МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский технический университет связи и информатики»
(МТУСИ)

Кафедра
«Сетевые информационные технологии и сервисы»

Лабораторная работа 4

По дисциплине «Интеллектуальные базы данных»

«Создание простых запросов на выборку»

Выполнил студент:
М092301(75) Леонов Н.Н.
Вариант 17
Проверил:
Ст.пр. Фатхулин Т.Д.

Москва 2024

1. Цель работы

В данной лабораторной работе необходимо выполнить запросы на выборку с использованием операторов и функций.

2. Задание

При выполнении лабораторной работы необходимо для заданной предметной области средствами MySQL:

- построить несколько простых запросов на выборку с использованием операторов и функций MySQL;
- составить отчет по лабораторной работе.

3. Теоретические сведения

Рассмотрим следующие вопросы:

- выборка данных из одной таблицы с помощью оператора *SELECT*;
- использование в запросах операторов и встроенных функций MySQL.

Для выполнения запросов (извлечения строк из одной или нескольких таблиц БД) используется оператор *SELECT*. Результатом запроса всегда является таблица. Результаты запроса могут быть использованы для создания новой таблицы. Таблица, полученная в результате запроса, может стать предметом дальнейших запросов.

Общая форма оператора *SELECT*:

SELECT столбцы FROM таблицы
[WHERE условия]
[GROUP BY группа [HAVING групповые_условия]]
[ORDER BY имя_поля]
[LIMIT пределы];

Оператор *SELECT* имеет много опций. Их можно использовать или не использовать, но они должны указываться в том порядке, в каком они приведены. Если требуется вывести все столбцы таблицы, необязательно перечислять их после ключевого слова *SELECT*, достаточно заменить этот список символом *.

```
mysql> SELECT * FROM orders;
+----------+-----------+-----------+---------------------+----------+
| order_ID | o_user_ID | o_book_ID | o_time              | o_number |
+----------+-----------+-----------+---------------------+----------+
|        1 |         3 |         8 | 2009-01-04 10:39:38 |        1 |
|        2 |         6 |        10 | 2009-02-10 09:40:29 |        2 |
|        3 |         1 |        20 | 2009-02-18 13:41:05 |        4 |
|        4 |         4 |        20 | 2009-03-10 18:20:00 |        1 |
|        5 |         3 |        20 | 2009-03-17 19:15:36 |        1 |
+----------+-----------+-----------+---------------------+----------+
5 rows in set (0.02 sec)
```

Список столбцов в операторе *SELECT* используют, если нужно изменить порядок следования столбцов в результирующей таблице или выбрать часть столбцов.

```
mysql> SELECT cat_name, cat_ID FROM catalogs;
+-----------------+--------+
| cat_name        | cat_ID |
+-----------------+--------+
| Программирование |      1 |
| Интернет        |      2 |
| Базы данных     |      3 |
| Сети            |      4 |
| Мультимедиа     |      5 |
+-----------------+--------+
5 rows in set (0.01 sec)
```

Условия выборки. Гораздо чаще встречается ситуация, когда необходимо изменить количество выводимых строк. Для выбора записей, удовлетворяющих определенным критериям поиска, можно использовать конструкцию *WHERE*.

```
mysql> SELECT user_ID, u_surname FROM users
    -> WHERE u_status='active';
+---------+-----------+
| user_ID | u_surname |
+---------+-----------+
|       1 | Иванов    |
|       3 | Симонов   |
|       4 | Кузнецов  |
+---------+-----------+
3 rows in set (0.03 sec)
```

В запросе можно использовать ключевое слово *DISTINCT*, чтобы результат не содержал повторений уже имеющихся значений, например:

```
mysql> SELECT DISTINCT u_status FROM users;
+----------+
| u_status |
+----------+
| active   |
| passive  |
| lock     |
| gold     |
+----------+
4 rows in set (0.01 sec)
```

