней датой курса валют, в этом случае действительно лучше использовать реверсивный индекс для даты курса валют.

Синтаксис

CREATE INDEX IDx_NAME ON TABLE_NAME (column_NAME) REVERSE;

Пример: создание реверсивного индекса для колонки MARK таблицы AUTO.

CREATE INDEX reg_DATE ON AUTO (reg_num) REVERSE;

Удаление индекса

Для удаления индекса используется команда

DROP INDEX IDx_NAME;

Индексы создаются для определенной колонки таблицы.

Процесс пересоздания индексов может занимать значительное время, это необходимо учитывать при операциях вставки и обновления, удаления данных.

Важные замечания

Обычно использование индексов улучшает производительность базы данных, но в таблицах, где предполагается большое количество операций вставок, обновлений, много индексов использовать не рекомендуется. В этом случае производительность базы данных может существенно снизиться.

Индексы рекомендуется создавать на колонках, которые используются в операциях объединения.

Индекс автоматически создается для столбцов первичных ключей и для столбцов, на которых есть ограничение уникальности.

При наименовании индексов следует придерживаться следующего правила: IDx_имя таблицы_имена_колонок.

Вопросы учеников

Если таблица небольшая, в ней не более 200 записей например, нужен ли в такой таблице индекс?

Нет, индексы для такой таблицы, скорее всего, не понадобятся.

Какие типы индексов существуют в различных СУБД?

Существует множество разных типов индексов, но более подробно мы разберем эту тему в следующих шагах.

В моей базе данных фильтр (WHERE) данных, поиск данных всегда осуществляется одновременно по определенному набору колонок. Какие индексы следует использовать в этом случае?

В этом случае создается композитный индекс.
Синтаксис:

CREATE INDEX IDx_NAME ON TABLE_NAME (column_NAME1, column_NAMEn) REVERSE;

Контрольные вопросы и задания для самостоятельного выполнения

1. В таблице есть ограничение уникальности, имеет ли смысл создавать на этой колонке индекс?
2. В некоторой таблице постоянно обновляются записи, следует ли использовать индексы в этой таблице?
3. Создайте индекс на колонку COLOR в таблице AUTO.
4. Создайте реверсивный индекс для колонки YEAROLD в таблице MAN.

День третий

Шаг 11. Простые запросы

Введение

А сейчас отвлечемся на некоторое время от структуры таблиц и поговорим о том, как извлекать данные из базы.

Логично предположить, что если у нас есть данные, нам необходимо их выбирать из базы, обрабатывать и выводить на экран в удобном, понятном, читаемом виде.

В нашей рабочей схеме уже есть три таблицы с данными – это таблицы AUTO, CITY, MAN.

Напомню, что в таблице MAN хранятся сведения о покупателях: их имена, их возраст, CITY – это данные о городах, а таблица AUTO содержит сведения об автомобилях некоторого автосалона.

Если человек приобретает автомобиль, то в таблице AUTO в колонке PHONEnum выставляется номер телефона человека, который приобрел машину.

Для извлечения данных из базы и вывода этих данных на экран используются команды, называемые запросами к базе данных, специальная команда SQL – SELECT. Эта команда является наиболее часто используемой командой в языке SQL и постоянно применяется на практике.

Теория и практика

Начнем с самого легкого запроса, рассмотрим синтаксис самого простого оператора SELECT:

SELECT * FROM TABLE_NAME

Здесь TABLE_NAME – имя таблицы, из которой мы запрашиваем данные.
Символ * означает, что мы выводим на экран данные из всех колонок.
Откройте тестовую среду и выполните запрос

SELECT * FROM CITY;

CITYCODE	CITYNAME	PEOPLES
1	Москва	10000000
7	Котлас	110000
8	Мурманск	400000
2	Владимир	500000
3	Орел	300000
4	Курск	200000
10	Казань	2010000

Рисунок 7. Запрос из таблицы CITY

На экран выведены названия колонок в первой строке, а также данные в каждой колонке из таблицы CITY.

А теперь другой вариант синтаксиса такого же простого запроса SQL:

SELECT column_NAME1, column_NAME1, column_NAMEn FROM TABLE_NAME

В этом варианте вместо звездочки используются наименования колонок и на экран будут выведены только перечисленные колонки из заданной в поле FROM таблицы.

Пример (выполните в нашей тестовой среде):

SELECT CITYCODE, CITYNAME FROM CITY;

CITYCODE	CITYNAME
1	Москва
7	Котлас
8	Мурманск
2	Владимир
3	Орел

Рисунок 8. Запрос к таблице CITY по колонкам CITYCODE, CITYNAME

Результат запроса – на экран выведены только те две колонки, которые мы указали после оператора SELECT.

Существует и другой вариант синтаксиса для SQL-запросов:

SELECT TABLE_NAME.* FROM TABLE_NAME

или

SELECT TABLE_NAME. column_NAME1, TABLE_NAME. column_NAME1, TABLE_NAME. column_NAMEn FROM TABLE_NAME

Прорешаем задания, которые мы уже выполнили, с использованием такого синтаксиса:

SELECT CITY.* FROM CITY

При выполнении этого запроса SELECT получается результат, совершенно аналогичный тому, что и в примерах выше.

Небольшой лайфхак.

Как я составляю запросы? Сначала пишу SELECT *, затем FROM, имя таблицы, выполняю запрос, а уже после перечисляю колонки, которые необходимо вывести на экран, и далее выполняю запрос повторно.

Фильтр строк WHERE в запросе SELECT

Итак, мы научились выводить на экран все данные из заданной таблицы, но как же поступить, если нам необходимо вывести на экран только избранные строки? Допустим, что в заданной таблице миллион строк, а нам необходимо посмотреть из них лишь 10.

К счастью, язык SQL позволяет это сделать. Для этого в языке SQL и в частности в команде SELECT предусмотрен специальный оператор – **WHERE**.

Рассмотрим синтаксис команды **SELECT** с оператором **WHERE**:

SELECT * или перечень колонок FROM TABLE_NAME WHERE условие отбора строк

Примеры:

Выберем названия городов, где население 300 000 человек.

SELECT * FROM CITY WHERE PEOPLES = 300000

Альтернативная форма записи:

SELECT CITY.* FROM CITY WHERE CITY.PEOPLES = 300000

Выражение в WHERE формируется с помощью математических операндов сравнения, рассмотрим этот момент подробнее.

Операнды сравнения

> больше
<меньше
= строгое равенство
или неравенство**! =**

Примеры

Выберем все колонки (*) из таблицы городов, где население больше 300 000 человек.

SELECT * FROM CITY WHERE PEOPLES> 300000

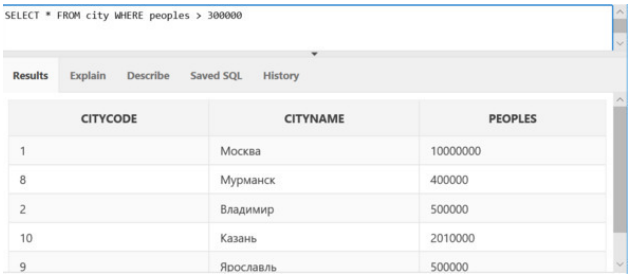


Рисунок 9. Запрос таблице CITY с условием

Выберем название города с кодом города, равным 2, из CIty

```
SELECT citycode,cityname FROM city WHERE citycode = 2
```

Альтернативная форма записи:

```
SELECT city.* FROM city WHERE city.citycode = 2
```

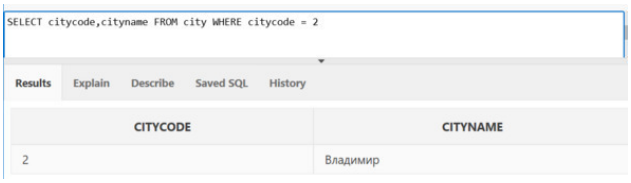


Рисунок 10. Запрос к таблице CITY по заданному CITYCODE

Выберем все имена и фамилии из таблицы MAN:

```
SELECT firstname, lastname FROM man
```

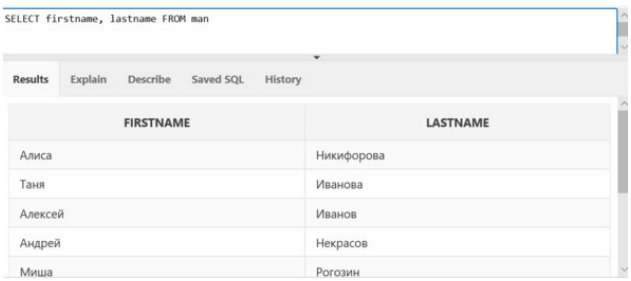


Рисунок 11. Запрос двух колонок к таблице MAN

Все колонки (*) возраст больше 27 лет из таблицы MAN:

```
SELECT * FROM man WHERE yearold > 27
```

