Оглавление

Оператор SELECT

Общая форма оператора SELECT

```
-- Syntax for SQL Server

<SELECT statement> ::=
    [ WITH { [ XMLNAMESPACES ,] [ <common_table_expression> [,...n] ] } ]
    <query_expression>
    [ ORDER BY { order_by_expression | column_position [ ASC | DESC ] }
  [ ,...n ] ]
    [ <FOR Clause>]
    [ OPTION ( <query_hint> [ ,...n ] ) ]
<query_expression> ::=
    { <query_specification> | ( <query_expression> ) }
    [  { UNION [ ALL ] | EXCEPT | INTERSECT }
        <query_specification> | ( <query_expression> ) [...n ] ]
<query_specification> ::=
SELECT [ ALL | DISTINCT ]
    [TOP ( expression ) [PERCENT] [ WITH TIES ] ]
    < select_list >
    [ INTO new_table ]
    [ FROM { <table_source> } [ ,...n ] ]
    [ WHERE <search_condition> ]
    [ <GROUP BY> ]
    [ HAVING < search_condition > ]
```

SELECT [**DISTINCT**] <список полей> | <список выражений>
FROM <список таблиц> | <табличное выражение>
[**WHERE** <условие выборки>]
[**GROUP BY** <список полей>]
[**HAVING** <условие>]
[[**UNION** <выражение с оператором **SELECT**>]]
[**ORDER BY** <список полей>]

Логический порядок обработки инструкции SELECT

Следующие список демонстрирует логический порядок обработки или порядок привязки инструкции SELECT. Этот порядок определяет, когда объекты, определенные в одном шаге, становятся доступными для предложений в последующих шагах.

1. FROM
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. WITH CUBE или WITH ROLLUP
7. HAVING
8. SELECT
9. DISTINCT
10. ORDER BY
11. В начало

Простой оператор select (выборка)

Пример:

```
select * from STUDENTS;
```

Пример со списком полей (проекция)

```
select ID, SURNAME, NAME  from  STUDENTS;
```

Повсеместное использование * является плохой практикой, так как это может привести к передаче лишних (больших) объемов данных и увеличению сетевого траффика. Кроме того, изменение порядка следования полей в исходной таблице может привести к ошибкам в прикладной программе.

Комментарии в SQL

Комментарии в коде SQL это текстовые строки, которые не обрабатывается сервером. Комментарии бывают строковые и блочные.

Строковые комментарии - начинаются с двух символов "-" и заканчиваются в конце строки.

```
select
       *      -- избегайте использования *
from STUDENTS;
```

Блочные комментарии – это одна или несколько строк кода заключенные между последовательностью символов /* и */

```
/*
       Пример использования блочного комментария.
       Блочный комментарий
       может располагаться в
       нескольких строках
*/
select
       * -- избегайте использования *
from STUDENTS;
```

Выражения в операторе select

Оператор select может извлекать не только значения из колонок таблицы, но и другие, иногда более сложные выражения. Такие выражения могут быть константами, вычисляемыми выражениями или некоторыми функциями над одним или несколькими полями таблицы БД. Пример использования выражения:

```
select
       id
       ,surname
       ,stipend + id
       ,'Это строковый литерал'
from STUDENTS;
```

```
select
	id
	, surname
	, name
	, birthday
	, DATEDIFF(YY, birthday, getdate()) as age
from STUDENTS;
```

Для MySQL

```
select
	id
	, surname
	, name
	, birthday
	, TIMESTAMPDIFF(YEAR, birthday, CURDATE()) as age
from  STUDENTS;
```

DISTINCT как особый случай фильтрации

Оператор **DISTINCT** используется для указания того, что следует работать только с уникальными значениями столбца или столбцов. Например, показать какие стипендии есть в нашей базе данных:

```
select distinct STIPEND from STUDENTS;
```

Если мы хотим посмотреть уникальные стипендии по курсам, то в выборку следует добавить поле курса. Тогда мы получим только уникальные пары <стипендия, курс>:

```
select distinct COURSE, STIPEND from STUDENTS;
```

Псевдонимы (alias)

