

Учебник SQL

в БД Oracle

Дата обновления: 2021.11.17

Содержание

| | |
|-------------------------------------------------|----|
| Введение в SQL..... | 7 |
| Диалекты SQL..... | 7 |
| Особенности SQL..... | 7 |
| Зачем изучать SQL..... | 7 |
| DML, DDL..... | 8 |
| Выполнение SQL. Облачные сервисы..... | 9 |
| LiveSQL..... | 9 |
| SQL Fiddle..... | 10 |
| Запуск примеров учебника..... | 11 |
| Инструменты для работы с БД Oracle..... | 12 |
| Средства разработки..... | 12 |
| Проектирование БД..... | 12 |
| Таблицы..... | 13 |
| Создание таблицы..... | 13 |
| Создание таблицы с несколькими полями..... | 13 |
| Значения по умолчанию..... | 14 |
| Понятие NULL. Not-null колонки..... | 14 |
| Комментарии к таблице, колонкам..... | 15 |
| Основные типы данных..... | 16 |
| Varchar2..... | 16 |
| Number..... | 16 |
| Date..... | 17 |
| Boolean..... | 17 |
| Пример SELECT запроса..... | 18 |
| Порядок выполнения..... | 18 |
| Написание SQL- кода..... | 19 |
| Комментарии..... | 19 |
| Разделение команд SQL..... | 19 |
| Регистр..... | 20 |
| Сортировка результатов. Order by..... | 21 |
| Подготовка тестовых данных..... | 21 |
| Сортировка по возрастанию. Asc..... | 21 |
| Сортировка по убыванию. Desc..... | 22 |
| Порядок сортировки по-умолчанию..... | 23 |
| Сортировка по порядковому номеру..... | 23 |
| Nulls last. Nulls first..... | 24 |
| Часть WHERE. Операторы сравнения..... | 26 |
| Операторы сравнения..... | 26 |
| Оператор "Меньше"(<)..... | 26 |
| Оператор "Больше"(>)..... | 27 |
| Оператор "Больше либо равно"(\geq)..... | 27 |
| Оператор "Меньше либо равно"(\leq)..... | 27 |
| Проверка нескольких условий. AND, OR..... | 29 |
| Проверка значения на NULL..... | 32 |
| IN, NOT IN..... | 34 |
| Вхождение в набор данных. IN..... | 34 |
| Отсутствие в наборе данных. NOT IN..... | 35 |
| Вхождение в диапазон. BETWEEN. NOT BETWEEN..... | 37 |
| Соединения таблиц..... | 39 |

| | |
|---------------------------------------------------------------|-----------|
| Подготовка данных..... | 39 |
| Join..... | 40 |
| Left join..... | 41 |
| Соединение таблиц без join..... | 43 |
| Древовидные структуры данных. Рекурсивные запросы..... | 44 |
| Реализация древовидных структур в РСУБД..... | 44 |
| Connect by..... | 45 |
| Псевдостолбец level..... | 45 |
| Псевдостолбец CONNECT_BY_ISLEAF..... | 46 |
| Сортировка в рекурсивных запросах..... | 47 |
| Нарушение древовидной структуры при выборке..... | 48 |
| Подзапросы в Oracle..... | 50 |
| Подготовка тестовых данных..... | 50 |
| Подзапросы в where- части запроса..... | 51 |
| Подзапросы в select-части..... | 52 |
| Подзапросы во FROM части..... | 52 |
| Коррелированные подзапросы..... | 53 |
| Подзапросы в IN, NOT IN..... | 54 |
| Exists. Наличие строк в подзапросе..... | 56 |
| Объединение запросов. UNION..... | 58 |
| Разница запросов. MINUS..... | 61 |
| Пересечение запросов..... | 65 |
| Работа с множествами. Общая информация..... | 66 |
| Следить за порядком колонок..... | 66 |
| Сортировка..... | 66 |
| Приоритет выполнения..... | 67 |
| Subquery factoring. WITH..... | 69 |
| Подготовка данных..... | 69 |
| Вариант без With..... | 70 |
| Вариант с WITH..... | 71 |
| Функции для работы со строками..... | 72 |
| UPPER, LOWER..... | 72 |
| Конкатенация строк..... | 72 |
| Поиск подстроки..... | 73 |
| Подобие строк. Like..... | 74 |
| Выражение ESCAPE в LIKE..... | 76 |
| Приведение к верхнему регистру. INITCAP..... | 77 |
| Замена подстроки. REPLACE..... | 78 |
| Удаление пробелов. TRIM..... | 78 |
| LPAD, RPAD..... | 79 |
| Функции для работы с NULL..... | 80 |
| Подготовка тестовых данных..... | 80 |
| Nvl..... | 80 |
| Nvl2..... | 81 |
| Coalesce..... | 81 |
| Условные функции..... | 83 |
| DECODE..... | 83 |
| CASE..... | 85 |
| Условные функции в WHERE части..... | 86 |
| Битовые операции..... | 89 |
| Тестовые данные..... | 89 |
| BIN_TO_NUM..... | 90 |

