Лабораторная работа 1

Структура оператора SELECT и формирование условий выбора

SQL (Structured Query Language) – язык структурированных запросов, является основным языком определения, манипулирования и управления данными в современных СУБД.

Операторы SQL разделены на три группы:

- Операторы манипулирования данными (Data Manipulation Language, DML) — предназначены для выборки и изменения данных: SELECT, INSERT, UPDATE, MERGE, DELETE.
- Операторы определения данных (Data Definition Language, DDL) — предназначены для создания и модификации объектов базы данных. Основными операторами этой группы являются: CREATE, ALTER, DROP.
- Операторы управления данными (Data Control Language, DCL) — предназначены для предоставления пользователям прав на выполнение определенных действий с базой данных: GRANT, REVOKE.

Оператор SELECT

Оператор SELECT предназначен для выборки данных из таблиц, то есть он реализует одно из основных назначений базы данных — предоставлять пользователю информацию. Результатом выполнения оператора SELECT является таблица.

Структура оператора SELECT

Структура оператора SELECT может быть представлена в следующем виде:

```
SELECT [ALL|DISTINCT] {список столбцов или выражений}
[FROM       {список таблиц}]
```

```
[WHERE     {условия выбора}]
[GROUP BY  {столбцы группировки}]
[HAVING    {условия на группу}];
[ORDER BY  {столбцы сортировки [ASC|DESC]]}
[LIMIT   {N}][OFFSET{M}];
```

(Квадратными скобками отмечены необязательные элементы.)

Оператор **SELECT** начинается со списка столбцов или выражений, значения которых будет отображаться в результате выполнения запроса. По умолчанию **SELECT** не исключает дублирование строк. Для исключения дублирования следует использовать ключевое слово **DISTINCT.**

В предложении **FROM** указываются источники данных. В качестве таких источников можно использовать таблицы базы данных, таблицы которые возвращают подзапросы или представления.

Предложение **WHERE** содержит условия выбора строк, а также может содержать условия соединения таблиц в многотабличных запросах.

В предложении **GROUP BY** можно указать столбцы, по которым следует осуществить группировку. Группировка состоит в том, что несколько строк, имеющих совпадающие значение столбцов, по которым осуществляется группировка, объединяются в одну строку.

При наличии **группировки** в предложении **HAVING** можно указать условия на группу. Результат выполнения запроса будет содержать данные только о тех группах записей, которые удовлетворяют этому условию.

Результат выполнения запроса может быть упорядочен по значениям одного или нескольких столбцов. В предложении **ORDER BY** указываются имена столбцов, по значению которых следует отсортировать результат выполнения запроса. По умолчанию строки упорядочиваются в порядке возрастания значений столбца. Для сортировки в порядке убывания после имени столбца следует указать параметр **DESC**. Если указать несколько столбцов, то результат будет упорядочиваться сначала по значению первого столбца; строки, имеющие

одинаковые значения первого столбца, упорядочиваются по значению второго столбца, и так далее.

Предложение **LIMIT N** позволяет ограничить количество строк в результате выполнения запроса. При наличии этого предложения будет выводиться не более первых **N** строк результата. Как правило, это предложение используется в запросах с сортировкой результата. При использовании **LIMIT N** можно использовать **OFFSET M**. В этом случае, сначала, пропускаются первые **M** строк результата, после этого выводится **N** следующих строк.

При изучении SQL следует обратить внимание на то, что для создания запроса необходимо:

1. Определить структуру запроса, соответствующую заданной задаче обработки данных.
2. Синтаксически правильно записать запрос.

В своей простейшей форме оператор SELECT должен включать в себя следующее:

- предложение SELECT, где указываются имена столбцов, значение которых будет отображаться в результате выполнения запроса;
- предложение FROM, в котором указывается имя таблицы, содержащей данные.

```
SELECT <список столбцов>
FROM <таблица>;
```

Запрос 2.1. Вывод содержимого одного столбца

```
SELECT employee_id
FROM Employees;
```

Запрос 2.2. Вывод содержимого нескольких столбцов

```
SELECT employee_id, first_name, last_name, department_id
FROM Employees
```

Если нужно вывести значения всех столбцов, то вместо списка столбцов указывается символ *.