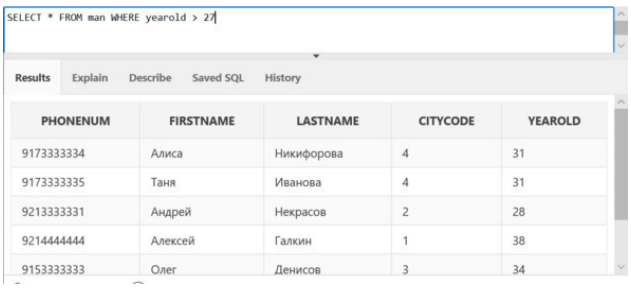


Рисунок 12. Запрос к таблице MAN, где возраст больше 27 лет

Из таблицы AUTO выберем номера автомобилей, выпущенных после 1 февраля 2005 года.

```
SELECT regnum, mark FROM auto WHERE releasedt > date'2005-01-01'
```

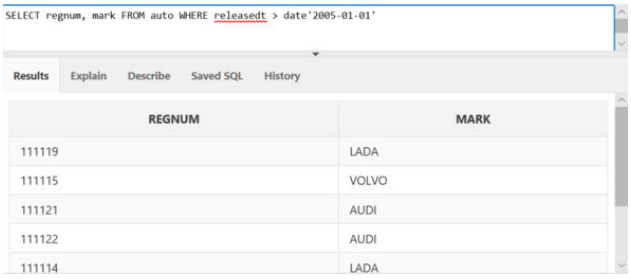


Рисунок 13. Запрос к таблице AUTO с ограничением по дате

Из таблицы AUTO выберем только зеленые автомобили.

```
SELECT * FROM auto WHERE color = 'ЗЕЛЕНЫЙ'
```

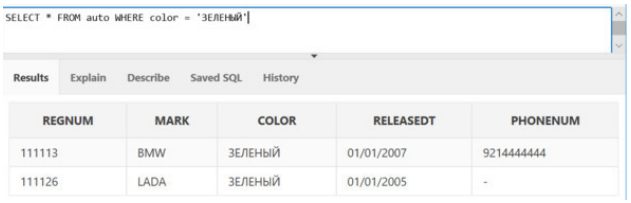


Рисунок 14. Запрос к таблице AUTO, где цвет авто зеленый

Из таблицы MAN выберем только людей с именем Миша.

```
SELECT * FROM man WHERE firstname = 'Миша'
```

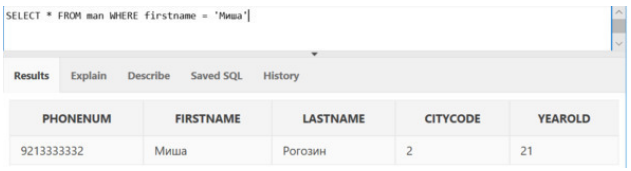


Рисунок 15. Запрос к таблице MAN: выбираем людей с именем Миша

Если осуществляется сравнение строковых данных, то есть тип данных в колонке сравнения VARCHAR, VARCHAR2, то строка сравнения заключается в одинарные кавычки.

Примеры

Выбрать из таблицы MAN все колонки (*), где имя Миша (равно Миша).

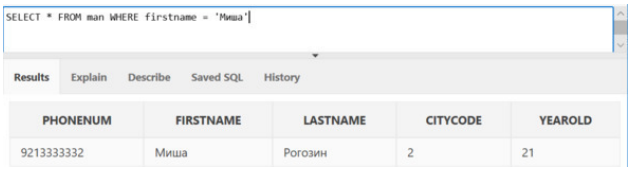


Рисунок 16. Запрос к MAN, где имя равно Миша

Выбрать из таблицы MAN все колонки (*), где имя не Олег (не равно Олег).

```
SELECT * FROM man WHERE firstname != 'Олег'
```

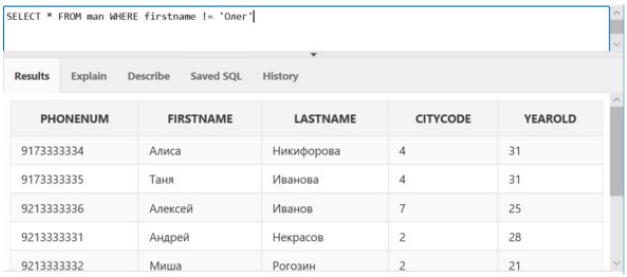


Рисунок 17. Запрос к MAN, где имя не равно Олег

Важные замечания

Несмотря на то что в SQL можно сочетать большие и маленькие буквы, в выражении в одинарных кавычках, при отборе и фильтрации текстовых данных регистр должен соблюдаться, иначе запрос отработает некорректно.

Выражение DATE'YYYY-MM-DD» работает только в СУБД ORACLE, в MS SQL SERVER и POSTGREESQL работа с данными типа «дата» осуществляется по-другому (смотрите подробности документации к этим СУБД).

Следует учитывать, что в некоторых типах баз данных для неравенства можно использовать <> или знак!=, подобную информацию необходимо уточнять в документации к СУБД.

Вопросы учеников

Какой способ написания команды SELECT наиболее часто используется?

Вы можете использовать любой способ записи, но наиболее удобным, с точки зрения синтаксиса и читаемости запроса, я считаю способ с указанием имени таблицы после оператора SELECT.

Так все-таки какой смысл в этой звездочке вместо перечисления колонок?

SELECT * выведет информацию о всех колонках в заданной таблице, и это можно использовать, чтобы посмотреть, какие именно колонки присутствуют и как они называются.

Мы можем использовать форму записи с именем таблицы в фильтре WHERE?

Да, и вот пример. SELECT * FROM MAN WHERE MAN.FIRSTNAME= «Олег».

Контрольные вопросы и задания для самостоятельного выполнения

1. Чем отличаются разные формы записи SQL-запроса SELECT?
2. Выбрать из таблицы MAN (*) людей, где возраст (YEAROLD) человека больше 30 лет.
3. Выбрать из таблицы городов все колонки (*), где город (CITYNAME) называется Москва.
4. Выбрать названия (CITYNAME) городов CITY с населением (PEOPLES) больше миллиона человек.
5. Выбрать телефоны людей из MAN, чья фамилия (LASTNAME) не Денисов.
6. Выбрать информацию о машинах car (*) синего цвета (COLOR).

Шаг 12. Более сложные условия. Знакомимся с логикой выбора строк

Введение

Язык SQL позволяет задавать и более сложные фильтры отбора строк с помощью оператора WHERE. Для этого в языке SQL применяются **логические операнды**, позволяющие комбинировать несколько условий, создавать тем самым сложные логические выражения.

Теория и практика

Итак, **логические операнды** позволяют объединять несколько условий, чтобы создать более сложные критерии выбора строк в операторе WHERE. Разберемся поподробнее, как это работает.

усл1 AND усл2 – логическое И, позволяет объединить несколько условных выражений, так что запрос вернет строку таблицы, если каждое из условий будет верным.

усл1 OR усл2 – логическое Или, позволяет выбрать строки, если одно из заданных условий верно.

NOT усл – логическое отрицание, выбирает строки, если выражение полностью неверно.

AND OR и NOT – как указано выше, можно гармонично сочетать в запросе.

Синтаксис

```
SELECT перечень колонок таблицы через запятую или * FROM
Tablename
WHERE условие1 AND условие2 OR  условие3 NOT условиеn
```

Где TABLENAME – имя таблицы, а условие1…условиеn – различные условия (WHERE) в SQL-запросе.

Последовательность логических операндов может комбинироваться.

Примеры

Разберем действие данных логических операндов на примерах:

Выбрать из таблицы AUTO машины (*) BMW синего цвета (COLOR).

```
SELECT * FROM auto WHERE color = 'СИНИЙ' and mark='BMW'
```

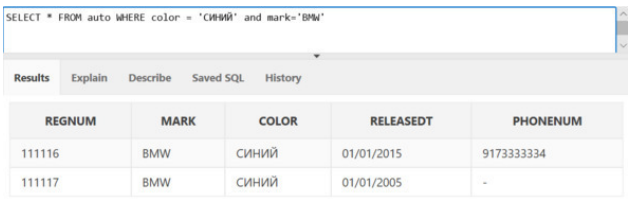


Рисунок 18. Запрос на синие авто BMW

Выбрать из таблицы MAN имена (FIRSTNAME) и фамилии (LASTNAME) людей, которым больше 27 лет и меньше 41 года (YEAROLD).

```
SELECT firstname, lastname FROM man WHERE yearold > 27 and yearold <41
```

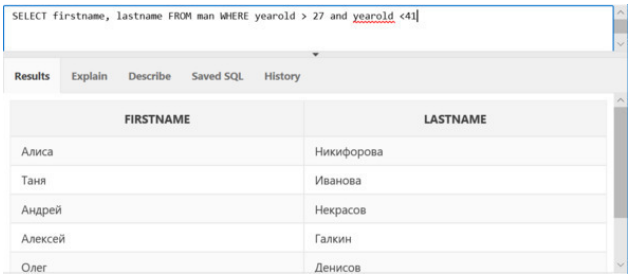


Рисунок 19. Запрос к MAN, где возраст больше 27 и меньше 41

Выбрать из таблицы MAN имена и фамилии людей с именем (FIRSTNAME) Андрей, которым больше 27 лет (YEAROLD).

```
SELECT firstname, lastname FROM man WHERE firstname='Андрей' and yearold > 27
```