| | |
|----------------------------------------------------------|------------|
| BITAND. Побитовое "И" | 90 |
| Агрегирующие функции..... | 93 |
| Подготовка данных..... | 93 |
| Having..... | 95 |
| Distinct. Удаление дубликатов..... | 96 |
| Подготовка данных..... | 96 |
| Удаление дубликатов из одной колонки..... | 97 |
| DISTINCT учитывает все колонки в строке..... | 98 |
| NULL учитывается..... | 98 |
| DISTINCT с агрегатными функциями..... | 99 |
| DISTINCT и GROUP BY..... | 100 |
| Работа с датами в Oracle..... | 101 |
| Тип DATE..... | 101 |
| Приведение строки к дате..... | 101 |
| Функция SYSDATE..... | 101 |
| Приведение даты к строке..... | 101 |
| Trunc..... | 102 |
| ADD_MONTHS..... | 102 |
| Разница между датами..... | 102 |
| Months_between..... | 103 |
| Тип TIMESTAMP..... | 103 |
| SYSTIMESTAMP..... | 103 |
| EXTRACT..... | 104 |
| Приведение строки к timestamp..... | 104 |
| Аналитические функции..... | 106 |
| Когда агрегирующая функция становится аналитической..... | 108 |
| Подсчет результатов по группам. Partition by..... | 108 |
| Порядок вычисления. Order by..... | 109 |
| Диапазон работы аналитических функций..... | 110 |
| Строки и значения..... | 113 |
| Смещения при определении окна..... | 113 |
| Ограничения на ORDER BY..... | 117 |
| Оператор INSERT..... | 118 |
| Вставка с указанием колонок..... | 118 |
| Вставка без указания колонок..... | 118 |
| INSERT INTO ... SELECT..... | 119 |
| Изменение данных. UPDATE..... | 121 |
| Удаление данных. DELETE..... | 123 |
| Удаление данных из связанных таблиц..... | 123 |
| Слияние данных. MERGE..... | 124 |
| Подготовка данных..... | 124 |
| Использование MERGE..... | 125 |
| Использование DELETE в MERGE..... | 126 |
| Изменение структуры таблицы. ALTER TABLE..... | 129 |
| Подготовка данных..... | 129 |
| Добавление колонки в таблицу..... | 129 |
| Добавление нескольких колонок в таблицу..... | 130 |
| Удаление колонки из таблицы..... | 131 |
| Удаление нескольких колонок в таблице..... | 131 |
| Логическое удаление колонок..... | 131 |
| Переименование колонки..... | 132 |
| Изменение типа данных колонки..... | 132 |