Запрос 2.3. Вывод значений всех столбцов

```
SELECT *
FROM Employees
```

Исключение дублирования данных

Для того чтобы исключить повторения значений следует добавить ключевое слово DISTINCT.

Запрос 2.4. Вывод значений столбца job_id таблицы Employees, без дублирования.

```
SELECT DISTINCT job_id
FROM Employees
```

Условия выбора

Для того чтобы выводить только те данные, которые удовлетворяют определенным условиям, оператор SELECT должен содержать предложение **WHERE**, которое содержит условное выражение.

```
SELECT {список столбцов}
FROM {таблица}
WHERE {условное выражение};
```

Условное выражение для каждой строки таблицы может принимать значения: **ИСТИНА (TRUE), ЛОЖЬ (FALSE), НЕОПРЕДЕЛЕНО (UNKNOWN).** Результат выполнения запроса будет содержать только те строки, для которых условное выражение будет иметь значение **ИСТИНА(TRUE).**

Запрос 2.5. Вывод данных о сотрудниках, зарплата которых больше 15000.

```
SELECT employee_id, first_name, last_name, salary, department_id
FROM Employees
WHERE salary > 15000
```

В условных выражениях предложения WHERE, могут быть использованы операторы сравнения: =, >, < и логические операторы: NOT, AND, OR. Логические операторы используются для формирования сложных условий выбора и имеют разный приоритет. Сначала выполняются все операторы NOT, потом операторы AND; операторы OR выполняются в последнюю очередь.

Запрос 2.6. Вывод данных о сотрудниках, которые работают в отделе 50 и занимают должность ST_MAN.

```
SELECT employee_id, first_name, last_name, department_id, job_id
FROM Employees
WHERE (department_id = 50) AND (job_id= 'ST_MAN')
```

Запрос 2.7. Вывод данных о договорах сотрудника 155, заключенных 15.03.2018 , и обо всех договорах, заключенных 02.11.219.

```
SELECT * FROM Orders
WHERE (salesman_id = 155) AND (order_date ='15.03.2018')
OR (order_date ='02.11.2019');
```

Специальные выражения

Для формирования условий выбора можно использовать специальные выражения, представленные в таблице 2.1.

Таблица 2.1. Специальные выражения

Выражение	Описание	Пример
LIKE 'шаблон'	Совпадет ли часть строкового значения с заданным шаблоном.	first_name LIKE '_a%'

BETWEEN V_MIN AND V_MAX	Находится ли значение столбца в диапазоне от V_MIN до V_MAX	rating_e BETWEEN 2 And 4
IN(список)	Совпадает ли значение столбца с одним из элементов списка значений.	rating_e IN(2;4)
IS NULL	Значение столбца неопределенно	rating_e IS NULL

Выражение LIKE

Выражение LIKE применяется при работе со строками. Оно проверяет, совпадет ли часть строки с заданным шаблоном. Если совпадение найдено, то оператор возвращает значение **TRUE**, в противном случае возвращается **FALSE**.

Для создания шаблонов в выражении LIKE используются следующие символы:

- символ подчеркивания _ обозначает один символ;
- символ процента % обозначает любую, том числе и пустую, последовательность символов.

Синтаксис:
{имя столбца} LIKE 'шаблон'

Запрос 2.8. Вывод данных о сотрудниках, имена которых начинаются на букву L.

```
SELECT employee_id, first_name, last_name, department_id
FROM  Employees
WHERE first_name LIKE 'L%';
```

Запрос 2.9. Вывести имена сотрудников, вторым символом которых является буква **a**.

```
SELECT DISTINCT first_name
FROM  Employees
WHERE first_name LIKE '_a%';
```

Запрос 2.10. Вывести имена сотрудников, которые состоят из четырех символов, начинаются на букву J и заканчиваются буквой n.

```
SELECT DISTINCT first_name
FROM  Employees
WHERE first_name LIKE 'J__n'
```

```
first_name|
----------+
Jean      |
John      |
```

Для поиска в строке, символов _ и %, при построении шаблона, используется опция ESCAPE '/'. Символ, который в шаблоне, будет располагаться после /, будет рассматриваться как символ поиска. Вместо символа / можно использовать и другие символы, например !.