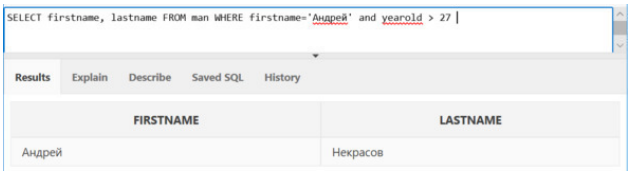


Рисунок 20. Запрос к таблице MAN: Андрей, возраст больше 27 лет

Выбрать из таблицы MAN имена (FIRSTNAME) и фамилии (LASTNAME) людей, которым не больше 27 лет (YEAROLD).

```
SELECT firstname, lastname FROM man WHERE firstname='Андрей' and not  yearold > 27
```

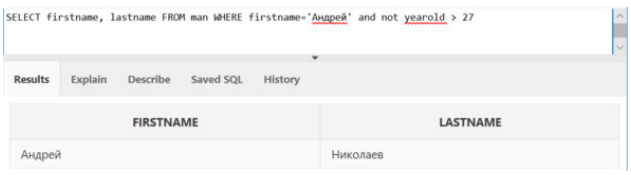


Рисунок 21. Запрос к таблице MAN: Андрей, возраст больше 27 лет

Выбрать из таблицы MAN имена (FIRSTNAME) и фамилии (LASTNAME) людей с именем Андрей или Алексей.

```
SELECT firstname, lastname FROM man WHERE firstname='Андрей' or firstname= 'Алексей'
```

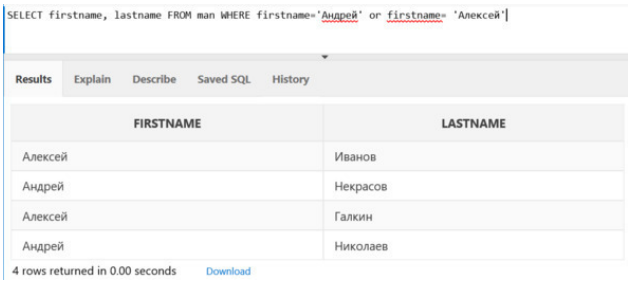


Рисунок 22. Запрос к таблице MAN: Андрей и Алексей

Выбрать из таблицы CITY города (*) с населением (PEOPLES) 400, 500 тысяч жителей.

```
SELECT * FROM city WHERE peoples=300000 or peoples=400000
```

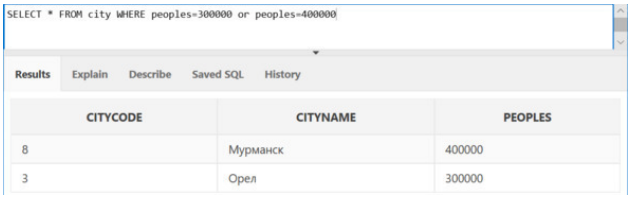


Рисунок 23. Запрос: города с населением 300 и 400 тысяч

Выбрать из таблицы CITY города (*) с населением (PEOPLES) не 400, 500 тысяч жителей.

```
SELECT * FROM city WHERE not( peoples=300000 or peoples=400000)
```

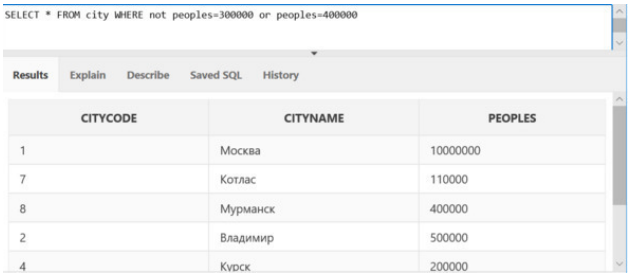


Рисунок 24. Запрос: города с населением не 300 и не 400 тысяч

Важные замечания

Несколько условий можно объединять скобками, например из таблицы MAN нам необходимо вывести те строки, где людям 25 или 28 лет с именем Иван. Это можно сделать с помощью следующего запроса:

```
SELECT * FROM man WHERE (yearsold = 25 or yearsold = 28) and firstname = 'Иван'
```

Очень важно понимать отличие AND от OR, например **выведите авто с марками LADA и BMW** – в этом запросе необходимо использовать инструкцию OR и ни в коем случае не AND.

```
SELECT * FROM auto WHERE mark = 'LADA' or mark = 'BMW'
```

Вопросы учеников

Можно ли неравенство заменить на инструкцию NOT?

Да, в большинстве запросов можно так сделать.
Например, запросы

```
SELECT * FROM man WHERE firstname <> 'Иван'
и
SELECT * FROM man WHERE not firstname = 'Иван'
```

идентичны.

Как поменять несколько условий, перечисленных после WHERE в SQL-запросе, на обратные?

Вопрос не очень понятен, но предположим, у нас есть запрос, который возвращает все строки из таблицы AUTO с марками BMW и LADA.

```
SELECT * FROM AUTO WHERE mark = 'BMW' or mark='LADA'
```

Если необходимо посмотреть авто не BMW и LADA, то запрос обретет следующий вид:

```
SELECT * FROM AUTO WHERE not(mark = 'BMW' or mark='LADA')
```

Контрольные вопросы и задания для самостоятельного выполнения

1. Сколько строк вернет запрос?

```
SELECT * FROM city WHERE cityname = 'Москва' and cityname = 'Воронеж'
```

2. Выберите из таблицы CITY (*) город, где 200 тысяч жителей (PEOPLES); город с наименованием Москва (CITYNAME).

3. Выберите из таблицы CITY города с населением (PEOPLES) не 500 тысяч жителей.

4. Выберите из таблицы AUTO (*) все синие AUDI (COLOR, MARK).

5. Выберите из таблицы AUTO номера (regnum) и марки (MARK) всех VOLVO и BMW.

6. Выберите из таблицы MAN имена, фамилии (FIRSTNAME, LASTNAME) и возраст людей (YEAROLD), которым больше 29 и меньше 35 лет.

Шаг 13. Сортировка результатов запросов

Введение

Данные, выводимые с помощью запроса **SELECT**, часто необходимо сортировать по одному или нескольким выводимым колонкам.

Для этого в языке SQL есть специальный оператор **ORDER BY**, который используется в команде **SELECT** и пишется в конце запроса. Для того чтобы изменить порядок сортировки на обратный, используется ключевое слово **DESC**.

Теория и практика

ORDER BY используется для сортировки выводимых данных по заданным колонкам: от большего к меньшему, и наоборот.

Текстовые данные будут отсортированы от А до Я или же от Я до А – при использовании ключевого операнда **DESC**.

Числовые данные сортируются от меньшего числа к большему числу и от большего числа к меньшему при использовании ключевого операнда **DESC**.

Данные DATE также: от меньшей даты к большей, и наоборот.

Синтаксис

```
SELECT перечень колонок через , или * FROM  таблица
WHERE условия выборки
ORDER BY  колонка сортировки1, колонка сортировки2, колонка сортировкиN
```

Где таблица – наименование таблицы, условия выборки которой могут быть объединены логическими операндами.

Примеры запросов

Выведите все записи из таблицы AUTO, отсортируйте автомобили по марке (MARK).

```
SELECT * FROM auto ORDER BY mark
```

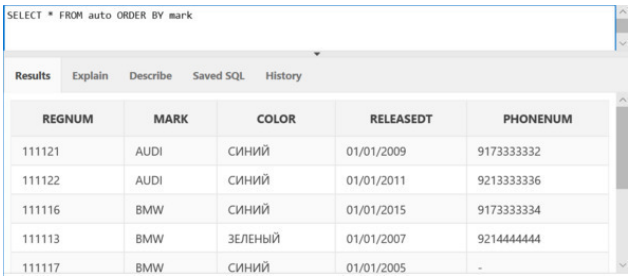


Рисунок 25. Выбрать из таблицы AUTO, отсортировать по марке

Выведите все записи из таблицы AUTO, отсортируйте автомобили по дате создания и цвету в обратном порядке (RELEASED, MARK).

```
SELECT * FROM auto ORDER BY releasedt desc, mark desc
```

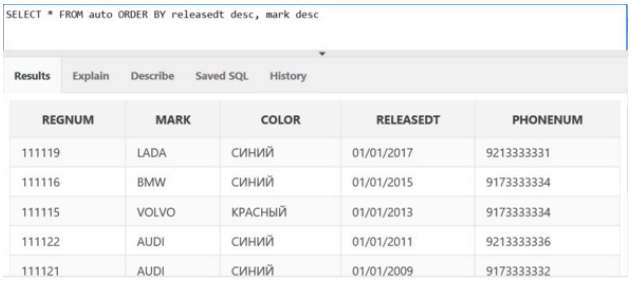


Рисунок 26. Запрос к таблице AUTO: сортировка по дате и цвету авто

Сортировка по двум колонкам RELEASED от меньшей даты к большей, MARK от A—Z.