| | |
|-----------------------------------------------------------------------|------------|
| Изменение атрибута NOT NULL в колонке..... | 133 |
| Переименование таблицы..... | 134 |
| Первичные ключи..... | 135 |
| Добавление первичного ключа в таблицу..... | 135 |
| Составные первичные ключи..... | 136 |
| Внешние ключи..... | 138 |
| Создание внешних ключей..... | 138 |
| Уникальные ключи..... | 141 |
| Создание уникальных ключей..... | 141 |
| Составные уникальные ключи..... | 142 |
| Представления(Views)..... | 144 |
| Что такое представления..... | 144 |
| Создание представлений..... | 144 |
| Символ * при создании представлений..... | 146 |
| Изменение данных представления..... | 147 |
| Представления с проверкой (WITH CHECK OPTION)..... | 148 |
| Изменение представлений из нескольких таблиц..... | 149 |
| Ограничения в изменяемых представлениях..... | 151 |
| Запрет изменения представления..... | 151 |
| Индексы..... | 152 |
| Что такое индексы..... | 152 |
| Создание индекса..... | 152 |
| Удаление индекса..... | 152 |
| Составные индексы..... | 152 |
| Index skip scan..... | 153 |
| Зачем использовать составные индексы..... | 153 |
| Уникальные индексы..... | 154 |
| Когда нужно создавать индекс..... | 154 |
| Виртуальные колонки..... | 155 |
| Создание виртуальных колонок..... | 155 |
| Добавление виртуальной колонки к уже существующей таблице..... | 155 |
| Когда использовать виртуальные колонки..... | 156 |
| Псевдостолбцы в Oracle..... | 157 |
| ROWNUM..... | 157 |
| Top-N query..... | 158 |
| ROWID..... | 159 |
| LEVEL..... | 159 |
| Транзакции в Oracle..... | 160 |
| Что такое транзакции..... | 160 |
| COMMIT. ROLLBACK..... | 160 |
| Почему до сих пор нигде в учебнике не использовались commit/rollback? | 161 |
| | 161 |
| Транзакции в многопользовательской среде..... | 161 |
| Итого..... | 162 |
| Где ещё брать информацию..... | 163 |
| Youtube..... | 163 |
| Блоги/сайты..... | 163 |
| Книги..... | 163 |

Введение в SQL

[SQL](#), или *Structure Query Language* (Структурированный язык запросов) является основным инструментом для взаимодействия с реляционными базами данных.

С помощью SQL можно:

- Получать данные из базы данных
- Сохранять данные в базу данных
- Производить манипуляции с объектами базы данных

Диалекты SQL

Реляционных систем управления базой данных (СУБД) существует достаточно много. И как правило, в каждой СУБД есть свои отличительные особенности в SQL, которые заключаются в наличии или отсутствии в нем определённых функций, различиях синтаксиса самого SQL, а также по функциональным возможностям этого языка.

В данном учебнике мы будем рассматривать [СУБД Oracle](#).

Особенности SQL

Пара слов о том, что необычного в SQL.

В отличие многих других языков программирования, например таких как Java, Pascal или JavaScript, программирование на которых заключается в том, чтобы описать, *как* нужно что-то сделать, в SQL описывается, *что* нужно сделать (т.е. какой результат мы хотим получить). SQL - ближайший к данным язык программирования. Он больше всего приближен к "чистым" данным системы. Под "чистыми" данными подразумевается то, что ниже тех абстракций, с которыми работает SQL, уже не будет.

Зачем изучать SQL

Как уже говорилось, SQL является основным средством общения с реляционными базами данных.

Когда какая-либо программа хочет получить, сохранить, или изменить данные в БД, то она это делает посредством SQL. Какой-нибудь список классов, с которыми работает объектно-ориентированный язык, должен получить данные, которые будут храниться в этих классах. Это все делается с помощью SQL.

