

# Содержание

Об авторе	13
Благодарности	15
<b>Введение</b>	17
Об этой книге	17
Кому адресована книга	19
Сопутствующий книге веб-сайт	19
Условные обозначения, принятые в книге	20
От издательства	21
<b>Урок 1. Общее представление о языке SQL</b>	23
Основы баз данных	23
Что такое база данных	24
Таблицы	25
Столбцы и типы данных	26
Строки	27
Первичные ключи	28
Что такое SQL	30
Опробуйте самостоятельно	31
Резюме	31
<b>Урок 2. Знакомство с Oracle и PL/SQL</b>	33
Что такое Oracle	33
Клиент-серверное программное обеспечение	33
PL/SQL	36
Клиентские инструментальные средства	36
Приступая к установке	37
Требуемое программное обеспечение	37
Приобретение программного обеспечения	39
Установка программного обеспечения	40
Резюме	42
<b>Урок 3. Работа с Oracle</b>	43
Создание рабочей среды	43
Создание специализированного экземпляра Oracle	44
Создание специального рабочего пространства	47
Установление соединения	49
Краткое введение в Oracle SQL Developer	51

---

Создание и заполнение примеров таблиц	53
Получение сценариев для создания и заполнения	54
Создание таблиц	55
Заполнение таблиц	56
Кое-что еще об Oracle SQL Developer	57
Резюме	58
<b>Урок 4. Извлечение данных</b>	<b>59</b>
Оператор <code>SELECT</code>	59
Извлечение отдельных столбцов	59
Извлечение нескольких столбцов	62
Извлечение всех столбцов	63
Извлечение отличающихся строк	65
Употребление полностью уточненных имен	67
Употребление комментариев	67
Резюме	70
<b>Урок 5. Сортировка извлекаемых данных</b>	<b>71</b>
Сортировка данных	71
Сортировка по нескольким столбцам	74
Указание направления сортировки	76
Резюме	79
<b>Урок 6. Фильтрация данных</b>	<b>81</b>
Применение предложения <code>WHERE</code>	81
Операции, употребляемые в предложении <code>WHERE</code>	83
Проверка на соответствие единственному значению	84
Проверка на несовпадение	87
Проверка на соответствие диапазону значений	88
Резюме	90
<b>Урок 7. Развитая фильтрация данных</b>	<b>91</b>
Сочетание предложений <code>WHERE</code>	91
Применение логической операции <code>AND</code>	91
Применение логической операции <code>OR</code>	93
Общее представление о порядке вычисления	94
Применение операции <code>IN</code>	96
Применение операции <code>NOT</code>	99
Резюме	100
<b>Урок 8. Фильтрация по шаблону</b>	<b>101</b>
Применение операции <code>LIKE</code>	101
Поиск по шаблону с метасимволом подстановки в виде знака процентов (%)	102

Поиск по шаблону с метасимволом подстановки в виде знака подчеркивания ( _ )	105
Рекомендации по применению метасимволов подстановки	107
Резюме	107
<b>Урок 9. Поиск по регулярным выражениям</b>	109
Общее представление о регулярных выражениях	109
Применение регулярных выражений в Oracle PL/SQL	110
Элементарное совпадение с символами	111
Совпадение по ИЛИ	114
Совпадение с одним из нескольких символов	115
Совпадение с диапазонами	118
Совпадение со специальными символами	119
Совпадение с классами символов	121
Совпадение с несколькими экземплярами	121
Метасимволы привязки	124
Резюме	126
<b>Урок 10. Создание вычисляемых полей</b>	127
Общее представление о вычисляемых полях	127
Сцепление полей	129
Применение псевдонимов	131
Выполнение математических расчетов	133
Резюме	135
<b>Урок 11. Применение функций манипулирования данными</b>	137
Общее представление о функциях	137
Применение функций	138
Функции манипулирования текстом	139
Функции манипулирования датой и временем	141
Функции манипулирования числами	146
Резюме	147
<b>Урок 12. Получение итоговых данных</b>	149
Применение агрегатных функций	149
Функция AVG ( )	150
Функция COUNT ( )	152
Функция MAX ( )	154
Функция MIN ( )	155
Функция SUM ( )	156
Агрегирование по отдельным значениям	158
Сочетание агрегатных функций	159
Резюме	160