Сортировка. Результат выборки – записи, расположенные в том порядке, в котором они хранятся в БД. Чтобы отсортировать значения по одному из столбцов, необходимо после конструкции *ORDER BY* указать этот столбец, например:

```
mysql> SELECT * FROM orders ORDER BY o_user_ID;
+----------+-----------+-----------+---------------------+----------+
| order_ID | o_user_ID | o_book_ID | o_time              | o_number |
+----------+-----------+-----------+---------------------+----------+
|        3 |         1 |        20 | 2009-02-18 13:41:05 |        4 |
|        1 |         3 |         8 | 2009-01-04 10:39:38 |        1 |
|        5 |         3 |        20 | 2009-03-17 19:15:36 |        1 |
|        4 |         4 |        20 | 2009-03-10 18:20:00 |        1 |
|        2 |         6 |        10 | 2009-02-10 09:40:29 |        2 |
+----------+-----------+-----------+---------------------+----------+
5 rows in set (0.00 sec)
```

Сортировку записей можно производить по нескольким столбцам (их следует указать после слов *ORDER BY* через запятую). Число столбцов, указываемых в конструкции *ORDER BY*, не ограничено.

По умолчанию сортировка производится в прямом порядке (записи располагаются от наименьшего значения поля сортировки до наибольшего). Обратный порядок сортировки реализуется с помощью ключевого слова *DESC*:

```
mysql> SELECT o_time FROM orders ORDER BY o_time DESC;
+---------------------+
| o_time              |
+---------------------+
| 2009-03-17 19:15:36 |
| 2009-03-10 18:20:00 |
| 2009-02-18 13:41:05 |
| 2009-02-10 09:40:29 |
| 2009-01-04 10:39:38 |
+---------------------+
5 rows in set (0.27 sec)
```

Для прямой сортировки существует ключевое слово *ASC*, но так как записи сортируются в прямом порядке по умолчанию, данное ключевое слово опускают.

Ограничение выборки. Результат выборки может содержать тысячи записей, вывод и обработка которых занимают значительное время. Поэтому информацию часто разбивают на страницы и предоставляют ее пользователю частями. Постраничная навигация используется при помощи ключевого слова *LIMIT*, за которым следует число выводимых записей. Следующий запрос

извлекает первые 5 записей, при этом осуществляется обратная сортировка по полю *b_count*:

```
mysql> SELECT book_ID, b_count FROM books
    -> ORDER BY b_count DESC
    -> LIMIT 5;
+---------+---------+
| book_ID | b_count |
+---------+---------+
|      28 |      20 |
|      25 |      20 |
|      26 |      15 |
|      29 |      12 |
|       9 |      12 |
+---------+---------+
5 rows in set (0.03 sec)
```

Для извлечения следующих пяти записей используется ключевое слово *LIMIT* с двумя цифрами. Первая указывает позицию, начиная с которой необходимо вернуть результат, вторая цифра – число извлекаемых записей, например:

```
mysql> SELECT book_ID, b_count FROM books
    -> ORDER BY b_count DESC
    -> LIMIT 5,5;
+---------+---------+
| book_ID | b_count |
+---------+---------+
|       1 |      10 |
|      27 |      10 |
|      24 |       8 |
|      30 |       8 |
|      20 |       6 |
+---------+---------+
5 rows in set (0.00 sec)
```

При определении смещения нумерация строк начинается с нуля (поэтому в последнем примере для шестой строки указано смещение 5).

Группировка записей. Конструкция *GROUP BY* позволяет группировать извлекаемые строки. Она полезна в комбинации с функциями, применяемыми к группам строк. Эти функции (табл. 6) называются агрегатами (суммирующими функциями) и вычисляют одно значение для каждой группы, создаваемой конструкцией *GROUP BY*. Функции позволяют узнать число строк в группе, подсчитать среднее значение, получить сумму значений столбцов. Результирующее значение рассчитывается для значений, не равных *NULL* (исключение – функция *COUNT(*)*). Допустимо использование этих функций в запросах без группировки (вся выборка – одна группа).

Пример использования функции *COUNT( )*, которая возвращает число строк в таблице, значения указанного столбца для которых отличны от *NULL*:

```
mysql> SELECT COUNT(book_ID) FROM books;
+----------------+
| COUNT(book_ID) |
+----------------+
|             30 |
+----------------+
1 row in set (0.16 sec)
```