Запрос 2.11. Вывести имя и адрес клиентов, столбец address которых содержит символ _.

```
SELECT c_name, address
FROM Customers
WHERE address LIKE '%/_%' ESCAPE '/'
```

```
c_name                      |address                                    |
----------------------------+-------------------------------------------+
DAIKIN INDUSTRIES           |1304 Kanaoka-cho, Osaka 591_8511, Japan    |
SAKAI HEAVY INDUSTRIES, LTD| Seiwa Bldg., 1-4-8, Minato_ku              |
Microsoft Corporation       |Microsoft Way Redmond, WA 98052_7329 USA   |
```

Выражение BETWEEN

Выражение BETWEEN используется для того чтобы результат запроса содержал только те строки, в которых значение проверяемого столбца находится в заданном диапазоне, включая его границы.

Синтаксис:

```
{имя столбца} BETWEEN V_MIN AND V_MAX
```

V_MIN – нижняя граница диапазона;
V_MAX – верхняя граница диапазона.

Выражение **BETWEEN** эквивалентно двум операциям сравнения, объединенным логическим оператором AND.

```
({имя столбца} >= V_MIN) AND ({имя столбца} >= V_MAX)
```

Для определения границ диапазона можно использовать числа, даты и строки.

Запрос 2.12. Вывести данные о сотрудниках, зарплата которых находится в определенном диапазоне.

```
SELECT employee_id, first_name, last_name, department_id
FROM  Employees
WHERE salary BETWEEN  6000 AND 8000;
```

Выражение IN

Выражение IN используется для того чтобы результат запроса содержал только те строки, в которых значение проверяемого столбца совпадает с одним из значений указанных в списке**.**

Синтаксис:

```
{имя столбца} IN {список значений}
```

Список значений, может формироваться в результате выполнения оператора SELECT (подзапроса).

Запрос 2.13. Вывести данные о сотрудниках, которые работают в отделах с определенными номерами.

```
SELECT employee_id, first_name, last_name, department_id
FROM Employees
WHERE department_id IN (40, 10, 110)
```

Выражение IS NULL

Выражение **IS NULL** используется для определения строк с неопределенным значением заданного столбца.

Синтаксис:

```
{имя столбца} IS NULL
```

Это выражение возвращает значение TRUE, если значение проверяемого столба будет NULL.

Запрос 2.14. Вывести данные о сотрудниках, у которых не задан номер отдела.

```
SELECT employee_id, first_name, last_name, department_id
FROM  Employees
WHERE department_id IS NULL;
```

```
employee_id|first_name|last_name|department_id|
-----------+----------+---------+-------------+
        178|Kimberely |Grant    |             |
```

Вычисляемые столбцы

В предложении SELECT, кроме списка столбцов таблиц участвующих в запросе, могут присутствовать вычисляемые столбцы, которые представляют собой выражения, состоящие из имен столбцов, констант, функций и арифметических операций. Значению вычисляемого столбца можно присвоить имя. Для этого используется следующая конструкция:

```
{Выражение} As {псевдоним}
```

Рассмотрим примеры использования вычисляемых столбцов. Значение столбца **commission_pct**, в таблице Employees обозначает надбавку к зарплате как часть от заработной платы.

Общая зарплата с учетом комиссионных может быть вычислена с использованием выражения:

```
salary*(1+commission_pct) As Total_Salary
```

Следует иметь в виду то, что у некоторых сотрудников значение столбца commission_pct равно NULL. А если один из элементов арифметического выражения равен NULL, то и все выражение будет иметь значение NULL. Данную проблему можно решить, используя специальные функции, которые мы рассмотрим позже.