Alias (от англ.alias-псевдоним) — псевдонимы могут быть использованы для переименования (в рамках запроса) таблиц и колонок. В отличие от настоящих имен, псевдонимы могут не соответствовать ограничениям на имена объектов базы данных и могут содержать до 255 знаков (включая пробелы, цифры и специальные символы). С помощью псевдонимов возможно:

- задавать таблицам или столбцам другие имена:
  - **COLUMN ALIASES** используются для упрощения чтения столбцов в вашем результирующем наборе.
  - **TABLE ALIASES** используются для сокращения вашего SQL-кода, чтобы упростить его чтение или когда вы выполняете самостоятельное соединение (то есть: перечисление одной и той же таблицы более одного раза);
- дать имя полю, у которого до этого вообще не было имени. Пример

  ```
  Select 1 As Num
  ```

- использовать одну и ту же таблицу в операторе **SELECT** много раз;
- при указании псевдонима (alias) присутствие AS **необязательно;**
- указывать точную принадлежность поля той или иной таблице, используемой в запросе.

```
select * from STUDENTS, LECTURERS
```

```
select
	s.SURNAME
	, l.SURNAME
from STUDENTS as s, LECTURERS as l;
```

```
select distinct
	s.SURNAME as "Студент"
	, l.SURNAME as "Переподаватель"
from STUDENTS as s, LECTURERS as l;
```

Выборка нескольких записей

```
select top 2 * from STUDENTS;
```

MySQL

```
select
	*
from students
limit 0, 2;
```

Сортировка ORDER BY

Оператор сортировки ORDER BY позволяет упорядочить результат выборки по одному или нескольким столбцам, а также позволяет указать «направление» сортировки: «от меньшего к большему» (ASC принято по умолчанию), или «от большего к меньшему» (DESC). В выражении ORDER BY может использоваться несколько полей в разных порядках

Примеры:
Список студентов в алфавитном порядке

```
select SURNAME, NAME from STUDENTS order by SURNAME;
select SURNAME, NAME, GENDER from STUDENTS order by GENDER desc, SURNAME;
```

В оператор ORDER BY могут входить поля, которые присутствуют в таблице, но не указаны явно в операторе SELECT

```
select SURNAME, NAME from STUDENTS order by GENDER desc, SURNAME;
```

Сортировать можно по выражению. Пример:

```
select
	SURNAME
	,NAME
	,BIRTHDAY
	,COURSE
	,DATEDIFF(YY, birthday, getdate()) as age
from STUDENTS
order by DATEDIFF(YY, birthday, getdate());
```

Для MySQL

```
select
	SURNAME
	,NAME
	,BIRTHDAY
	,COURSE
	,TIMESTAMPDIFF(YEAR, birthday, CURDATE()) as age
from STUDENTS
order by TIMESTAMPDIFF(YEAR, birthday, CURDATE());
```

Выражение ORDER BY можно упростить использовав alias и информацию о порядке вычисления оператора select.

```
select
       SURNAME
       ,NAME
       ,BIRTHDAY
       ,COURSE
       ,DATEDIFF(YY, birthday, getdate()) as age
from STUDENTS
order by age --DATEDIFF(YY, birthday, getdate());
```

Фильтрация данных WHERE

При помощи оператора WHERE, можно указать условие поиска, а именно какие записи включаются в конечный результат (в выборку), а какие нет. Решение о включении записи в результирующую выборку принимается на основании того, является ли результат вычисления некоего оператора (предиката) **для строки** истиной (TRUE) или ложью (FALSE).

Предикаты это выражения, принимающие истинностные значения. Они могут представлять собой как одно простое булево выражение, так и любую комбинацию из неограниченного количества булевых выражений, построенную с помощью булевых операторов AND, OR или NOT. Кроме того, в этих комбинациях может использоваться SQL-оператор IS, а также круглые скобки для конкретизации порядка выполнения операций.

Булева алгебра

Булева алгебра (алгебра высказываний) — раздел математической логики, в котором изучаются логические операции над высказываниями. Предполагается, что высказывания могут быть только истинными или ложными, то есть используется так называемая бинарная или двоичная логика, в отличие от, например, троичной логики.

Таким образом, в булевой алгебре используются только два значения «Истина» True (T) и «Ложь» False (F).
В булевой алгебре определены три операции.