Даже если в программе нигде явно не пишутся SQL-запросы, а используется с виду обычный программный код (например на языке Java), то это вовсе не значит, что в данном случае общение с БД происходит каким-то другим способом. Скорее всего, в программе используется специальная библиотека, которая превратит код на языке Java в соответствующий код на языке SQL и отправит его на выполнение БД. Подобных библиотек существует великое множество почти для всех популярных языков программирования.

DML, DDL

Команды языка SQL можно разбить на две группы - **DML** и **DDL**.

Кроме DML и DDL существуют еще команды *DCL* и *TCL*. На текущий момент они не рассматриваются в этом учебнике.

DML расшифровывается как *Data Manipulation Language* (Язык манипулирования данными). В него входят те команды SQL, которые могут изменять уже имеющиеся данные в БД. Под изменением следует понимать также добавление новой информации в БД и удаление уже существующей.

К командам DML относятся:

- SELECT
- INSERT
- UPDATE
- DELETE
- MERGE

Интересный момент - команда **SELECT** не изменяет данные, а только получает, но она все равно относится к категории DML.

DDL расшифровывается как *Data Definition Language* (Язык определения данных). В него входят те команды, которые отвечают за создание или изменение структуры данных или новых объектов в БД.

К DDL командам языка SQL относятся:

- CREATE
- RENAME
- ALTER
- DROP
- RENAME
- TRUNCATE
- COMMENT

Более подробно большая часть этих команд будет рассмотрена далее в этом учебнике.

Выполнение SQL. Облачные сервисы

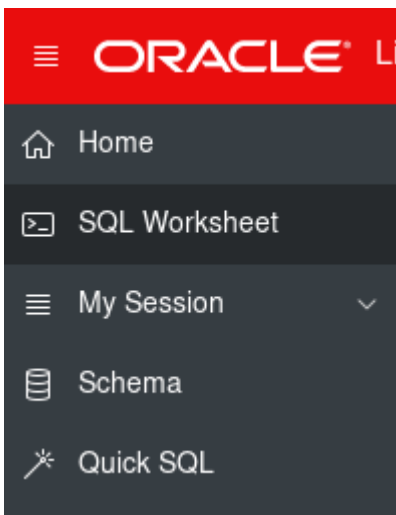
Для того, чтобы начать работу с БД (причём любой), она должна быть где-либо установлена, и к ней должен быть доступ на подключение и выполнение запросов.

LiveSQL

В этом учебнике для выполнения sql-запросов будет использоваться сервис [Live SQL](#). Он позволяет выполнять SQL в облаке, что непременно большой плюс - там гораздо быстрее зарегистрироваться, чем скачивать, устанавливать и настраивать себе БД Oracle.

Работать с livesql очень просто; опишем стандартные шаги, необходимые для запуска своих sql-запросов.

Входим под своей учёткой, после чего в левом боковом меню выбираем "SQL WorkSheet":

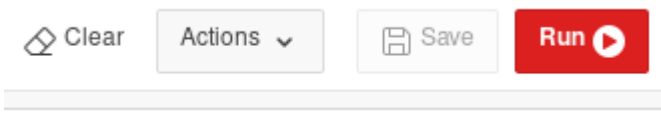


В открывшемся окне вводим наши SQL-запросы:

SQL Worksheet

```
1 create table test(  
2     id number(10) not null primary key,  
3     value varchar2(100)
```

Чтобы выполнить запрос, написанный в SQL Worksheet, нажимаем на кнопку "Run", которая находится сверху над полем для ввода текста запроса:



Вообще, работа с LiveSQL не должна вызывать вопросов, но на всякий случай вот видео с youtube(на английском) с подробным описанием работы в нем:

<https://youtu.be/4oxsxJQQC-s>.