<b>Урок 13. Группирование данных</b>	163
Общее представление о группировании данных	163
Создание групп	164
Фильтрация групп	166
Группирование и сортировка	169
Порядок указания предложений в операторе SELECT	172
Резюме	173
<b>Урок 14. Применение подзапросов</b>	175
Общее представление о подзапросах	175
Фильтрация по подзапросу	175
Применение подзапросов в качестве вычисляемых полей	180
Резюме	185
<b>Урок 15. Соединение таблиц</b>	187
Общее представление о соединениях	187
Общее представление о реляционных таблицах	187
Причины для применения соединений	189
Создание соединения	190
О значении предложения WHERE	192
Применение внутренних соединений	196
Соединение нескольких таблиц	197
Резюме	200
<b>Урок 16. Создание расширенных соединений</b>	201
Применение псевдонимов таблиц	201
Применение других типов соединений	202
Самосоединения	203
Естественные соединения	205
Внешние соединения	207
Применение соединений вместе с агрегатными функциями	209
Применение соединений и условий соединения	211
Резюме	212
<b>Урок 17. Объединение запросов</b>	213
Общее представление об объединенных запросах	213
Создание объединенных запросов	214
Применение операции UNION	214
Правила объединения	217
Дублирование строк и его исключение	218
Сортировка результатов обработки объединенных запросов	220
Резюме	221

<b>Урок 18. Ввод данных</b>	223
Общее представление о вводе данных	223
Ввод полных строк	224
Ввод извлекаемых данных	229
Резюме	232
<b>Урок 19. Обновление и удаление данных</b>	233
Обновление данных	233
Удаление данных	235
Рекомендации по обновлению и удалению данных	237
Резюме	238
<b>Урок 20. Создание таблиц и манипулирование ими</b>	239
Создание таблиц	239
Основы создания таблиц	240
Обработка пустых значений NULL	242
Указание значений по умолчанию	244
Обновление таблиц	245
Еще раз о первичных ключах	245
Определение внешних ключей	247
Удаление таблиц	248
Переименование таблиц	248
Резюме	249
<b>Урок 21. Применение представлений</b>	251
Общие сведения о представлениях	251
Причины для применения представлений	252
Правила и ограничения для представлений	253
Применение представлений	254
Упрощение сложных соединений с помощью представлений	254
Переформатирование извлекаемых данных с помощью представлений	256
Отсевивание ненужных данных с помощью представлений	258
Применение представлений вместе с вычисляемыми полями	259
Обновление представлений	261
Резюме	262
<b>Урок 22. Обращение с хранимыми процедурами</b>	263
Общее представление о хранимых процедурах	263
Причины для применения хранимых процедур	265
Применение хранимых процедур	266
Элементарный синтаксис хранимых процедур	267
Применение программных конструкций в хранимых процедурах	268

---

Построение логически развитых хранимых процедур	269
Удаление хранимых процедур	273
Резюме	273
<b>Урок 23. Применение курсоров</b>	275
Общее представление о курсорах	275
Обращение с курсорами	276
Создание курсоров	277
Открытие и закрытие курсоров	277
Извлечение данных из курсора	278
Обработка данных из курсора	281
Резюме	284
<b>Урок 24. Применение триггеров</b>	285
Общее представление о триггерах	285
Создание триггеров	286
Удаление триггеров	287
Применение триггеров	288
Триггеры типа INSERT	288
Триггеры типа DELETE	291
Триггеры типа UPDATE	293
Многособытийные триггеры	294
Еще о триггерах	295
Резюме	296
<b>Урок 25. Обработка транзакций</b>	297
Общее представление об обработке транзакций	297
Управление транзакциями	300
Применение оператора ROLLBACK	300
Применение оператора COMMIT	301
Применение точек сохранения	302
Резюме	303
<b>Урок 26. Управление безопасностью</b>	305
Общее представление об управлении доступом	305
Ведение учетных записей пользователей	307
Создание учетных записей пользователей	307
Удаление учетных записей пользователей	308
Установка прав доступа	308
Изменение паролей	311
Резюме	312
<b>Приложение А. Примеры таблиц</b>	313
Общее представление о примерах таблиц	313
Таблица vendors	314

Таблица products	315
Таблица customers	316
Таблица orders	316
Таблица orderitems	317
Таблица productnotes	318
<b>Приложение Б. Типы данных в Oracle PL/SQL</b>	319
Строковые типы данных	320
Числовые типы данных	322
Типы данных даты и времени	322
Двоичные типы данных	323
<b>Приложение В. Зарезервированные и ключевые слова в Oracle PL/SQL</b>	325
<b>Предметный указатель</b>	329

## УРОК 7

# Развитая фильтрация данных

На этом уроке демонстрируется сочетание предложений `WHERE` для создания эффективных и изощренных условий поиска, а также применение условных операций `NOT` и `IN`.

