Краткий справочник по SQL

Оператор «SELECT»

Оглавление

Общий синтаксис SQL 2
Общий формат инструкции SQL 2
Предложение SELECT 2
Предложение FROM 3
Предложение WHERE 3
Сортировка результатов: предложение ORDER BY 3
Работа со сводными данными: предложения GROUP BY и HAVING 4
Задание полей, которые не используются в статистической функции: предложение GROUP BY 4
Ограничение статистических значений с помощью условий группировки: предложение HAVING 5
Объединение результатов запроса: оператор UNION 5
Использование псевдостолбцов (литералов) 6
Конкатенация 6
Псевдонимы столбцов (квалификтор AS) 7
Агрегатные функции 7
Операции сравнения 7
BETWEEN 8
IN 9
LIKE 9
IS NULL 10
Устранение дублирования (модификатор DISTINCT) 10
Соединение (JOIN) 11
Внутренние соединения 12
Самосоединения 15
Внешние соединения 15

Общий синтаксис SQL

Инструкция SQL состоит из нескольких частей, называемых предложениями. Каждое предложение в инструкции SQL имеет свое назначение. Некоторые предложения являются обязательными. В приведенной ниже таблице указаны предложения SQL, используемые чаще всего.

Предложение SQL	**Описание**	**Обязательное**
SELECT	Определяет поля, которые содержат нужные данные.	Да
FROM	Определяет таблицы, которые содержат поля, указанные в предложении SELECT.	Да
WHERE	Определяет условия отбора полей, которым должны соответствовать все записи, включаемые в результаты.	Нет
ORDER BY	Определяет порядок сортировки результатов.	Нет
GROUP BY	В инструкции SQL, которая содержит статистические функции, определяет поля, для которых в предложении SELECT не вычисляется сводное значение.	Только при наличии таких полей
HAVING	В инструкции SQL, которая содержит статистические функции, определяет условия, применяемые к полям, для которых в предложении SELECT вычисляется сводное значение.	Нет

Общий формат инструкции SQL

```
SELECT Адрес, Название
FROM Контакты
WHERE Город = «Ростов»
;
```

Примечания:

Access игнорирует разрывы строк в инструкции SQL. Несмотря на это, каждое предложение рекомендуется начинать с новой строки, чтобы инструкцию SQL было удобно читать как тому, кто ее написал, так и всем остальным.

Каждая инструкция SELECT заканчивается точкой с запятой (;). Точка с запятой может стоять как в конце последнего предложения, так и на отдельной строке в конце инструкции SQL.

Читать нужно так: Выбрать данные из полей Адрес и Название таблицы Контакты, где поле Город равно значению «Ростов»

Предложение SELECT

```
SELECT Адрес, Название
```

Это предложение SELECT. Оно содержит оператор (SELECT), за которым следуют два идентификатора (Адрес и Название).

Если идентификатор содержит пробелы или специальные знаки (например, зарезервированные слова (table, union)), он должен быть заключен в прямоугольные скобки. В Oracle – в двойные кавычки “ ”

В предложении SELECT не нужно указывать таблицы, в которых содержатся поля, и нельзя задать условия отбора, которым должны соответствовать данные, включаемые в результаты.

В инструкции SELECT предложение SELECT всегда стоит **перед** предложением FROM.

Предложение FROM

FROM Контакты

Это предложение FROM. Оно содержит оператор (FROM), за которым следует идентификатор (Контакты).

В предложении FROM **не указываются** поля для выборки.

Предложение WHERE

```
WHERE Город = «Ростов»
```

Это предложение WHERE. Оно содержит оператор (WHERE), за которым следует выражение (Город=«Ростов»).

Примечание: В отличие от предложений SELECT и FROM предложение WHERE является **необязательным** элементом инструкции SELECT.

Сортировка результатов: предложение ORDER BY

Во всех СУБД можно сортировать результаты запроса в таблице. Используя предложение ORDER BY, в запросе также можно указать способ сортировки результатов при выполнении запроса. Если используется предложение ORDER BY, оно должно находиться **в конце** инструкции SQL. ORDER BY – **не обязательное** предложение.

Предложение ORDER BY содержит список полей, для которых нужно выполнить сортировку, в том же порядке, в котором будут применена сортировка.