Таблица 6

Обозначение	**Описание**
AVG ([DISTINCT] expr)	Возвращает среднее значение аргумента *expr*. В качестве аргумента обычно выступает имя столбца. Необязательное слово *DISTINCT* позволяет обрабатывать только уникальные значения столбца *expr*
COUNT ( )	Подсчитывает число записей и имеет несколько форм. Форма *COUNT (выражение)* возвращает число записей в таблице, поле *выражение* для которых не равно *NULL*. Форма *COUNT(*)* возвращает общее число строк в таблице независимо от того, принимает какое-либо поле значение *NULL* или нет. Форма *COUNT (DISTINCT выражение1, выражение2, ...)* позволяет использовать ключевое слово *DISTINCT*, которое позволяет подсчитать только уникальные значения столбца
MIN ([DISTINCT] expr)	Возвращает минимальное значение среди всех непустых значений выбранных строк в столбце *expr*. Необязательное слово *DISTINCT* позволяет обрабатывать только уникальные значения столбца *expr*
MAX ([DISTINCT] expr)	Возвращает максимальное значение среди всех непустых значений выбранных строк в столбце *expr*. Необязательное слово *DISTINCT* позволяет обрабатывать только уникальные значения столбца *expr*

STD (expr)	Возвращает стандартное среднеквадратичное отклонение в аргументе *expr*
STDDEV_SAMP (expr)	Возвращает выборочное среднеквадратичное отклонение в аргументе *expr*
SUM ([DISTINCT] expr)	Возвращает сумму величин в столбце *expr*. Необязательное слово *DISTINCT* позволяет обрабатывать только уникальные значения столбца *expr*

Использование ключевого слова *DISTINCT* с функцией *COUNT( )* позволяет вернуть число уникальных значений *b_cat_ID* в таблице *books*, например:

```
mysql> SELECT COUNT(DISTINCT b_cat_ID) FROM books;
+--------------------------+
| COUNT(DISTINCT b_cat_ID) |
+--------------------------+
|                        5 |
+--------------------------+
1 row in set (0.02 sec)
```

В *SELECT*-запросе столбцу можно назначить новое имя с помощью оператора *AS*. Например, результату функции *COUNT( )* присваивается псевдоним *total*:

```
mysql> SELECT COUNT(order_ID) AS total FROM orders;
+-------+
| total |
+-------+
|     5 |
+-------+
1 row in set (0.05 sec)
```

Использование функций в конструкции *WHERE* приведет к ошибке. В следующем примере показана попытка извлечения из таблицы *catalogs* записи с максимальным значением поля *cat_ID*:

```
mysql> SELECT * FROM catalogs WHERE cat_ID=MAX(cat_ID);
ERROR 1111 (HY000): Invalid use of group function
```

Решение задачи следует искать в использовании конструкции *ORDER BY*:

```
mysql> SELECT * FROM catalogs ORDER BY cat_ID DESC LIMIT 1;
+--------+-------------+
| cat_ID | cat_name    |
+--------+-------------+
|      5 | Мультимедиа |
+--------+-------------+
1 row in set (0.00 sec)
```

Для извлечения уникальных записей используют конструкцию *GROUP BY* с именем столбца, по которому группируется результат:

```
mysql> SELECT b_cat_ID FROM books
    -> GROUP BY b_cat_ID ORDER BY b_cat_ID;
+----------+
| b_cat_ID |
+----------+
|        1 |
|        2 |
|        3 |
|        4 |
|        5 |
+----------+
5 rows in set (0.03 sec)
```

При использовании *GROUP BY* возможно использование условия *WHERE*:

```
mysql> SELECT b_cat_ID, COUNT(b_cat_ID) FROM books
    -> WHERE b_cat_ID > 2
    -> GROUP BY b_cat_ID
    -> ORDER BY b_cat_ID;
+----------+----------------+
| b_cat_ID | COUNT(b_cat_ID) |
+----------+----------------+
|        3 |              4 |
|        4 |              5 |
|        5 |              6 |
+----------+----------------+
3 rows in set (0.02 sec)
```

Часто при задании условий требуется ограничить выборку по результату функции (например, выбрать каталоги, где число товарных позиций больше 5). Использование для этих целей конструкции *WHERE* приводит к ошибке. Для решения этой проблемы вместо ключевого слова *WHERE* используется ключевое слово *HAVING*, располагающееся за конструкцией *GROUP BY*:

```
mysql> SELECT b_cat_ID, COUNT(b_cat_ID) AS total FROM books
    -> GROUP BY b_cat_ID
    -> HAVING total > 5
    -> ORDER BY b_cat_ID;
+----------+-------+
| b_cat_ID | total |
+----------+-------+
|        1 |     9 |
|        2 |     6 |
|        5 |     6 |
+----------+-------+
3 rows in set (0.00 sec)
```