## Сочетание предложений `WHERE`

Все предложения `WHERE`, представленные на уроке 6, позволяют отфильтровать данные по единому критерию. Для более полного контроля над фильтрацией в Oracle допускается указывать несколько предложений `WHERE`. Этими предложениями можно пользоваться двумя способами: сочетая их с логической операцией `AND` или `OR`.

*Операция* — это специальное ключевое слово, применяемое для сочетания или изменения выражений, употребляемых в предложении `WHERE`. Называется также *логической операцией*.

## Применение логической операции `AND`

Чтобы отфильтровать извлекаемые данные по нескольким столбцам, достаточно воспользоваться логической операцией `AND`, присоединив условия фильтрации к предложению `WHERE`. В следующем примере кода показано, как это делается:

### Ввод ▼

```
SELECT prod_id, prod_price, prod_name
FROM products
WHERE vend_id = 1003 AND prod_price <= 10;
```



## Анализ ▼

В приведенном выше запросе SQL извлекаются наименования и цены на все товары, произведенные поставщиком с идентификатором **1003** по цене **10** или еще ниже. В операторе `SELECT` из этого запроса предложение `WHERE` состоит из двух условий, соединяемых логической операцией `AND`, которая предписывает СУБД вернуть только те строки, которые удовлетворяют обоим заданным условиям. Так, если товар, произведенный поставщиком с идентификатором **1003**, стоит больше **10**, то сведения о нем не извлекаются из таблицы. Аналогично из таблицы не извлекаются товары стоимостью меньше **10**, произведенные другим, а не указанным в запросе поставщиком. Результаты, выводимые по данному запросу SQL, выглядят следующим образом:

## Вывод ▼

```
+-----+-----+-----+
| prod_id | prod_price | prod_name      |
+-----+-----+-----+
| FB      |          10 | Bird seed      |
| FC      |           2.5 | Carrots        |
| SLING   |          4.49 | Sling          |
| TNT1    |           2.5 | TNT (1 stick)  |
| TNT2    |           10 | TNT (5 sticks) |
+-----+-----+-----+
```

**AND** — это ключевое слово, употребляемое в предложении `WHERE` для указания на то, что из таблицы будут извлечены только те строки, которые удовлетворяют всем заданным в запросе условиям.

В приведенном выше примере употреблена единственная логическая операция `AND`, а следовательно, она состоит из двух условий фильтрации. В предложении `WHERE` можно указать и больше условий фильтрации, разделив их ключевым словом `AND`.

ПРИМЕЧАНИЕ: **запросы SQL без предложения ORDER BY**

В приведенных выше примерах отсутствует предложение **ORDER BY**. Но при желании вы можете ввести его в свой запрос SQL.

## Применение логической операции OR

По своему действию логическая операция OR совершенно противоположна логической операции AND. В частности, логическая операция OR предписывает Oracle извлечь строки, удовлетворяющие любому из двух заданных условий.

Рассмотрим в качестве примера следующий запрос SQL с оператором SELECT:

### Ввод ▼

---

```
SELECT prod_name, prod_price  
FROM products  
WHERE vend_id = 1002 OR vend_id = 1003;
```

---

### Анализ ▼

---

По приведенному выше запросу SQL из таблицы извлекаются наименования и цены любых товаров, произведенных указанными поставщиками. Логическая операция OR предписывает СУБД обнаружить соответствие любому из двух заданных условий, а не обоим вместе. Если бы в данном запросе была употреблена логическая операция AND, то никаких данных не было бы возвращено, поскольку в этом случае было бы составлено такое предложение WHERE, которому нельзя было бы вообще удовлетворить. Результаты, выводимые по данному запросу SQL, выглядят следующим образом:

## Вывод ▼

---

prod_name	prod_price
Detonator	13
Bird seed	10
Carrots	2.5
Fuses	3.42
Oil can	8.99
Safe	50
Sling	4.49
TNT (1 stick)	2.5
TNT (5 sticks)	10

**OR** — это ключевое слово, употребляемое в предложении **WHERE** для указания на то, что из таблицы будут извлечены строки, совпадающие с любым из заданных в запросе условий.