Предположим, например, что результаты сначала нужно отсортировать по убыванию значения поля «Организация», а затем, если присутствуют записи с одинаковым значением поля «Организация», отсортировать их по возрастанию

значения поля «Адрес». Предложение ORDER BY будет выглядеть следующим образом:

ORDER BY Организация DESC, Адрес

Примечание: По умолчанию в Access выполняется сортировка по возрастанию (от А до Я, от наименьшего к наибольшему, ключевое слово ASC – обычно не пишется, но подразумевается). Чтобы вместо этого выполнить сортировку значений по убыванию, необходимо указать ключевое слово DESC.

Порядок строк может задаваться одним из двух способов:

- именами столбцов
- номерами столбцов.

Работа со сводными данными: предложения GROUP BY и HAVING

Иногда возникает необходимость работы со сводными данными, такими как итоговые продажи за месяц или самые дорогие товары на складе. Для этого в предложении SELECT к полю следует применить агрегатная функция. Например, если в результате выполнения запроса нужно получить количество адресов каждой компании, предложение SELECT может выглядеть следующим образом:

SELECT COUNT(Адрес), Компания

Возможность употребления той или иной статистической функции зависит от типа данных в поле или используемого выражения.

Задание полей, которые не используются в статистической функции: предложение GROUP BY

При использовании статистических функций обычно необходимо создать предложение GROUP BY. В предложении GROUP BY указываются все поля, к которым не применяется статистическая функция. Если статистические функции применяются ко всем полям в запросе, предложение GROUP BY создавать не нужно.

Предложение GROUP BY должно следовать **сразу же** за предложением WHERE или FROM, если предложение WHERE отсутствует. В предложении GROUP BY поля указываются **в том же порядке**, что и в предложении SELECT.

Продолжим предыдущий пример. Пусть в предложении SELECT статистическая функция применяется только к полю Адрес, тогда предложение GROUP BY будет выглядеть следующим образом:

GROUP BY Компания

Ограничение статистических значений с помощью условий группировки: предложение HAVING

Если необходимо указать условия для ограничения результатов, но поле, к которому их требуется применить, используется в статистической функции, предложение WHERE использовать нельзя. Вместо него следует использовать предложение HAVING. Предложение HAVING работает так же, как и WHERE, но используется для статистических данных.

Предположим, например, что к первому полю в предложении SELECT применяется функция AVG (которая вычисляет среднее значение):

SELECT COUNT(Адрес), Компания

Чтобы ограничить результаты запроса на основе значения функции COUNT, к этому полю нельзя применить условие отбора в предложении WHERE. Вместо него условие следует поместить в предложение HAVING. Например, если нужно, чтобы запрос возвращал строки только в том случае, если у компании имеется несколько адресов, можно использовать следующее предложение HAVING:

HAVING COUNT(Адрес)>1

Примечание: Запрос может включать и предложение WHERE, и предложение HAVING, при этом условия отбора для полей, которые не используются в статистических функциях, указываются в предложении WHERE, а условия для полей, которые используются в статистических функциях, — в предложении HAVING.

Объединение результатов запроса: оператор UNION

Оператор UNION используется для одновременного просмотра всех данных, возвращаемых несколькими сходными запросами на выборку, в виде объединенного набора.

Оператор UNION позволяет объединить две инструкции SELECT в одну. Объединяемые инструкции SELECT должны иметь одинаковое число и порядок выходных полей с такими же или совместимыми типами данных. При выполнении запроса данные из каждого набора соответствующих полей объединяются в одно выходное поле, поэтому выходные данные запроса имеют столько же полей, сколько и каждая инструкция SELECT по отдельности.

Примечание: В запросах на объединение числовой и текстовый типы данных являются совместимыми.

Используя оператор UNION, можно указать, должны ли в результаты запроса включаться **повторяющиеся строки**, если таковые имеются. Для этого следует использовать ключевое слово **ALL**.