Выберите только те записи из таблицы MAN, где возраст (YEAROLD) человека больше 30 лет; отсортируйте выбранные записи по возрасту, сортировку произведите в обратном порядке.

```
SELECT * FROM man WHERE yearold>30 ORDER BY yearold DESC
```

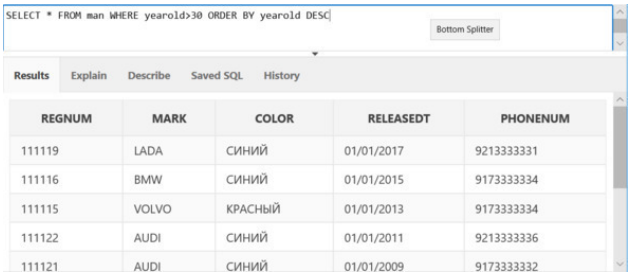


Рисунок 27. Выбор из таблицы MAN, где возраст больше 30 лет; сортировка по году выпуска

Пример использования ORDER BY и WHERE

Выберите наименования городов (CITYNAME) и население (PEOPLES) из таблицы CITY, отсортируйте выбранные данные в обратном порядке по наименованию города и количеству населения (PEOPLES).

```
SELECT cityname, peoples from CITY order by cityname desc, peoples desc
```

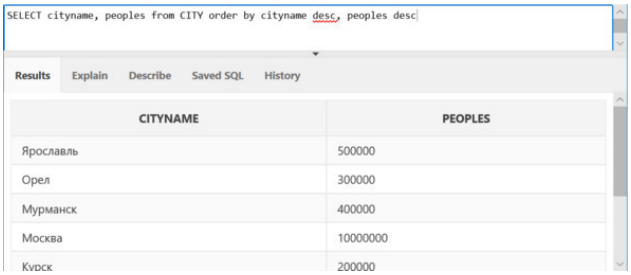


Рисунок 28. Запрос к CITY: сортировка по названию и населению в обратном порядке

Важные замечания

Следует пояснить, как работает сортировка по нескольким колонкам.

Сначала данные сортируются по первой колонке в инструкции **ORDER BY**, затем уже в рамках этой сортировки данные сортируются по второму признаку, по второй колонке из инструкции **ORDER BY**, затем третьей и так далее.

Команда сортировки **ORDER BY** выполняется в запросе последней, сортируется итоговое выражение запроса.

Это важная информация, и она пригодится нам в дальнейшем, при создании сложных SQL-запросов.

Вопросы учеников

Как сделать, если я хочу одну колонку отсортировать по возрастанию, а две другие по убыванию?

Давайте рассмотрим пример.

Выберите наименования городов (CITYNAME) и население (PEOPLES) из таблицы CITY, отсортируйте выбранные данные по коду города (CITYCODE), в обратном порядке по наименованию города и количеству населения.

```
SELECT citycode, cityname, peoples from CITY ORDER BY citycode , cityname desc, peoples
desc
```

Я знаю, что для инструкции ORDER BY существует альтернативный синтаксис. Расскажите о нем.

Перепишем этот запрос с использованием альтернативного синтаксиса:

```
SELECT citycode ,cityname, peoples FROM city ORDER BY 1 , 2 desc, 3 desc
```

То есть вместо названий колонок таблиц в инструкции ORDER BY мы используем порядковый номер колонки в нашем запросе – у CITYCODE он равен 1, у CITYNAME 2, у PEOPLES 3.

Можно ли отсортировать запрос по полю, которого нет в запросе, но которое есть в таблице после инструкции FROM?

Да, синтаксис SQL это допускает, и вот пример:

```
SELECT cityname, peoples FROM city ORDER BY citycode
```

Контрольные вопросы и задания для самостоятельного выполнения

1. Выберите записи из таблицы CITY, где в названии города (CITYNAME) есть слог «ем», отсортируйте запрос по названию города (CITYNAME) и по популяции (PEOPLES) в обратном порядке.

2. Выберите все записи из таблицы AUTO, отсортируйте записи по цвету (COLOR) и по марке (MARK) автомобиля в обратном порядке.

Шаг 14. Ограничение на количество выбранных строк ROWNUM, TOP (n)

Введение

Иногда запросы строятся таким образом, что на экран сразу выводится множество строк.

А что если нам необходимо ограничить количество строк выводимой информации, то есть из десятков тысяч строк нам достаточно нескольких строк для анализа информации?

Теория и практика

Для решения этой задачи в разных диалектах языка SQL используются разные синтаксические конструкции: в MS SQL это конструкция TOP, в ORACLE есть специальный предикат ROWNUM, в PostgreSql, MYSQL для этого существует конструкция LIMIT.

Разберем диалект SQL ORACLE.

Конструкция ROWNUM позволит ограничить количество выводимых строк на заданную величину.

Синтаксис

```
SELECT перечень полей или * FROM таблица
WHERE rownum<колстрок +1 and or not  доп условия
```

Примеры
Вывести первые 5 строк из таблицы AUTO.

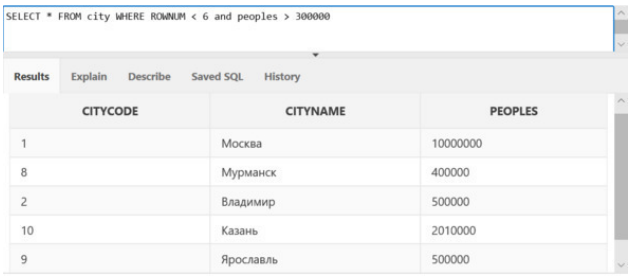
```
SELECT rownum, auto.* FROM auto WHERE ROWNUM < 6
```

ROWNUM	REGNUM	MARK	COLOR	RELEASEDT	PHONENUM
1	111119	LADA	СИНИЙ	01/01/2017	9213333331
2	111115	VOLVO	КРАСНЫЙ	01/01/2013	9173333334
3	111121	AUDI	СИНИЙ	01/01/2009	9173333332
4	111122	AUDI	СИНИЙ	01/01/2011	9213333336
5	111114	LADA	КРАСНЫЙ	01/01/2008	9152222221

Рисунок 29. Запрос с ограничением строк (первые 5)

Вывести первые 5 строк из таблицы CITY, где население (PEOPLES) городов больше 300 000.

```
SELECT * FROM City WHERE ROWNUM < 6 and peoples > 300000
```



CITYCODE	CITYNAME	PEOPLES
1	Москва	10000000
8	Мурманск	400000
2	Владимир	500000
10	Казань	2010000
9	Ярославль	500000

Рисунок 30. Выбрать первые 5 строк, запрос города: население больше 300 000

Важные замечания

Если использовать ROWNUM совместно с сортировкой, то необходимо прибегнуть к специальному приему, иначе ROWNUM не будет работать как нужно и запрос вернет неверные данные.

Правильно следует написать так:

```
SELECT * FROM (SELECT поле1, поле2 или * FROM имя таблицы
WHERE условия ORDER BY поля сортировки)
```

Вывести первые 5 автомобилей, отсортированных по дате производства (RELEASEDT).
Ошибочный запрос:

```
SELECT * FROM auto WHERE ROWNUM < 6 ORDER BY releasedt
```

Правильный запрос:

```
SELECT * FROM (SELECT * FROM Auto ORDER BY releasedt) WHERE ROWNUM<кол строк +1
```

Вопросы учеников

Покажите, как ограничивать вывод строк в MYSQL, MS SQL и PostgreSQL.

Разберемся на примере.

Вывести первые 5 строк из таблицы CITY, где население (PEOPLES) городов больше 3000.

ORACLE SQL

```
SELECT * FROM City WHERE ROWNUM < 6 and peoples > 3000
```

MY SQL

```
SELECT * FROM City WHERE peoples > 30000 limit 5
```

MS SQL

```
SELECT top 5 * FROM City WHERE peoples > 30000
```

PostgreSQL

```
SELECT * FROM City WHERE peoples>3000 limit 5
```

Контрольные вопросы и задания для самостоятельного выполнения

1. Поясните, как использовать ROWNUM с сортировкой в SQL-запросе.
2. Выведите на экран первые 3 строчки из таблицы CITY.
3. Выведите на экран первые 4 строчки из таблицы CITY, отсортированные по количеству населения (PEOPLES).
4. Выведите на экран данные из таблицы MAN.

Шаг 15. Вставка данных в таблицу – INSERT

Введение

Ранее мы с вами изучили, как создавать таблицы и изменять структуру таблиц, а также как извлекать данные из таблиц на экран с помощью команды SELECT. Мы выбирали данные, которые уже были в таблицах. Настало время узнать о том, как добавить необходимые данные в таблицу.