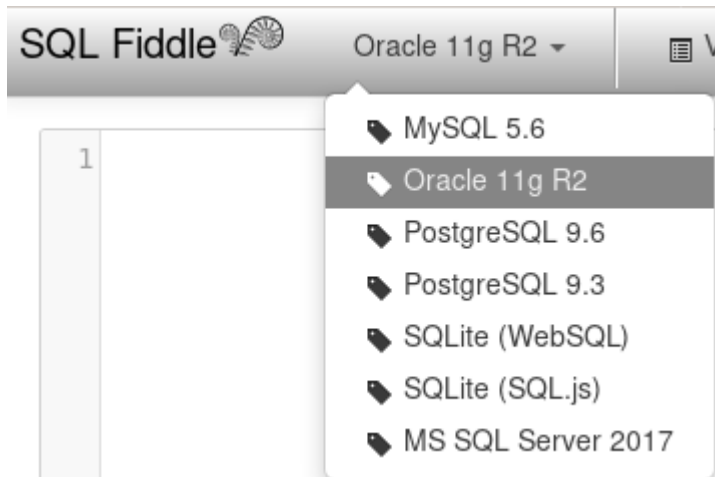
SQL Fiddle

[SQL Fiddle](#) - еще один популярный сервис для работы с SQL. Поддерживает разные базы данных. Для работы SQLFiddle даже не требует регистрации.

Далее будет описано, как работать с данным сервисом.


Сначала заходим на [SQL Fiddle](#).

Так как сервис поддерживает работу с несколькими БД, нужно выбрать ту, с которой будем работать - это Oracle:



Перед началом работы SQL Fiddle требует создания схемы. Это значит, что таблицы, с которыми нужно работать, должны быть созданы на этом этапе. Вводим текст ddl-скрипта (скрипта, который создаёт таблицы и др. объекты БД), после чего нажимаем на кнопку "Build Schema":

```
1 Create table test(  
2   msg varchar2(10) not null  
3 );
```

Build Schema  Edit Fullscreen 

После того, как схема будет построена, можно выполнять SQL-запросы. Они вводятся в правой панели(она называется "Query Panel"). Чтобы выполнить запрос, нажимаем на кнопку "Run Sql":

```
1 select *  
2 from dual
```

Run SQL ▶

Edit Fullscreen ↗

[;] ▼

Результаты выполнения запросов отображаются под панелями создания схемы и ввода sql:

DUMMY

X

Запуск примеров учебника

Запускать примеры из учебника можно в любой среде. Тем не менее, в силу того, что тема транзакций будет рассматриваться в самом конце, лучше всего (и удобнее) использовать сервис LiveSQL.

В дальнейшем, при изучении PL/SQL, придётся выбрать какую-нибудь IDE, но при изучении базового SQL это необязательно.

Инструменты для работы с БД Oracle

Далее будут приведены ссылки на полезные инструменты, которые могут пригодиться для работы с Oracle.

Средства разработки

- [Pl/sql developer](#) - известная среда разработки для Oracle, платная, есть пробный период.
- [SQL Developer](#) - бесплатная среда разработки от Oracle.
- [Toad for Oracle](#)
- [DBForge studio for Oracle](#)
- [JetBrains Datagrip](#) - отлично подходит, если необходимо работать одновременно с разными БД. Если рассматривать функционал, доступный с БД Oracle, то немного отстаёт от всех вышеперечисленных.
- [VSCode](#)(совместно с дополнением [Oracle Developer Tools for VS Code](#)) — неплохой вариант, особенно если не хочется использовать тяжёлый SQL Developer, а платить за остальные IDE не хочется.

Проектирование БД

- [SQL Data Modeler](#) - бесплатный, предоставляется корпорацией Oracle. Обладает обширным функционалом, заточенным именно на работу с БД Oracle.
- [ERWin DataModeler](#) - платный. Есть пробный период. Хорошо подходит для моделирования структуры данных без привязки к БД.

Таблицы

Данные в реляционных базах данных хранятся в таблицах. Таблицы - это ключевой объект, с которыми придётся работать в SQL.