## Общее представление о порядке вычисления

Предложения **WHERE** могут состоять из любого количества логических операций **AND** и **OR**. Благодаря сочетанию обоих этих операций можно организовать довольно изощренную фильтрацию.

Но сочетание логических операций **AND** и **OR** представляет собой интересную проблему. Продемонстрируем ее решение на конкретном примере. Допустим, требуется получить список всех товаров стоимостью **10** или выше, произведенных поставщиками с идентификаторами **1002** и **1003**. В следующем запросе SQL с оператором **SELECT** логические операции сочетаются в предложении **WHERE** для извлечения товаров по указанным выше условиям:

## Ввод ▼

---

```
SELECT prod_name, prod_price
FROM products
WHERE vend_id = 1002 OR vend_id = 1003 AND prod_price >= 10;
```

---

## Вывод ▼

---

```
+-----+-----+
| prod_name | prod_price |
+-----+-----+
| Detonator |          13 |
| Bird seed |          10 |
| Fuses     |         3.42 |
| Oil can   |         8.99 |
| Safe      |          50 |
| TNT (5 sticks) |        10 |
+-----+-----+
```

## Анализ ▼

---

Проанализируем приведенные выше результаты. В двух из всех возвращаемых строк содержатся цены ниже **10**, а следовательно, эти строки отфильтрованы не так, как требуется. Почему так произошло? Ответ на этот вопрос кроется в порядке вычисления. В языке SQL, как и в большинстве других языков, логические операции AND выполняются прежде логических операций OR. Если в запросе SQL обнаруживается предшествующее предложение WHERE, то из таблицы извлекаются товары, произведенные поставщиком с идентификатором **1002**, независимо от их цены, а также любые товары стоимостью **10** или выше, произведенные поставщиком с идентификатором **1003**. Иными словами, логическая операция AND имеет более высокий порядок вычисления, и поэтому в данном запросе сочетаются вместе неверные логические операции.

Решение описанной выше проблемы состоит в использовании круглых скобок для явного группирования связанных вместе логических операций. Рассмотрим в качестве примера следующий запрос SQL с оператором SELECT и проанализируем результаты его выполнения:

## Ввод ▼

---

```
SELECT prod_name, prod_price
FROM products
WHERE (vend_id = 1002 OR vend_id = 1003) AND prod_price >= 10;
```

---

## Вывод ▼

---

```
+-----+-----+
| prod_name      | prod_price |
+-----+-----+
| Detonator      |          13 |
| Bird seed      |          10 |
| Safe           |          50 |
| TNT (5 sticks) |          10 |
+-----+-----+
```

## Анализ ▼

---

Единственное отличие этого запроса SQL от предыдущего заключается в том, что первые два условия в предложении `WHERE` заключены в круглые скобки. А поскольку круглые скобки имеют более высокий порядок вычисления, чем логические операции `AND` или `OR`, то СУБД отфильтровывает сначала данные по условию `OR` в этих круглых скобках, а затем по данному запросу SQL извлекаются любые товары, произведенные поставщиком с идентификатором `1002` или `1003` стоимостью `10` или выше, что, собственно, и требуется.

### СОВЕТ: употребление круглых скобок в предложениях `WHERE`

Всякий раз, когда вы составляете предложение `WHERE`, в котором употребляются логические операции `AND` и `OR`, заключите группу логических операций в круглые скобки явным образом. Не полагайтесь только на соблюдаемый по умолчанию порядок вычисления, даже если это именно то, что вам требуется. Использование круглых скобок не таит в себе никаких недостатков, напротив — они позволяют избежать любой неоднозначности в коде.