Теория и практика

Для добавления данных в SQL используется команда INSERT.
Команда INSERT существует в SQL в двух вариантах.
Во-первых, для добавления заданных значений – VALUES.
Синтаксис

```
INSERT INTO  имя таблицы (колонки через запятую)
VALUES (значения через запятую);
```

Важно помнить, что количество колонок в скобках и количество добавляемых значений должны соответствовать друг другу.

Также очень важно, чтобы колонки и значения соответствовали по типу данных.

Примеры
Добавим новые сведения о человеке в таблицу MAN:

```
INSERT INTO man(phonenum, firstname, lastname, citycode, yearold)
VALUES('120120120','Максим','Леонидов',2,25);
Commit;
```

В таблицу MAN добавлена строка о человеке с номером телефона «120120120», именем «Максим», фамилией «Леонидов», кодом города 2, ему 25 лет (PHONENUM, FIRSTNAME, LASTNAME, CITYCODE, YEAROLD).

Добавим сведения о новой машине в таблицу AUTO:

```
INSERT INTO auto(regnum, mark, color, released, phonenum)
VALUES('128877655','LADA','КРАСНЫЙ',date'2001-01-01','123114444');
Commit;
```

В таблицу AUTO добавлена строка о авто с номером «128877655» марки «LADA», цвет «КРАСНЫЙ», дата выпуска «2001-01-01», «123114444».

Второй вариант, для добавления из запроса SELECT

Здесь источником данных являются не одиночные значения, а настоящий запрос SELECT.

Синтаксис:

```
INSERT INTO   имя таблицы (колонки через запятую)
SELECT FROM имя таблицы WHERE условия
```

Примеры

В таблицу CITY1 добавить все строки из CITY, где население больше миллиона человек.

```
INSERT INTO city1(citycode, cityname, peoples)
SELECT citycode, cityname, peoples WHERE peoples>1000000
```

В таблицу MAN1 добавить все строки из MAN.

```
INSERT INTO man1(phonenum, firstname, lastname, citycode, yearold)
SELECT phonenum, firstname, lastname, citycode, yearold FROM man;
```

Важные замечания

Команда INSERT является командой модификации данных, поэтому ее выполнение необходимо завершить одной из следующих команд:

COMMIT – фиксация изменений;

ROLLBACK – откат изменений.

Только после выполнения фиксации изменений данные появятся в базе.

Обратите внимание: при добавлении данных типа дата (**DATE**) мы использовали следующую конструкцию: **DATE'YYYY-MM-DD»**, YYYY – текущий год, MM – месяц, DD – день.

При использовании **INSERT** с запросом **SELECT** также необходимо, чтобы последовательность и типы колонок, перечисленных после команды **INSERT**, совпадали с последовательностью и типами колонок в запросе **SELECT**.

Вопросы учеников

Все же зачем применять COMMIT или ROLLBACK и что будет, если эти команды не выполнять?

Во многих СУБД применяется транзакционная модель, что это такое – узнаем чуть позже, но сейчас необходимо понимать, что при запуске операций изменения, вставки, удаления данных эти изменения появятся в базе только после выполнения команды COMMIT.

Я использовал только команду INSERT и не применял ни COMMIT, ни ROLLBACK, но данные все равно появились в базе. Почему так вышло?

Некоторые редакторы поддерживают режим автофиксации изменений, то есть выполняют команду COMMIT за вас.

Если я перечислю не все колонки, какие есть в таблице, куда мы добавляем данные, будет ошибка?

Необязательно. В колонки, которые вы не перечислили, будет добавлено пустое (NULL) значение, ошибка возникнет только если на колонках этой таблицы есть ограничение NOT NULL или первичный ключ.

Контрольные вопросы и задания для самостоятельного выполнения

1. Повторите материалы записей из данного шага.
2. Добавьте в таблицу городов новый город; код города и название придумайте сами.
3. В таблицу CITY1 добавьте все строки из CITY, где популяция меньше полумиллиона человек.
4. В таблицу AUTO добавьте новую машину 999999 ЛАДА, цвет зеленый, 1999 года выпуска.

День четвертый

Шаг 16. Обновление данных – UPDATE

Введение

Мы научились добавлять данные в таблицу, но очень часто возникают ситуации, когда нам необходимо обновить, пересчитать данные в одной или нескольких колонках.

Теория и практика

Обновление данных в строчках и ячейках таблицы осуществляется с помощью команды UPDATE.

Синтаксис команды

```
UPDATE table_name SET column1 = знач1, column 2 = знач2 , column n = значn WHERE
условия отбора строк для обновления
```

Здесь TABLE_NAME – имя таблицы, где обновляются данные.

Здесь через запятую перечислены имена колонок = значения, а после WHERE описываются условия для отбора обновляемых строк.

Примеры

Добавить к наименованию (CITYNAME) города CITY с кодом (CITYCODE) меньше 2 символ #.

```
UPDATE city SET cityname = cityname| |'#' WHERE citycode<2
```

К населению города с населением (PEOPLES) больше 1 000 000 добавить 10.

```
UPDATE city SET peoples = peoples+10 WHERE peoples > 1000 000
```

У всех у людей MAN с именем (FIRSTNAME) Алексей поменять имя на Максим.

```
UPDATE man SET firstname = 'Максим' WHERE firstname = 'Алексей'
```

У всех у людей с телефоном (PHONENUM), заканчивающихся на 3, поменять имя (FIRSTNAME) на Александр.

```
UPDATE man SET firstname = 'Александр' WHERE phonenum like '%3'
```

Применение обновления UPDATE возможно также без использования предиката WHERE, в этом случае обновятся все строки указанной таблицы в заданных колонках.

Пример, который не нужно выполнять, но обязательно следует изучить.

Обнулить колонку PEOPLES в таблице CITY.

```
UPDATE city SET proples = 0
```

Во всей таблице значение колонки PEOPLES будет равно 0.

Важные замечания

Команда обновления данных UPDATE тоже должна завершаться выполнением **COMMIT –** фиксацией изменений, либо **ROLLBACK –** откатом изменений.

При выполнении обновления данных следует быть предельно аккуратным и внимательным: последующее восстановление измененных данных, возврат таблицы к состоянию до выполнения UPDATE очень часто бывает затруднителен.

Команда UPDATE TABLE set column1 = val1, columnn = valn без инструкции WHERE обновит значения во всех строчках таблицы TABLE колонок column1, column, будьте особо осторожны при выполнении таких команд.

Вопросы учеников

Если мы обновляем текстовое поле, необходимо ли значение писать в одинарных кавычках?

Да, при обновлении текстовых данных необходимо заключать значения в одинарные кавычки.

Приведите пример обновления данных типа DATE.

Пример: обновите даты выпуска автомобилей LADA на 01.01.2010.

```
UPDATE auto SET releasedate = date'2010-01-01' WHERE mark = 'BMW'
```

Я выполнил команду UPDATE, попытался выполнить ее повторно из другого окна, но ничего не получилось и программа зависла, в чем причина?

Причина в том, что вы забыли зафиксировать изменения с помощью команды COMMIT. При выполнении первого UPDATE на строки были установлены блокировки, повторное обновление не выполнилось, так как изменения не были вами зафиксированы. Об этом будет более подробная информация в книге в дальнейшем.

Контрольные вопросы и задания для самостоятельного выполнения

1. Сколько строк обновит команда UPDATE AUTO SET MARK = «BMW» WHERE 1=0?
2. Добавить 1 к возрасту людей (MAN) с именем Андрей.
3. Добавить 2 к возрасту людей (MAN), чей возраст больше 19 лет.
4. К имени людей (MAN), чей телефон содержит 915, добавить префикс 915.
5. У всех у людей (MAN) с телефоном, чей возраст больше 22 лет, поменять имя на Роман.

Шаг 17. Удаление данных – DELETE

Введение

Иногда некоторые данные нам необходимо удалять из таблиц. Для этого в языке SQL есть специальная команда – DELETE.

Теория и практика

Команда DELETE.

Синтаксис

```
DELETE имя таблицы WHERE условие для удаления строк
```

Здесь Имя таблицы – имя таблицы, из которой мы удаляем строки, WHERE – условие для отбора строк, которые мы удаляем.

Примеры

Удалить записи из таблицы CITY1 с кодами городов CITYCODE 7, 9.

```
DELETE city1 WHERE citycode = 7 or citycode =9
```

Удалить из таблицы AUTO1 все автомобили BMW (MARK).

```
DELETE auto1 WHERE mark = 'BMW'
```

Удалить из таблицы AUTO1 все автомобили с датой выпуска больше 01.01.2017.

```
DELETE auto1 WHERE releasedate > date'2017-01-01'
```

Завершите выполнение командой COMMIT для фиксации изменений.

Важные замечания

Данная команда может удалить записи только в одной заданной таблице.

Команда DELETE также должна завершаться инструкцией COMMIT для фиксации изменений в базе.

Следует помнить, что язык SQL в ORACLE поддерживает механизм транзакций, поэтому модификации данных, в том числе и операции удаления, чаще всего необходимо завершать командой **COMMIT** или **ROLLBACK**.