Запрос, извлекающий уникальные значения столбца *b_cat_ID*, большие двух:

```
mysql> SELECT b_cat_ID, COUNT(b_cat_ID) FROM books
    -> GROUP BY b_cat_ID
    -> HAVING b_cat_ID > 2
    -> ORDER BY b_cat_ID;
+----------+-----------------+
| b_cat_ID | COUNT(b_cat_ID) |
+----------+-----------------+
|        3 |               4 |
|        4 |               5 |
|        5 |               6 |
+----------+-----------------+
3 rows in set (0.00 sec)
```

При этом в случае использования ключевого слова *WHERE* сначала производится выборка из таблицы с применением условия и лишь затем группировка результата, а в случае использования ключевого слова *HAVING* сначала происходит группировка таблицы и лишь затем выборка с применением условия. Допускается использование условия *HAVING* без группировки *GROUP BY*.

Использование функций. Для решения специфических задач при выборке удобны встроенные функции MySQL. Большинство функций предназначено для использования в выражениях *SELECT* и *WHERE*. Существуют также специальные функции группировки для использования в выражении *GROUP BY* (см. выше).

Каждая функция имеет уникальное имя и может иметь несколько аргументов (перечисляются через запятую в круглых скобках). Если аргументы отсутствуют, круглые скобки все равно следует указывать. Пробелы между именем функции и круглыми скобками недопустимы.

Число доступных для использования функций велико, в приложениях приведены наиболее полезные из них.

Пример использования функции, возвращающей версию сервера MySQL:

```
mysql> SELECT VERSION();
+----------------------+
| VERSION()            |
+----------------------+
| 5.0.51b-community-nt |
+----------------------+
1 row in set (0.00 sec)
```

Отметим также возможность использования оператора *SELECT* без таблиц вообще. В такой форме *SELECT* можно использовать как калькулятор:

```
mysql> SELECT 2+3;
+-----+
| 2+3 |
+-----+
|   5 |
+-----+
1 row in set (0.00 sec)
```

Можно вычислить любое выражение без указания таблиц, получив доступ ко всему разнообразию математических и других операторов и функций. Возможность выполнять математические расчеты на уровне *SELECT* позволяет проводить финансовый анализ значений таблиц и отображать полученные результаты в отчетах. Во всех выражениях MySQL (как в любом языке программирования) можно использовать скобки, чтобы контролировать порядок вычислений.

Операторы. Под операторами подразумеваются конструкции языка, которые производят преобразование данных. Данные, над которыми совершается операция, называются операндами.

В MySQL используются три типа операторов:

- арифметические операторы;
- операторы сравнения;
- логические операторы.

Арифметические операции. В MySQL используются обычные арифметические операции: сложение (+), вычитание (–), умножение (*),

деление (/) и целочисленное деление *DIV* (деление и отсечение дробной части). Деление на 0 дает безопасный результат *NULL*.

Операторы сравнения. При работе с операторами сравнения необходимо помнить о том, что, за исключением нескольких особо оговариваемых случаев, сравнение чего-либо со значением *NULL* дает в результате *NULL*. Это касается и сравнения значения *NULL* со значением *NULL*:

```
mysql> SELECT NULL=NULL;
+-----------+
| NULL=NULL |
+-----------+
|      NULL |
+-----------+
1 row in set (0.02 sec)
```

Корректнее использовать следующий запрос:

```
mysql> SELECT NULL IS NULL;
+--------------+
| NULL IS NULL |
+--------------+
|            1 |
+--------------+
1 row in set (0.00 sec)
```

Поэтому следует быть предельно внимательными при работе с операторами сравнения, если операнды могут принимать значения *NULL*.

Наиболее часто используемые операторы сравнения приведены в табл. 7.

Логические операторы. MySQL поддерживает все обычные логические операции, которые можно использовать в выражениях. Логические выражения в MySQL могут принимать значения 1 (истина), 0 (ложь) или *NULL*.

Кроме того, следует учитывать, что MySQL интерпретирует любое ненулевое значение, отличное от *NULL*, как значение «истина».

4. Выполнение лабораторной работы

Выполним несколько запросов для выборки данных из БД Cost Accounting.