## Применение операции `IN`

Круглые скобки находят еще одно, но совсем другое применение в предложениях `WHERE`. В частности, для указания ряда

условий, каждое из которых может быть удовлетворено, служит условная операция IN. Эта операция принимает разделяемый запятой список достоверных значений, причем все они заключены в круглые скобки, как демонстрируется в следующем примере:

## Ввод ▼

---

```
SELECT prod_name, prod_price
FROM products
WHERE vend_id IN (1002,1003)
ORDER BY prod_name;
```

---

## Вывод ▼

---

```
+-----+-----+
| prod_name      | prod_price |
+-----+-----+
| Bird seed      |          10 |
| Carrots        |           2.5 |
| Detonator      |           13 |
| Fuses          |           3.42 |
| Oil can        |           8.99 |
| Safe           |           50 |
| Sling          |           4.49 |
| TNT (1 stick)  |           2.5 |
| TNT (5 sticks) |           10 |
+-----+-----+
```

## Анализ ▼

---

По приведенному выше запросу SQL оператором SELECT извлекаются все товары, произведенные поставщиками с идентификаторами 1002 и 1003. Разделяемый запятой список достоверных значений следует после операции IN, и поэтому он должен быть заключен в круглые скобки.

Если вы считаете, что условная операция IN служит той же цели, что и логическая операция OR, то совершенно правы. В приведенном ниже примере по запросу SQL выполняются те же действия, что и в предыдущем примере.

**Ввод ▼**

```
SELECT prod_name, prod_price
FROM products
WHERE vend_id = 1002 OR vend_id = 1003
ORDER BY prod_name;
```

**Вывод ▼**

```
+-----+-----+
| prod_name | prod_price |
+-----+-----+
| Bird seed |         10 |
| Carrots   |         2.5 |
| Detonator |         13 |
| Fuses     |        3.42 |
| Oil can   |         8.99 |
| Safe      |         50 |
| Sling     |         4.49 |
| TNT (1 stick) |         2.5 |
| TNT (5 sticks) |         10 |
+-----+-----+
```

Зачем вообще употреблять операцию `IN`? Она дает следующие преимущества.

- ▶ Синтаксис операции `IN` более ясен и прост для работы с длинными списками достоверных значений.
- ▶ Порядок вычисления проще контролировать при использовании операции `IN`, поскольку в этом случае употребляется меньше операций.
- ▶ Условные операции `IN` почти всегда выполняются быстрее, чем списки логических операций `OR`, хотя обработка таких коротких списков, как в приведенных выше примерах запросов, практически никак не сказывается на производительности.
- ▶ Самое главное преимущество операции `IN` заключается в том, что она может содержать другой оператор `SELECT`, что дает возможность составлять весьма динамичные предложения `WHERE`. Подробнее об этом см. урок 14.

**IN** — это ключевое слово, употребляемое в предложении **WHERE** для указания списков значений, совпадающих при сравнении по логической операции **OR**.

## Применение операции NOT

Условная операция **NOT**, употребляемая в предложении **WHERE**, выполняет одну и только одну функцию. Она отрицает любое следующее далее условие.

**NOT** — это ключевое слово, употребляемое в предложении **WHERE** для отрицания заданного условия.

В приведенном ниже примере демонстрируется применение операции **NOT**. Чтобы перечислить товары, произведенные всеми поставщиками, кроме обозначенных идентификаторами **1002** и **1003**, можно сделать следующий запрос SQL:

### Ввод ▼

```
SELECT prod_name, prod_price
FROM products
WHERE vend_id NOT IN (1002,1003)
ORDER BY prod_name;
```

### Вывод ▼

```
+-----+-----+
| prod_name | prod_price |
+-----+-----+
| .5 ton anvil | 5.99 |
| 1 ton anvil | 9.99 |
| 2 ton anvil | 14.99 |
| JetPack 1000 | 35 |
| JetPack 2000 | 55 |
+-----+-----+
```

### Анализ ▼

В данном примере операция **NOT** служит для отрицания указанного после нее условия. Таким образом, вместо совпадения со



значением **1002** или **1003** в Oracle обнаруживается совпадение с любыми значениями в столбце `vend_id`, кроме **1002** или **1003**.

Так зачем же нужна условная операция `NOT`? Для составления простых предложений `WHERE` ее применение действительно не дает никаких преимуществ. Условная операция `NOT` оказывается более полезной для составления сложных предложений. Например, сочетая условные операции `NOT` и `IN`, можно упростить поиск всех строк, не соответствующих заданным критериям.

## Резюме

На этом уроке было продолжено рассмотрение материала предыдущего урока. В частности, вы научились сочетать предложения `WHERE` с логическими операциями `AND` и `OR`, а также управлять явным образом порядком вычисления и пользоваться условными операциями `IN` и `NOT`.