Запрос на объединение двух инструкций SELECT имеет следующий базовый синтаксис:

```
SELECT field_1
FROM table_1
UNION [ALL]
SELECT field_a
FROM table_a
;
```

Предположим, например, что имеется две таблицы, которые называются «Товары» и «Услуги». Обе таблицы содержат поля с названием товара или услуги, ценой и сведениями о гарантии, а также поле, в котором указывается эксклюзивность предлагаемого товара или услуги. Несмотря на то, что в таблицах «Продукты» и «Услуги» предусмотрены разные типы гарантий, основная информация одна и та же (предоставляется ли на отдельные продукты или услуги гарантия качества). Для объединения четырех полей из двух таблиц можно использовать следующий запрос на объединение:

```
SELECT name, price, warranty_available, exclusive_offer
FROM Products
UNION ALL
SELECT name, price, guarantee_available, exclusive_offer
FROM Services
;
```

Использование псевдостолбцов (литералов)

Для придания большей наглядности получаемому результату можно использовать литералы. Литералы - это строковые константы, которые применяются наряду с наименованиями столбцов и, таким образом, выступают в роли "псевдостолбцов". Строка символов, представляющая собой литерал, должна быть заключена в одинарные или двойные скобки.

```
SELECT first_name, "получает", salary,
    "долларов в год"
FROM employee

FIRST_NAME   SALARY
Robert получает 105900.00 долларов в год
Bruce получает 97500.00 долларов в год
```

Конкатенация

Имеется возможность соединять два или более столбца, имеющие строковый тип, друг с другом, а также соединять их с литералами. Для этого используется операция конкатенации (||).

```
SELECT "сотрудник " || first_name || " " ||
```

```
        last_name
FROM employee

получить список всех сотрудников
=============================================
сотрудник    Robert Nelson
сотрудник Bruce Young
сотрудник Kim Lambert
```

Псевдонимы столбцов (квалификтор AS)

Для придания наглядности получаемым результатам наряду с литералами в списке выбираемых элементов можно использовать квалификатор AS. Данный квалификатор заменяет в результирующей таблице существующее название столбца на заданное. Это наиболее эффективный и простой способ создания заголовков (к сожалению, InterBase, как уже отмечалось, не поддерживает использование русских букв в наименовании столбцов).

```
SELECT count(*) AS number
FROM employee

подсчитать количество служащих
    NUMBER
===========
        42
```

Агрегатные функции

К агрегирующим функциям относятся функции вычисления суммы (SUM), максимального (MAX) и минимального (MIN) значений столбцов, арифметического среднего (AVG), а также количества строк, удовлетворяющих заданному условию (COUNT).

Операции сравнения

Рассмотрим операции сравнения. Реляционные операторы могут использоваться с различными элементами. При этом важно соблюдать следующее правило: *элементы должны иметь сравнимые типы*. Если в базе данных определены домены, то сравниваемые элементы должны относиться к одному домену.

Что же может быть элементом сравнения? Элементом сравнения может выступать:

- значение поля
- литерал
- арифметическое выражение

- агрегирующая функция
- другая встроенная функция
- значение (значения), возвращаемые подзапросом.

Операторы сравнения

- = равно
- <> не равно
- != не равно
- > больше
- < меньше
- >= больше или равно
- <= меньше или равно

```
SELECT first_name, last_name, dept_no
FROM employee
WHERE job_code = "Admin"
```

получить список сотрудников (и номера их отделов), занимающих должность администраторов

```
FIRST_NAME LAST_NAME  DEPT_NO
Terri Leev 000
Ann Bennet 120
```

BETWEEN

Предикат BETWEEN задает диапазон значений, для которого выражение принимает значение true. Разрешено также использовать конструкцию NOT BETWEEN.

```
SELECT first_name, last_name, salary
FROM employee
WHERE salary BETWEEN 20000 AND 30000
```

получить список сотрудников, годовая зарплата которых больше 20000 и меньше 30000

```
FIRST_NAME LAST_NAME SALARY
Ann Bennet 22935.00
Kelly Brown 27000.00
```

Тот же запрос с использованием операторов сравнения будет выглядеть следующим образом:

```
SELECT first_name, last_name, salary
```

```
FROM employee
WHERE salary >= 20000
  AND salary <= 30000
```

IN

Предикат IN проверяет, входит ли заданное значение, предшествующее ключевому слову "IN" (например, значение столбца или функция от него) в указанный в скобках список. Если заданное проверяемое значение равно какому-либо элементу в списке, то предикат принимает значение true. Разрешено также использовать конструкцию NOT IN.

```
SELECT first_name, last_name, job_code
FROM employee
WHERE job_code IN ("VP", "Admin", "Finan")
```