Запрос 2.15.. Для сотрудников, которые получают комиссионные, вывести зарплату с учетом комиссионных.

```
SELECT employee_id, first_name, last_name, salary,commission_pct
c_pct,
ROUND(salary*(1+commission_pct)) As Total_Salary
FROM  Employees
WHERE commission_pct IS NOT null
```

```
employee_id|first_name |last_name |salary  |c_pct|total_salary|
-----------+-----------+----------+--------+-----+------------+
        145|John       |Russell   |14000.00|0.400|       19600|
        146|Karen      |Partners  |13500.00|0.300|       17550|
        147|Alberto    |Errazuriz |12000.00|0.300|       15600|
        148|Gerald     |Cambrault |11000.00|0.300|       14300|
        149|Eleni      |Zlotkey   |10500.00|0.200|       12600|
        150|Peter      |Tucker    |10000.00|0.300|       13000|
        151|David      |Bernstein | 9500.00|0.250|       11875|
```

Операция конкатенации

Операция конкатенации (слияния) используются для того чтобы объединить, при выводе данных, два или несколько столбцов или литералов в один столбец. Синтаксис:

```
{столбец1/литерал1} || {столбец2/литерал2}… As {псевдоним}
```

Операцию конкатенации можно применять для строк, чисел и дат. Даты и числа при слиянии конвертируются в строковые значения.

Пример 2.16. Вывести данные о заказах, оформленных сотрудником 165, объединив их в одну строку.

```
SELECT 'Order  '||order_id||' from  '||order_date||' is
'||status AS Order_Statys
FROM Orders
WHERE salesman_id =165
```

```
order_statys                          |
--------------------------------------+
Order  66 from  2020-01-23 is Pending|
Order  67 from  2018-10-22 is Shipped|
```

Условные выражения

Довольно часто значение столбца, которое должен вернуть SQL запрос зависит от условий, которые нужно проверять для каждой строки. Для реализации подобного выбора используются выражение CASE. Используя это выражение можно реализовать условную логику if-then-else в операторе SELECT. Можно использовать два варианта выражения CASE:

- Выражение CASE с параметром
- Выражение CASE с условием

Выражение CASE с параметром

Выражение CASE с параметром имеет следующий синтаксис:

```
CASE {параметр}
     WHEN {значение1} THEN {результат1}
    [WHEN {значение2} THEN {результат2}
...
     WHEN {значениеN} THEN {результатN}]
    [ELSE {результат_ELSE}]
END
```

Выражение CASE выполняется следующим образом:

Сравниваются значение {**параметр}** со значениями {**значение i}** в предложениях WHEN, и возвращает результат **{результатi}** первого предложения в котором будет выполнено условие {**параметр} =** {**значениеi}.**

Если ни в одном из предложений WHEN не выполняется условие {**параметр} =** {**значениеi}** то возвращается значение {**результат_ELSE}**. Если предложение ELSE отсутствует, то выражение CASE вернет результат NULL.

Возвращаемый результат, может быть значением или выражением. Выражения **{параметр}** и **{значение}** должны иметь один и тот же тип данных. Все возвращаемые значения должны иметь одинаковый тип данных.

Примечание. Выражение CASE может содержать другие выражения CASE.

Пример 2.17. Вывести данные о сотрудниках, и размере их премии, которая задана в виде фиксированной суммы, размер которой зависит от отдела, где работает сотрудник.

```
SELECT department_id, employee_id, first_name, last_name, job_id,
salary,
       CASE department_id
              WHEN  10 THEN  1000
              WHEN  30 THEN  1200
              WHEN  40 THEN  1500
              ELSE 500
        END AS bonus
FROM Employees
WHERE department_id IN (10,20,30,40)
ORDER BY department_id;
```

```
department_id|employee_id|first_name|last_name |job_id  |salary  |bonus|
-------------+-----------+----------+----------+--------+--------+-----+
           10|        200|Jennifer  |Whalen    |AD_ASST | 4400.00| 1000|
           20|        201|Michael   |Hartstein |MK_MAN  |13000.00|  500|
           20|        202|Pat       |Fay       |MK_REP  | 6000.00|  500|
```

Выражение CASE с условием

Синтаксис:

```
CASE
     WHEN {условие1} THEN {результат1}
    [WHEN {условие2} THEN {результат2}
…
     WHEN {условиеN} THEN {результатN}]
     [ELSE {результат_ELSE}]
END;
```

При использовании этой разновидности оператора CASE, последовательно поверяются значения условных выражений, в предложениях WHEN и возвращается результат из первого предложения, в котором это выражение будет иметь значение TRUE.