- ∧ Конъюнкция (логическое умножение, «И»), бинарная операция
- ∨ Дизъюнкция (логическое сложение, «ИЛИ»), бинарная операция
- ¬ Отрицание (логическое отрицание, «НЕ»), унарная операция

Эти операции удовлетворяют следующим аксиоматическим правилам:

Таблицы истинности:

∧	F	T
F	F	F
T	F	T

∨	F	T
F	F	T
T	T	T

- $a \wedge a = a,\ a \vee a = a$
- $\neg(\neg a) = a$
- $a \wedge b = b \wedge a,\quad a \vee b = b \vee a$ - коммутативность
- $a \wedge (b \wedge c) = (a \wedge b) \wedge c,\quad a \vee (b \vee c) = (a \vee b) \vee c$ - ассоциативность

- $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ - дистрибутивность
- $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

Правила Де Моргана

- $\neg(a \wedge b) = \neg a \vee \neg b$
- $\neg(a \vee b) = \neg a \wedge \neg b$

Приоритет выполнения логических операций

Приоритет (порядок выполнения) логических операций следующий:

1. Вычисляются значения выражений внутри скобок;
2. Выполняются отрицания над отдельными переменными (отрицание, НЕ);
3. Вычисляются конъюнкции (И, И-НЕ);
4. Вычисляются дизъюнкции (ИЛИ, ИЛИ-НЕ);

Если сложное выражение содержит несколько операторов, порядок выполнения этих операторов определяется их приоритетом. Порядок исполнения может существенно повлиять на результирующее значение.

Если два оператора в выражении имеют один и тот же уровень старшинства, они выполняются в порядке слева направо по мере их появления в выражении.

Логические операторы SQL

Операторы сравнения

Сравнение – используется для сравнения двух значений одного типа. Применяются следующие операторы:

- **= -** равенство двух значений;
- <> - не равно;
- > - строго больше;
- < - строго меньше;
- >= - больше равно;
- <= - меньше равно;

Специальные операторы

- **Диапазон** – BETWEEN проверка того, что значение попадает в определенный диапазон значений;
- **Принадлежность** – IN позволяет проверить совпадает ли тестируемое значение с каким-либо элементом заданного списка;
- **Поиск по шаблону** – LIKE позволяет определить совпадает ли тестируемая строка с некоторым шаблоном строки;
- **IS NULL –** позволяет определить является ли тестируемое значение значением NULL
- **Существование** – EXISTS определяет, существует ли определенное значение. Обычно используется в подзапросах.

Пример:

```
select
       ID
       , SURNAME
       , NAME
       , STIPEND
from STUDENTS where STIPEND >= 500;
```

Булевы операторы AND, OR, NOT (SQL)

Оператор AND (∧, и, конъюнкция, логическое умножение)

T **AND** T = T
T **AND** F = F
F **AND** T = F
F **AND** F = F

A	B	A AND B
F	F	F
T	F	F
F	T	F
T	T	T

```
select
       NAME
       ,SURNAME
from
       STUDENTS
where NAME='Марина' AND SURNAME='Шуст'
```

Оператор OR (∨,или, дизъюнкция, логическое сложение)

F **OR** F = F
F **OR** T = T
T **OR** F = T
T **OR** T = T

A	B	A OR B
F	F	F
F	T	T
T	F	T
T	T	T

```
select
       NAME
       ,SURNAME
       ,STIPEND
from
       STUDENTS
where SURNAME='Шуст' OR STIPEND > 800;
```

Оператор NOT (¬, инверсия, логическое отрицание)

«Обращение» результата. Аналог умножения на -1

```
select
	NAME
	,SURNAME
	,STIPEND
from
	STUDENTS
where STIPEND > 700 and not name = 'Роман';
```

Приоритетность логических операций в SQL

Уровни приоритета операторов показаны в следующей таблице. Оператор с более высоким уровнем приоритета выполняется прежде, чем оператор с более низким уровнем.