получить список сотрудников, занимающих должности "вице-президент", "администратор", "финансовый директор"

```
FIRST_NAME LAST_NAME JOB_CODE
Robert Nelson VP
Terri Lee Admin
Stewart Hall Finan
```

LIKE

Предикат LIKE используется только с символьными данными. Он проверяет, соответствует ли данное символьное значение строке с указанной маской. В качестве маски используются все разрешенные символы (с учетом верхнего и нижнего регистров), а также специальные символы:
•% - замещает любое количество символов (в том числе и 0),
•_ - замещает только один символ.

Разрешено также использовать конструкцию NOT LIKE.

```
SELECT first_name, last_name
FROM employee
WHERE last_name LIKE "F%"
```

получить список сотрудников, фамилии которых начинаются с буквы "F"

```
FIRST_NAME LAST_NAME
Phil            Forest
Pete            Fisher
Roberto         Ferrari
```

```
SELECT first_name, last_name
FROM employee
```

```
WHERE first_name LIKE "%er"
```

получить список сотрудников, имена которых заканчиваются буквами "er"

```
FIRST_NAME LAST_NAME
De Souza        RogerRogerWalter
```

IS NULL

В SQL-запросах NULL означает, что значение столбца неизвестно. Поисковые условия, в которых значение столбца сравнивается с NULL, всегда принимают значение unknown (и, соответственно, приводят к ошибке), в противоположность true или false, т.е.

```
WHERE dept_no = NULL
```
или даже WHERE NULL = NULL.

Предикат IS NULL принимает значение true только тогда, когда выражение слева от ключевых слов "IS NULL" имеет значение null (пусто, не определено). Разрешено также использовать конструкцию IS NOT NULL, которая означает "не пусто", "имеет какое-либо значение".

```
SELECT department, mngr_no
FROM department
WHERE mngr_no IS NULL
```

получить список отделов, в которых еще не назначены начальники

```
DEPARTMENT MNGR_NO
Marketing               <null>
Software Products Div. <null>
```

Устранение дублирования (модификатор DISTINCT)

Дублированными являются такие строки в результирующей таблице, в которых идентичен каждый столбец.

Иногда (в зависимости от задачи) бывает необходимо устранить все повторы строк из результирующего набора. Этой цели служит модификатор DISTINCT. Данный модификатор может быть указан только один раз в списке выбираемых элементов и действует на весь список.

```
SELECT job_code
FROM employee
```

получить список должностей сотрудников

```
JOB_CODE
========
VP
Eng
Eng
Mktg
Mngr
SRep
```

Данный пример некорректно решает задачу "получения" списка должностей сотрудников предприятия, так как в нем имеются многочисленные повторы, затрудняющие восприятие информации. Тот же запрос, включающий модификатор DISTINCT, устраняющий дублирование, дает верный результат.

```
SELECT DISTINCT job_code
FROM employee
```

получить список должностей сотрудников

```
JOB_CODE
========
Admin
CEO
CFO
Dir
Doc
```

Модификатор DISTINCT действует на всю строку сразу.

```
SELECT DISTINCT first_name, last_name
FROM employee
WHERE first_name = "Roger"
```

получить список служащих, имена которых - Roger

```
FIRST_NAME LAST_NAME
Roger  De Souza
Roger  Reeves
```

Соединение (JOIN)

Операция соединения используется в языке SQL для вывода связанной информации, хранящейся в нескольких таблицах, в одном запросе. В этом проявляется одна из наиболее важных особенностей запросов SQL - способность определять связи между многочисленными таблицами и выводить информацию из них в рамках этих связей. Именно эта операция придает гибкость и легкость языку SQL.

После изучения этого раздела мы будем способны:

- соединять данные из нескольких таблиц в единую результирующую таблицу;
- задавать имена столбцов двумя способами;
- записывать внешние соединения;
- создавать соединения таблицы с собой.