Пример 2.18. Вывести данные о сотрудниках и размере их премии, которая зависит от зарплаты сотрудника.

```
SELECT department_id, employee_id, first_name, last_name, job_id,
salary,
   CASE
       WHEN  salary > 10000 THEN 5000
       WHEN  salary > 7000 THEN  3000
       WHEN  salary > 5000 THEN  2000
       ELSE  1000
    END AS bonus
FROM Employees
WHERE department_id in (10,20,30,40)
ORDER BY department_id;
```

Сортировка

Результат выполнения оператора **SELECT** может быть упорядочен по значению одного или нескольких столбцов. Для этого служит предложение ORDER BY, которое имеет следующий синтаксис:

```
ORDER BY {имя столбца | номер столбца  [ASC|DESC]}
```

Пример 2.19. Вывести данные о сотрудниках отдела 30, упорядочив их в порядке убывания зарплаты.

```
SELECT employee_id, first_name, last_name, department_id, salary
FROM  Employees
WHERE department_id =30
ORDER BY salary DESC;
```

```
employee_id|first_name|last_name |department_id|salary  |
-----------+----------+----------+-------------+--------+
        114|Den       |Raphaely  |           30|11000.00|
        115|Alexander |Khoo      |           30| 3100.00|
        116|Shelli    |Baida     |           30| 2900.00|
        117|Sigal     |Tobias    |           30| 2800.00|
        118|Guy       |Himuro    |           30| 2600.00|
        119|Karen     |Colmenares|           30| 2500.00|
```

Ограничение количества строк в результате выполнения запроса

Используя предложения **LIMIT N** и **OFFSET M** можно ограничить количество строк в результате выполнения запроса. Если оператор SELECT содержит

предложение **LIMIT N**, то будет выводиться не более первых **N** строк результата. При использовании **LIMIT N** можно использовать **OFFSET M**. В этом случае, сначала, пропускаются первые **M** строк результата, после этого выводится **n** следующих строк.

Пример 2.20. Вывести данные о сотрудниках отдела 30, зарплата которых занимает места с 3 по 5.

```
SELECT employee_id, first_name, last_name, department_id, salary
FROM Employees
WHERE department_id =30
ORDER BY salary DESC
LIMIT 3 OFFSET 2
```

```
employee_id|first_name|last_name|department_id|salary |
-----------+----------+---------+-------------+-------+
        116|Shelli    |Baida    |           30|2900.00|
        117|Sigal     |Tobias   |           30|2800.00|
        118|Guy       |Himuro   |           30|2600.00|
```

Задание

Задача 1. Вывести location_id городов, в которых расположены отделы фирмы.

Задача 2. Вывести данные о товарах, у которых столбец rating_p имеет значение 3 или 4, a price > 700.

Задача 3. Вывести last_name сотрудников, у которых last_name содержит две и более буквы e.

Задача 4. Вывести значения столбцов employee_id , department_id, first_name, last_name, salary, job_id сотрудников, которые получают зарплату > 1100, но не являются менеджерами. Менеджерами являются те сотрудники, у которых столбец job_id содержит подстроку 'MAN'.

Задача 5. Вывести first_name, last_name, job_id и суммарную зарплату за год в следующем виде:

Michael Hartstein занимает должность MK_MAN и зарплата за год составляет 156000.

Задача 6. Вывести значения столбцов employee_id , department_id, first_name, last_name, salary, job_id и столбец level, который должен принимать следующие значения: low – если зарплата сотрудника <= 5000; midl – если зарплата сотрудника >5000 но <=10000; high – если зарплата сотрудника >10000.

Задача 7. Вывести данные о 3х сотрудниках с наибольшим размером премии (bonus), которую они должны получить. Размер премии сотрудника рассчитывается по формуле bonus = 0.1*salary*rating_e.