Таблицы в БД совсем не отличаются от тех таблиц, с которыми все уже знакомы со школы - они состоят из колонок и строк.

Каждая колонка в таблице имеет своё имя и свой тип, т.е. тип данных, которые будут в ней содержаться. Помимо типа данных для колонки можно указать максимальный размер данных, которые могут содержаться в этой таблице.

Например, мы можем указать, что для колонки возраст тип данных - это целое число, и это число должно состоять максимум из 3-х цифр. Таким образом максимальное число, которое может содержаться в этой колонке = 999. А с помощью дополнительных конструкций можно задать и правила проверки корректности для значения в колонке,- например, мы можем указать, что для колонки возраст в таблице минимальное значение = 18.

Создание таблицы

```
create table hello(  
    text_to_hello varchar2(100)  
)
```

После выполнения данной sql-команды в базе данных будет создана таблица под названием hello. Эта таблица будет содержать всего одну колонку под названием text_to_hello. В этой колонке мы можем хранить только строковые значения(т.е. любой текст, который можно ввести с клавиатуры) длиной до 100 байт.

Обратите внимание на размер допустимого текста в колонке text_to_hello. 100 байт - это не одно и то же, что и 100 символов! Для того, чтобы сказать базе данных Oracle, что длина строки может быть 100 символов, нужно было определить столбец следующим образом:

```
text_to_hello varchar2(100 char)
```

Создание таблицы с несколькими полями

В таблице может много столбцов. Например, можно создать таблицу с тремя, пятью или даже 100 колонками. В версиях oracle с 8i по 11g максимальное количество колонок в одной таблице достигает 1000.

Для того, чтобы создать таблицу с несколькими колонками, нужно перечислить все колонки через запятую.

Например, создадим таблицу cars, в которой будем хранить марку автомобиля и страну производитель:

```
create table cars(  
    model varchar2(50 char),  
    country varchar2(70 char)  
)
```

Эта таблица может содержать, например, такие данные:

| model | country |
|--------|---------|
| toyota | japan |
| BA3 | Россия |
| Tesla | |
| | |

Следует обратить внимание на последние 2 строки в таблице cars - они не полные. Первая из них содержит данные только в колонке model, вторая - не содержит данных ни в одной из колонок. Эта таблица может даже состоять из миллиона строк, подобных последней - и каждая строка не будет содержать в себе абсолютно никаких данных.

Значения по умолчанию

При создании таблицы можно указать, какое значение будет принимать колонка по умолчанию:

```
create table cars(
    model varchar2(50 char),
    country varchar2(50 char),
    wheel_count number(2) default 4
)
```

В этом примере создаётся таблица cars, в которой помимо модели и страны производителя хранится ещё и количество колес, которое имеет автомобиль. И поле wheel_count по умолчанию будет принимать значение, равное 4.

Что значит по умолчанию? Это значит, что если при вставке данных в эту таблицу не указать значение для колонки wheel_count, то оно будет равно числу 4.

Понятие NULL. Not-null колонки

Ячейки в таблицах могут быть пустыми, т.е. не содержать значения. Для обозначения отсутствия значения в ячейке используется ключевое слово NULL. Null могут содержать ячейки с любым типом данных.

Рассмотрим таблицу cars из предыдущего примера. В каждой из трёх ее колонок может храниться Null (даже в колонке wheel_count, если указать значение Null явно при вставке).

Но представляют ли информационную ценность строки в таблице, где абсолютно нет значений? Конечно нет. Если рассматривать таблицу cars как источник информации об автомобилях, то нам хотелось бы получать хоть какую-то полезную информацию. Наиболее важной здесь будет колонка model - без нее информация о стране производителя и количестве колес будет бесполезной.

Для того, чтобы запретить Null-значения в колонке при создании таблицы, к описанию колонки добавляется not null:

```
create table cars(
    model varchar2(50 char) not null,
    country varchar2(50 char),
    wheel_count number(2) default 4
)
```