Операции соединения подразделяются на два вида - внутренние и внешние. Оба вида соединений задаются в предложении WHERE запроса SELECT с помощью специального *условия соединения*. Внешние соединения (о которых мы поговорим позднее) поддерживаются стандартом ANSI-92 и содержат зарезервированное слово "JOIN", в то время как внутренние соединения (или просто соединения) могут задаваться как без использования такого слова (в стандарте ANSI-89), так и с использованием слова "JOIN" (в стандарте ANSI-92).

Связывание производится, как правило, по первичному ключу одной таблицы и внешнему ключу другой таблицы - для каждой пары таблиц. При этом очень важно учитывать все поля внешнего ключа, иначе результат будет искажен. Соединяемые поля могут (но не обязаны!) присутствовать в списке выбираемых элементов. Предложение WHERE может содержать множественные условия соединений. Условие соединения может также комбинироваться с другими предикатами в предложении WHERE.

Внутренние соединения

Внутреннее соединение возвращает только те строки, для которых условие соединения принимает значение true.

```
SELECT first_name, last_name, department
FROM employee, department
WHERE job_code = "VP"
```

получить список сотрудников, состоящих в должности "вице-президент", а также названия их отделов

FIRST_NAME	LAST_NAME	DEPARTMENT
Robert	Nelson	Corporate Headquarters
Mary S.	MacDonald	Corporate Headquarters
Robert	Nelson	Sales and Marketing
Mary S.	MacDonald	Sales and Marketing
Robert	Nelson	Engineering
Mary S.	MacDonald	Engineering
Robert	Nelson	Finance
Mary S.	MacDonald	Finance

Этот запрос ("без соединения") возвращает неверный результат, так как имеющиеся между таблицами связи не задействованы. Отсюда и появляется дублирование

информации в результирующей таблице. Правильный результат дает запрос с использованием операции соединения:

```
SELECT first_name, last_name, department
```

```
SELECT first_name, last_name, department
FROM employee, department
WHERE job_code = "VP"
  AND employee.dept_no = department.dept_no
```

имена таблиц

получить список сотрудников, состоящих в должности "вице-президент", а также названия их отделов

FIRST_NAME	LAST_NAME	DEPARTMENT
Robert	Nelson	Engineering
Mary S.	MacDonald	Sales and Marketing

В вышеприведенном запросе использовался способ непосредственного указания таблиц с помощью их имен. Возможен (а иногда и просто необходим) также способ указания таблиц с помощью алиасов (псевдонимов). При этом алиасы определяются в предложении FROM запроса SELECT и представляют собой любой допустимый идентификатор, написание которого подчиняется таким же правилам, что и написание имен таблиц. Потребность в алиасах таблиц возникает тогда, когда названия столбцов, используемых в условиях соединения двух (или более) таблиц, совпадают, а названия таблиц слишком длинны...

Замечание 1: в одном запросе нельзя смешивать использование написания имен таблиц и их алиасов.

Замечание 2: алиасы таблиц могут совпадать с их именами.

```
SELECT first_name, last_name, department
FROM employee e, department d
WHERE job_code = "VP"
  AND e.dept_no = d.dept_no
```

алиасы таблиц

получить список сотрудников, состоящих в должности "вице-президент", а также названия их отделов

FIRST_NAME	LAST_NAME	DEPARTMENT
Robert	Nelson	Engineering
Mary S.	MacDonald	Sales and Marketing

А вот пример запроса, соединяющего сразу три таблицы:

```
SELECT first_name, last_name, job_title,
       department
FROM employee e, department d, job j
WHERE d.mngr_no = e.emp_no
  AND e.job_code = j.job_code
  AND e.job_grade = j.job_grade
  AND e.job_country = j.job_country
```

получить список сотрудников с названиями их должностей и названиями отделов

FIRST_NAME	LAST_NAME	JOB_TITLE	DEPARTMENT
Robert	Nelson	Vice President	Engineering
Phil	Forest	Manager	Quality Assurance
K. J.	Weston	Sales Representative	Field Office: East Coast
Katherine	Young	Manager	Customer Support
Chris	Papadopoulos	Manager	Research and Development
Janet	Baldwin	Sales Co-ordinator	Pacific Rim Headquarters
Roger	Reeves	Sales Co-ordinator	European Headquarters
Walter	Steadman	Chief Financial Officer	Finance

В данном примере последние три условия необходимы в силу того, что первичный ключ в таблице JOB состоит из трех полей.