COMMIT – для фиксации изменений;
ROLLBACK – для отката изменений.

Команда DELETE без инструкции WHERE очищает все строки в таблице, поэтому при использовании команды DELETE надо соблюдать предельную осторожность.

Вопросы учеников

Чем отличается команда DELETE от команды DROP?

Это совершенно разные команды. Я предлагаю повторить материал на эту тему и ответить на данный вопрос самостоятельно.

Как еще можно очистить таблицу?

Для этого есть специальный оператор TRUNCATE TABLE TABLENAME. При его выполнении происходит быстрая очистка всех записей таблицы. Мы еще будем изучать эту тему в следующих шагах.

Я пытаюсь удалить записи из некоторой таблицы, но возникает ошибка.

Вероятно, есть связанные записи в другой таблице по внешнему ключу. Сначала необходимо удалить записи в связанной таблице.

Контрольные вопросы и задания для самостоятельного выполнения

1. Чем команда DELETE TABLE_NAME отличается от команды DROP TABLE TABLE_NAME?
2. Напишите команду для удаления из таблицы AUTO таких записей, где дата выпуска авто больше 2018 года.
3. Повторите материалы данного шага.

Шаг 18. Псевдонимы

Введение

В языке SQL есть такая синтаксическая конструкция, как псевдонимы.

С помощью псевдонимов мы можем большим или сложным наименованиям таблиц или колонок таблиц в запросе SQL присвоить более короткие, удобные и понятные нам псевдонимы (ALIAS).

Теория и практика

Псевдонимы для колонок, выводимых в запросе, задаются с помощью инструкции AS, псевдонимы же для таблиц указываются сразу же после имени таблицы.

Примеры

Вывести из таблицы MAN колонки «имя», «фамилия» и «возраст».

Для таблицы задать псевдоним m для колонки имени (FIRSTNAME), для колонки фамилии (LASTNAME) – Fml.

```
SELECT firstname as Im, lastname as Fml FROM man m
```

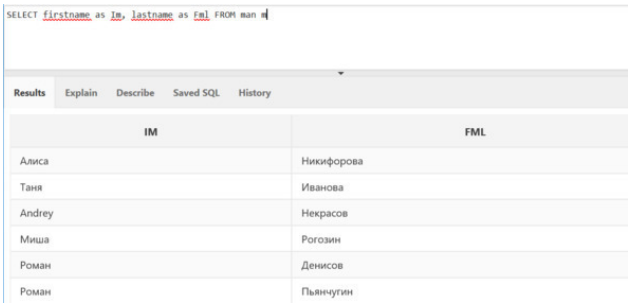


Рисунок 31. Демонстрация работы псевдонимов: запрос

Вывести из таблицы AUTO марку и цвет автомобиля.

- Для таблицы AUTO задать псевдоним AV;
- для колонки «марка» задать псевдоним Mr;
- для колонки «цвет» задать псевдоним CV.

Вывести только автомобили с годом выпуска больше 01.01.2001.

```
SELECT mark as Mr, color as CV FROM auto Av WHERE av.releasedt > date'2001-01-01'
```

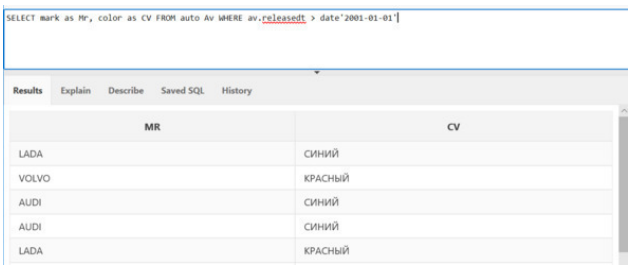


Рисунок 32. Запрос к AUTO: псевдонимы

Обратите внимание, как формируется условие для при обращении к колонке типа DATE.

Важные замечания

Псевдонимы не могут повторяться в рамках одного запроса и подзапроса, то есть их имена должны быть уникальными.

Если мы задали псевдоним для таблицы, из которой SQL-запрос выбирает данные, то и в условии WHERE мы также должны использовать заданный псевдоним.

Вопросы учеников

Если мы используем псевдоним для таблицы и псевдонимы для колонок, должны ли мы обращаться к колонкам в инструкции WHERE по псевдонимам колонок?

Как ни странно, так делать нельзя. Вы должны указать именно настоящее имя колонки в этом случае.

Пример

```
SELECT MARK as mr, COLOR  as cl FROM AUTO WHERE color='СИНИЙ'
```

Запись вида

```
SELECT MARK as mr, COLOR  as cl FROM AUTO WHERE cl='СИНИЙ'
```

будет неверной.

Можно ли нескольким выводимым колонкам запроса SQL задать псевдоним, а другим не задавать?

Конечно; вот пример такого запроса:

```
SELECT firstname as fn, lastname  FROM man.
```

Контрольные вопросы и задания для самостоятельного выполнения

1. Найдите ошибку в следующем запросе:

```
SELECT citycode as cc, cityname as cn  FROM city cc
```

2. Найдите ошибку в еще одном запросе:

```
SELECT citycode as cc, cityname as cn  FROM city c WHERE city.citycode = 1;
```

3. Выведите с помощью запроса SQL наименование города из таблицы CITY. Для таблицы задайте псевдоним GR, для колонки «наименования» задайте псевдоним NM.

4. Выведите количество жителей в городе Москва, для колонки PEOPLES задайте псевдоним CLZ, для таблицы задать псевдоним MS.

Шаг 19. BETWEEN

Введение

В языке SQL есть специальная конструкция, которая позволяет работать с интервалами – своего рода фильтр, позволяющий выбирать данные, соответствующие заданному интервалу значений. Этот оператор называется BETWEEN и может использоваться как в выборке SELECT, так и в операциях модификации и удаления данных (UPDATE, DELETE).

Теория и практика

Рассмотрим применение BETWEEN в команде SELECT.
Синтаксис:

```
SELECT перечень полей или * FROM TAB1
WHERE поле1 BETWEEN нижняя граница интервала AND верхняя граница интервала
Из  TAB1 Будут выбраны строки,  соответствующее интервалу ЗАДАННОМУ В BETWEEN
```

Примеры
Выбираем из таблицы MAN (*) всех людей, чей возраст (YEAROLD) от 25 до 32 лет.

```
SELECT * FROM man WHERE yearold BETWEEN 25 and 32
```

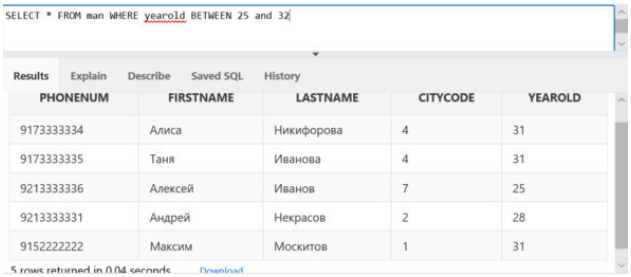


Рисунок 33. Запрос к MAN: возраст от 25 до 32

Выбираем города CITY * с населением (PEOPLES) от 300 000 до 1 000 000.

```
SELECT * FROM city WHERE peoples BETWEEN 300000 and 1000000
```

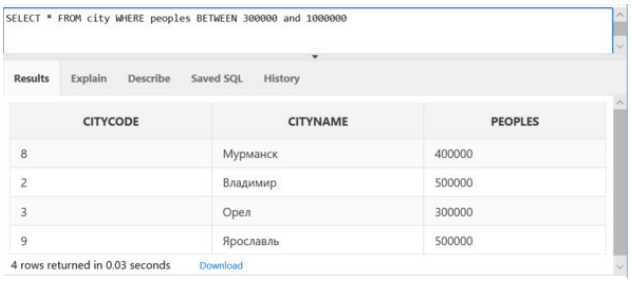


Рисунок 34. Запрос к таблице CITY: население от 300 000 до 1 000 000

Выбираем людей * с именем Андрей (FIRSTNAME) и возрастом (YEAROLD) от 25 до 28 лет.

```
SELECT * FROM man WHERE yearold BETWEEN 25 and 28 and firstname = 'Андрей'
```

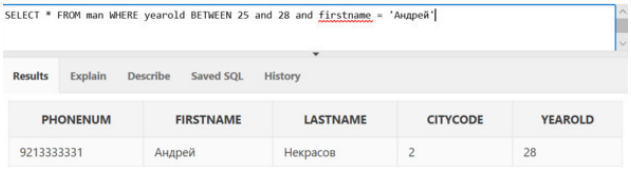


Рисунок 35. Запрос к таблице MAN: население от 25 до 28 лет, имя Андрей

Выбираем людей * с возрастом не в интервале от 25 до 28 лет.

```
SELECT * FROM man WHERE NOT yearold BETWEEN 25 and 28
```

Выбрать машины * с годом выпуска (RELEASEDT) от 2001 до 2014.

```
SELECT * FROM auto WHERE Releasedt BETWEEN date'2001-01-01' and date'2014-01-01'
```

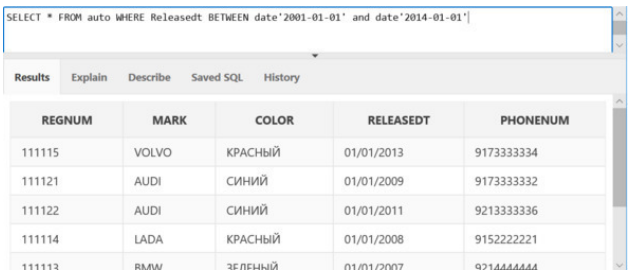


Рисунок 36. Запрос всех записей из AUTO

Важные замечания

Следует отметить, что использование BETWEEN разумно только для данных числовых типов и для данных типа DATE, что естественно, так как мы можем найти числа в заданном интервале и даты в заданном интервале, а вот со строкам и текстом это сделать затруднительно.