1. **()**
2. **NOT**
3. **AND**
4. **OR**

Применение отрицания к операторам сравнения

NOT =	<>
NOT >	<=
NOT <	>=
<> (!=)	=
>=	<
<=	>

Примеры:

```
SELECT * FROM STUDENTS  WHERE NAME='Виталий';
```

ID	SURNAME	NAME	GENDER	STIPEND	COURSE	CITY	BIRTHDAY	UNIV_ID
1	Кабанов	Виталий	m	550.00	4	Харьков	1990-12-01	2
17	Осипуков	Виталий	m	500.00	1	Днепропетровск	1993-08-10	7
37	Запорожец	Виталий	m	350.00	5	Луцк	1989-02-28	13

```
SELECT
	*
FROM STUDENTS
WHERE
	(NAME='Виталий' AND SURNAME='Кабанов')
	OR (NAME='Виталий' AND SURNAME='Осипуков');
```

ID	SURNAME	NAME	GENDER	STIPEND	COURSE	CITY	BIRTHDAY	UNIV_ID
1	Кабанов	Виталий	m	550.00	4	Харьков	1990-12-01	2
17	Осипуков	Виталий	m	500.00	1	Днепропетровск	1993-08-10	7

```
select
	*
from STUDENTS
where NAME = 'Виталий'
	and (SURNAME = 'Кабанов' or SURNAME = 'Осипуков');
```

Теперь выберем всех Виталиев, но не Кабанов и не Осипуков

```
select *
from STUDENTS
where NAME = 'Виталий'
         and not (SURNAME = 'Кабанов' or SURNAME = 'Осипуков');
```

Используем правило Де Моргана

```
select
       *
from STUDENTS
where NAME = 'Виталий'
         and SURNAME <> 'Кабанов'
         and SURNAME <> 'Осипуков';
```

Операторы IN, BETWEEN, IS NULL, LIKE

IN принадлежность множеству

```
select * from STUDENTS where STIPEND in (500, 650);
```

BETWEEN диапазон

```
select * from STUDENTS where STIPEND between 500 and 650;
```

Примеры операторов IN и BETWEEN

Выбрать студентов у которых стипендия находится в диапазоне от 500 до 650.

```
select * from STUDENTS
where STIPEND = 500
         or STIPEND = 550
         or STIPEND = 600
         or STIPEND = 650;

SELECT * FROM STUDENTS where STIPEND >= 500 AND STIPEND <= 650

SELECT * FROM STUDENTS where STIPEND IN (500, 550, 600, 650);
SELECT * FROM STUDENTS where STIPEND between 500 and  650;

select * from UNIVERSITIES where CITY = 'Киев';
select * from STUDENTS where UNIV_ID in (select ID from UNIVERSITIES where CITY = 'Киев');
```

IS NULL

```
select * from STUDENTS where BIRTHDAY = null; -- неправильно
select * from STUDENTS where BIRTHDAY IS NULL;
```

LIKE – выбор по образцу

like ‘pattern’

‘%’ – 0 или несколько символов

‘_’ – один символ на конкретном месте

'Иг%'	Строка произвольной длины начинающаяся на Иг	**Иг**орь, **Иг**нат
'%ко'	Строка произвольной длины заканчивающаяся на ко	Козьмен**ко,** Саен**ко….**
%ме%'	Строка произвольной длины но должна содержать ме	Козь**ме**нко, Пи**ме**нчук
'____т'	Строка длиной 5 символов заканчивающаяся на т	Игнат
'_и_о%'	Строка, у которой второй	Т**и**м**о**фей

	символ ‘и’, затем следует **один обязательный символ, затем обязательный символ ‘о’** и потом произвольное количество символов	

```
select * from STUDENTS where NAME LIKE '_и_о%'
```

Использование select в операторе from (свойство реляционной замкнутости)

```
select
	*
from
(
	select
		SURNAME as last_name
	   ,NAME as first_name
	   ,BIRTHDAY as student_birthday
	   ,UNIV_ID as student_university
	   ,COURSE as student_course
	   ,DATEDIFF(YY,birthday,GETDATE()) as student_age
	from STUDENTS
) as st
where student_course = 1
	and student_age >= 23;
```

Для MySQL

```
select * from
	(select
		SURNAME as last_name
		,NAME as first_name
		,BIRTHDAY as student_birthday
		,UNIV_ID as student_university
		,COURSE as student_course
		,TIMESTAMPDIFF(YEAR, birthday, CURDATE()) as student_age
	from STUDENTS) as st
where student_course = 1 AND student_age >= 23;
```