Мы рассмотрели внутренние соединения с использованием стандарта ANSI-89. Теперь опишем новый (ANSI-92) стандарт:

- условия соединения записываются в предложении FROM, в котором слева и справа от зарезервированного слова "JOIN" указываются соединяемые таблицы;
- условия поиска, основанные на правой таблице, помещаются в предложение ON;
- условия поиска, основанные на левой таблице, помещаются в предложение WHERE.

```
SELECT first_name, last_name, department
FROM employee e JOIN department d
 ON e.dept_no = d.dept_no
AND department = "Customer Support"
WHERE last_name starting with "P"
```

получить список служащих (а заодно и название отдела), являющихся сотрудниками отдела "Customer Support", фамилии которых начинаются с буквы "P"

FIRST_NAME	LAST_NAME	DEPARTMENT
Leslie	Phong	Customer Support
Bill	Parker	Customer Support

Самосоединения

В некоторых задачах необходимо получить информацию, выбранную особым образом только из одной таблицы. Для этого используются так называемые самосоединения, или рефлексивные соединения. Это не отдельный вид соединения, а просто соединение таблицы с собой с помощью алиасов. Самосоединения полезны в случаях, когда нужно получить пары аналогичных элементов из одной и той же таблицы.

```
SELECT one.last_name, two.last_name,
       one.hire_date
FROM employee one, employee two
WHERE one.hire_date = two.hire_date
  AND one.emp_no < two.emp_no
```

получить пары фамилий сотрудников, которые приняты на работу в один и тот же день

LAST_NAME	LAST_NAME	HIRE_DATE
Nelson	Young	28-DEC-1988
Reeves	Stansbury	25-APR-1991
Bishop	MacDonald	1-JUN-1992
Brown	Ichida	4-FEB-1993

```
SELECT d1.department, d2.department, d1.budget
FROM department d1, department d2
WHERE d1.budget = d2.budget
  AND d1.dept_no < d2.dept_no
```

получить список пар отделов с одинаковыми годовыми бюджетами

DEPARTMENT	DEPARTMENT	BUDGET
Software Development	Finance	400000.00
Field Office: East Coast	Field Office: Canada	500000.00
Field Office: Japan	Field Office: East Coast	500000.00
Field Office: Japan	Field Office: Canada	500000.00
Field Office: Japan	Field Office: Switzerland	500000.00
Field Office: Singapore	Quality Assurance	300000.00
Field Office: Switzerland	Field Office: East Coast	500000.00

Внешние соединения

Напомним, что внутреннее соединение возвращает только те строки, для которых условие соединения принимает значение true. Иногда требуется включить в результирующий набор большее количество строк.

Вспомним, запрос вида

```
SELECT first_name, last_name, department
FROM employee e, department d
WHERE e.dept_no = d.dept_no
```

возвращает только те строки, для которых условие соединения (e.dept_no = d.dept_no) принимает значение true.

Внешнее соединение возвращает **все** строки из одной таблицы и только те строки из другой таблицы, для которых условие соединения принимает значение true. Строки второй таблицы, не удовлетворяющие условию соединения (т.е. имеющие значение false), получают значение null в результирующем наборе.

Существует два вида внешнего соединения: **LEFT JOIN** и **RIGHT JOIN**.

В левом соединении (LEFT JOIN) запрос возвращает все строки из левой таблицы (т.е. таблицы, стоящей *слева* от зарезервированного словосочетания "LEFT JOIN") и только те из правой таблицы, которые удовлетворяют условию соединения. Если же в правой таблице не найдется строк, удовлетворяющих заданному условию, то в результате они замещаются значениями null.

Для правого соединения - все наоборот.

```
SELECT first_name, last_name, department
FROM employee e LEFT JOIN department d
  ON e.dept_no = d.dept_no
```

получить список сотрудников и название их отделов, включая сотрудников, еще не назначенных ни в какой отдел

FIRST_NAME	LAST_NAME	DEPARTMENT
Robert	Nelson	Engineering
Bruce	Young	Software Development

В результирующий набор входит и отдел "Software Products Div." (а также отдел "Field Office: Singapore", не представленный здесь), в котором еще нет ни одного сотрудника.