Также важное замечание: обратите внимание, что BETWEEN позволяет выбрать данные, принадлежащие и равные нижней и верхней границам заданного интервала.

Вопросы учеников

Оператор BETWEEN на что-то похож, можно ли обойтись без него?

Оператор BETWEEN создан для удобства, и, разумеется, его можно заменить несколькими логическими выражениям, и вот пример.

Пример:

Выбираем города CITY * с населением (PEOPLES) от 300 000 до 1 000 000.

```
SELECT * FROM city WHERE peoples BETWEEN 300000 and 1000000
```

Не будем использовать BEETWEEN.

```
SELECT * FROM city WHERE peoples >=300000 and peoples <=1000000
```

Можно ли в запросе использовать несколько BETWEEN, объединенных логическими операндами?

Да, разумеется, такой запрос можно написать. Выберем из таблицы MAN людей с возрастом в интервалах от 20 до 30 и от 35 до 39 лет.

```
SELECT * FROM man WHERE yearold BETWEEN 20 and 30 or yearold BETWEEN 35 and 39
```

Вы говорили, что BETWEEN также можно использовать и в операторах модификации данных?

Да, как и все другие операторы, используемые для выбора строк в WHERE.

Примеры

```
UPDATE auto SET Releasedt= date'2014-01-01'   WHERE Releasedt BETWEEN date'2011-01-01'
and date'2014-01-01'
```

Обновим дату создания авто на 01.01.2014 для автомобилей с датой создания в интервале от 01.01.2011 до 01.01.2014.

DELETE MAN WHERE YEAROLD BETWEEN 7 AND 16

Удаление всей информации о людях, которым от 7 до 16 лет. Обратите внимание: возраст 7 и 16 входит в заданный интервал и эта информация также будет удалена.

Контрольные вопросы и задания для самостоятельного выполнения

1. Как можно заменить оператор BETWEEN и обойтись без него, чтобы запрос возвращал правильные данные?

2. Выберите города с населением от 40 000 до 2 000 000 человек, напишите SQL-запрос.

3. Выберите города с населением НЕ в интервале 40 000 до 2 000 000 человек, напишите SQL-запрос.

4. Выберите людей * с возрастом НЕ в интервале 25 до 28 лет, используйте NOT, напишите SQL-запрос.

5. Выберите машины с годом выпуска от 2007 до 2011, напишите SQL-запрос.

Шаг 20. DISTINCT, дубликаты значений

Введение

В определенных запросах SQL как неприятный побочный результат выводится множество одинаковых, повторяющихся записей. Иногда нам необходимо уйти от данных повторений, убрать дубли из результатов запроса.

Теория и практика

В SQL существует специальная команда DISTINCT, которая предназначена для выбора в запросе только уникальных значений, уникальных строк, то есть исключает из вывода повторения и дублирования строк.

Синтаксис

```
SELECT DISTINCT перечень полей или * FROM таблица WHERE условия
```

Примеры

Вывести из таблицы AUTO марки автомобилей (MARK), исключить повторения – дубли.
Запрос без DISTINCT:

```
SELECT mark FROM auto
```

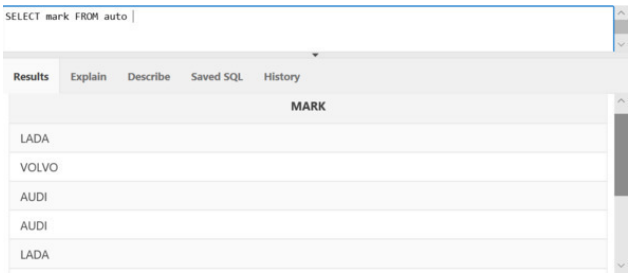


Рисунок 37. Запрос: дубли марок

Есть дублирующиеся марки авто AUDI, LADA в результате вывода.
Используем DISTINCT:

SELECT DISTINCT MARK FROM AUTO

```
SELECT DISTINCT mark FROM auto
```

Дубли строк не выводятся.

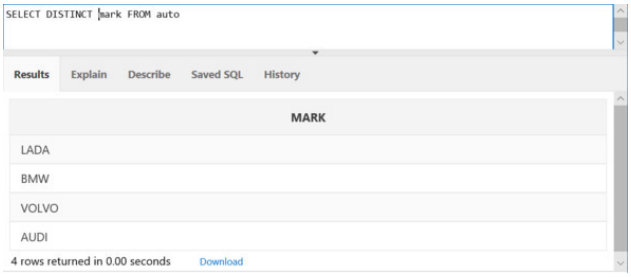


Рисунок 38. Запрос: марки авто

Вывести из таблицы AUTO марки и цвета автомобилей (MARK, COLOR), исключить повторения.

```
SELECT mark, color FROM auto
```

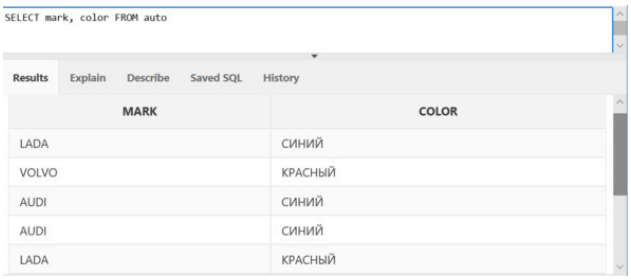


Рисунок 39. Запрос: марки авто и цвета

Несколько синих AUDI, используем DISTINCT.

```
SELECT DISTINCT mark, color FROM auto
```

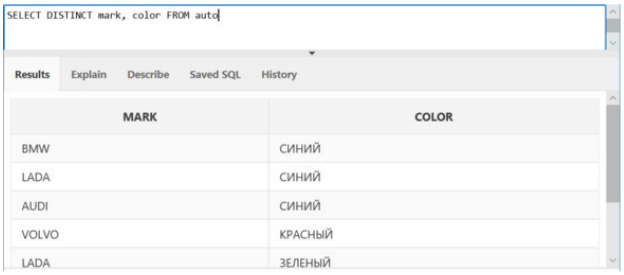


Рисунок 40. Запрос: марки, цвета авто, только уникальные записи

Важные замечания

Важно отметить, что дубли исключаются DISTINCT только из колонок, перечисленных в SELECT; других колонок DISTINCT не касается.

С помощью DISTINCT очень удобно просмотреть, какие вообще значения есть в заданной колонке, например автомобили каких цветов присутствуют в таблице AUTO:

```
SELECT DISTINCT COLOR FROM AUTO
```

Вопросы учеников

Есть ли еще какой-либо способ исключить дубли из запроса?

Да, подобный способ называется группировка записей, и мы изучим его позже.

Можно ли использовать DISTINCT с ROWNUM?

Да, но тогда оператор DISTINCT бесполезен, его использование потеряет смысл.

Можно ли использовать DISTINCT с WHERE?

Конечно, это возможно, нет никаких ограничений, и вот пример такого запроса. Выведите из таблицы AUTO автомобили марки LADA уникальных цветов.

```
SELECT DISTINCT mark, color FROM auto WHERE mark = 'LADA'
```

Контрольные вопросы и задания для самостоятельного выполнения

1. Повторите материалы данного шага.
2. Выведите из таблицы AUTO цвета автомобилей, исключите повторения, напишите SQL-запрос.
3. Выведите из таблицы MAN имена людей, исключите повторения, напишите SQL-запрос.

День пятый

Шаг 21. Математика в запросах

Введение

Мы уже использовали математику в SQL-командах ранее: вспомните, в одном из примеров использования UPDATE мы добавляли год к возрасту человека из MAN.

Так вот, в запросах SQL мы можем использовать результаты математических вычислений, причем мы можем как выводить результаты математических вычислений в колонках выбора SELECT, так и использовать математические выражения при формировании условий отбора строк WHERE.

Теория и практика

Для создания математических выражений в языке SQL используются следующие операции:

+ сложение;
– вычитание;
/ деление;
* умножение.

А также знакомые нам со школы функции:
Sqrt – квадратный корень;
MOD – остаток от деления;
trunc – округление до целого;
sIN – синус;
cos – косинус
(на самом деле этих функций больше, мы рассматриваем основные).

Все математические операции выполняются только для числовых значений, числовых колонок с типами NUMBER или производными от NUMBER (INt, float) – более подробную информацию можно посмотреть в документации к СУБД. То есть мы можем использовать в математических выражениях значения соответствующих колонок при выводе на экран и в фильтре WHERE.

Запомним, что математические операции используются также в критериях отбора строк WHERE.

Посмотрим, как это делается.
Вывести из таблицы MAN имя, фамилию и возраст (FIRSTNAME, LASTNAME, YEAROLD) человека, разделенный на 10.

```
SELECT firstname,lastname,yearold/10 as y10 FROM man
```

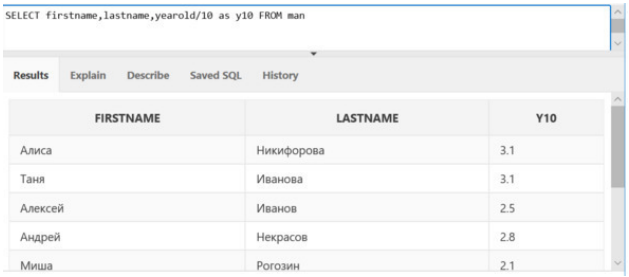


Рисунок 41. Запрос к MAN: результат математического вычисления

В колонке с псевдонимом y10 выводится результат выражения YEAROLD/10.

Вывести из таблицы MAN имя, фамилию и возраст человека (FIRSTNAME, LASTNAME, YEAROLD), умноженный на sIN (1), округлить до целого.

```
SELECT firstname,lastname,trunc(yearold*sin(1)) as ysin FROM man
```

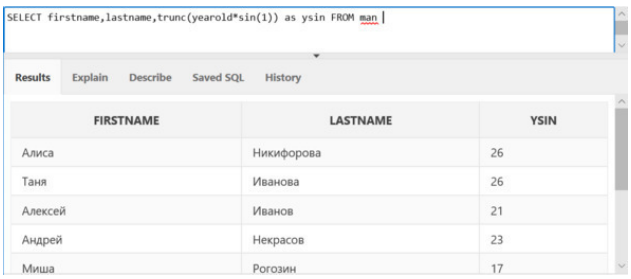


Рисунок 42. Запрос к MAN: результат вычисления sIN

Вывести из таблицы CITY записи (*), где квадратный корень от количества населения города больше 1000 (PEOPLES).

```
SELECT * FROM city WHERE sqrt(city.peoples)>1000
```

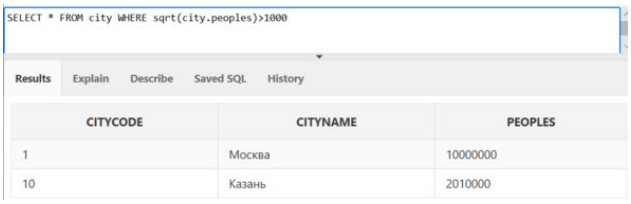


Рисунок 43. Запрос к CITY: квадратный корень больше 1000

Пример демонстрирует использование математического выражения sqrt(CITY.PEOPLES) при фильтрации строк в WHERE.

Вывести из таблицы CITY название города (CITYNAME), квадратный корень от количества населения (PEOPLES), где значение кода города (CITYCODE) делится нацело на 3.

```
SELECT cityname, sqrt(peoples) as spep FROM city WHERE mod(citycode,3) = 0
```

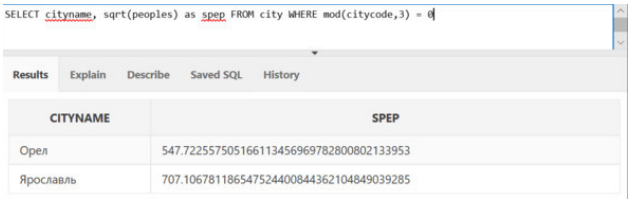


Рисунок 44. Запрос к CITY с математическим выражением

Вывести из таблицы CITY название города (CITYNAME), код города, разделенный на 3 (CITYCODE), где значение населения (PEOPLES), разделенное на 100, не больше 1 000 000.

```
SELECT cityname, citycode/3 as ccode FROM city WHERE Not peoples / 100 > 1000000
```

Рисунок 45. Запрос к CITY: математическое выражение WHERE и SELECTLISt

Важные замечания

При выполнении математических выражений иногда возникают ошибочные исключительные ситуации, например при делении некоторого значения на ноль: в этом случае запрос выдает некорректный результат или вообще не выполняется, но при этом некоторые среды разработки подавляют сообщения об ошибке.

Об этом следует помнить, когда вы пишете запросы, использующие математические выражения.

Вопросы учеников

Зачем использовать псевдонимы для колонок, которые рассчитываются как математические выражения?

Для удобного, понятного отображения названия колонки.

Где найти полный перечень математических функций, поддерживаемых СУБД?

В документации по вашей СУБД.

Контрольные вопросы и задания для самостоятельного выполнения

1. Вывести из таблицы MAN имя, фамилию и квадратный корень 10.
2. Вывести из таблицы MAN имя, фамилию и возраст человека, умноженный на cos (5).
3. Вывести из таблицы CITY записи (*), где популяция делится без остатка на 10 000.
4. Вывести из таблицы CITY название города, квадратный корень от популяции, умноженный на 10, где значение кода города делится нацело на 3.

Продолжением данного шага является следующий шаг, посвященный таблице DUAL.

Шаг 22. Запрос к результату выражения и специальная таблица DUAL

Введение

В SQL ORACLE диалекта есть специальная таблица DUAL. Если в других СУБД существуют альтернативные варианты синтаксиса, то для SQL ORACLE необходимо использовать таблицу DUAL.

Теория и практика

Данная таблица выполняет следующие задачи:

• делает удобным вывод в запросе «результаты вычислений математического выражения»;

• помогает при вычислении функции, которая возвращает одно значение.

При этом для результата вычисления данных выражений нам не требуется участие колонок из каких-либо таблиц в нашей базе данных.

Разберем на примерах:

Вычислить значение математического выражения 1000* (123—11) /100 с помощью SELECT.

```
SELECT 1000*(123-11)/100 as v FROM dual
```

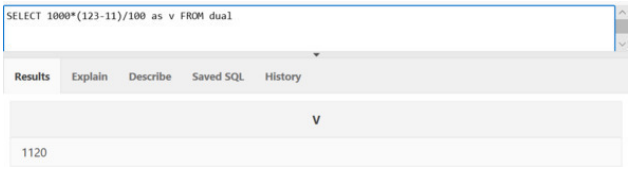


Рисунок 46. Вычисление выражения

Вычислить значение функции sIN (10) /100 с помощью SELECT.

```
SELECT sin(10)/100 as v FROM dual
```

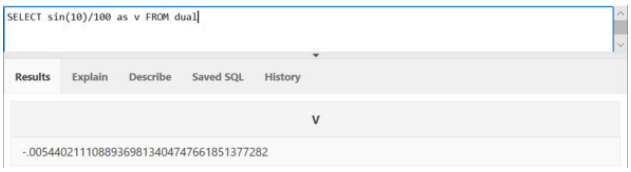


Рисунок 47. Вычисление выражения, таблица DUAL

Вычислить значение текущей даты с помощью SELECT.

```
SELECT sysdate  as dt FROM dual
```

– 21.11.1999 11:22

Вычислить значение sIN (1) *cos (1) с помощью SELECT.

```
SELECT sin(1)*cos(1) as n1 FROM dual
```

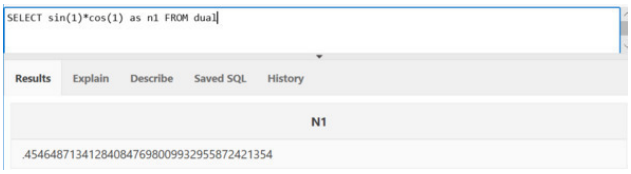


Рисунок 48. Вычисление выражения и таблица sIN (1) * cos (1)

Важные замечания

При обращении к таблице **DUAL** запрос возвращает всего одну строку, при попытке добавить запись в таблицу **DUAL** могут произойти непредсказуемые последствия, не рекомендую это делать.

Вопросы учеников

Вы говорили, что в других СУБД, например MS SQL, можно обойтись без таблицы DUAL?

Да в MS SQL, например, будет работать такой запрос: SELECT sIN (1), здесь нет FROM, ORACLE в этом случае более соответствует стандарту SQL ANSI 92.

Контрольные вопросы и задания для самостоятельного выполнения

1. Вычислить значение площади квадрата со стороной 5 с помощью SELECT и DUAL.
2. Вычислить значение площади круга с радиусом 7 помощью SELECT и DUAL.
3. Вычислить значение выражения 100*sIN (1) *cos (3) с помощью SELECT и DUAL.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «ЛитРес».

Прочитайте эту книгу целиком, купив полную легальную версию на ЛитРес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.