Выведем все столбцы таблицы Employee с помощью команды SELECT * FROM Employee (рис. 1).

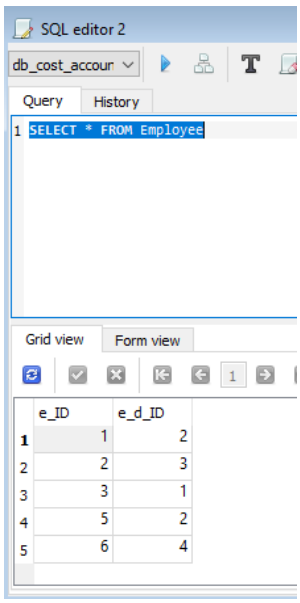

Рисунок 1 – Выборка всех столбцов из таблицы Employee

Выведем название отдела, в котором числится больше 10 сотрудников с помощью команды SELECT d_name FROM Department WHERE d_count_of_employees > 10 (рис. 2).

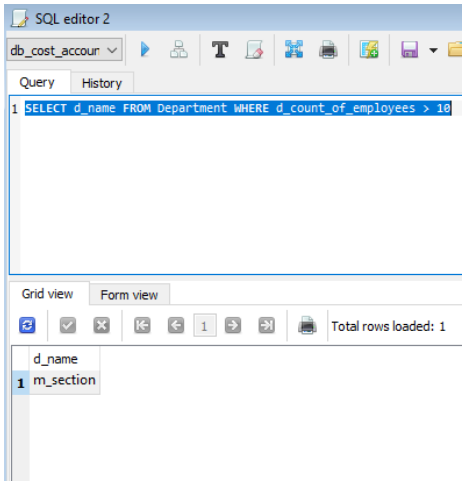

Рисунок 2 – Получение названия отдела, в котором больше 10 сотрудников

Отсортируем все столбцы данной таблицы по количеству сотрудников с помощью команды SELECT * FROM Department ORDER BY d_count_of_employees (рис. 3).

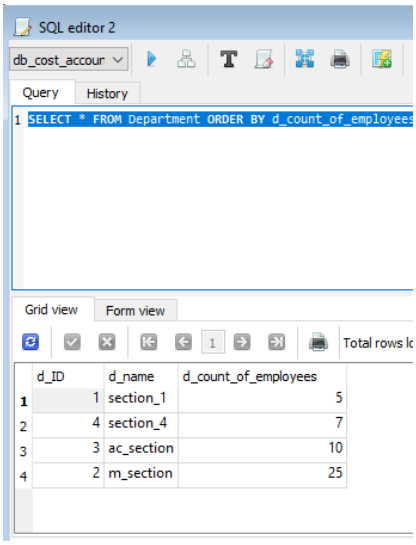

Рисунок 3 – Результат сортировки таблицы Department по количеству сотрудников

Также произведем сортировку столбцов таблицы Purchase по дате, начиная с самых свежих заказов с помощью команды SELECT * FROM Purchase ORDER BY p_date DESC (рис. 4).

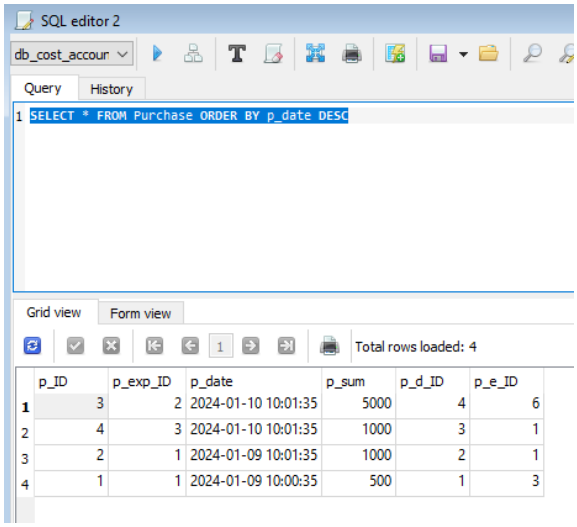

Рисунок 4 – Результат сортировки таблицы Purchase по дате в обратном порядке

Сделаем выборку стоимости двух наиболее свежих заказов с помощью команды SELECT p_sum FROM Purchase ORDER BY p_date DESC LIMIT 2 (рис. 5).

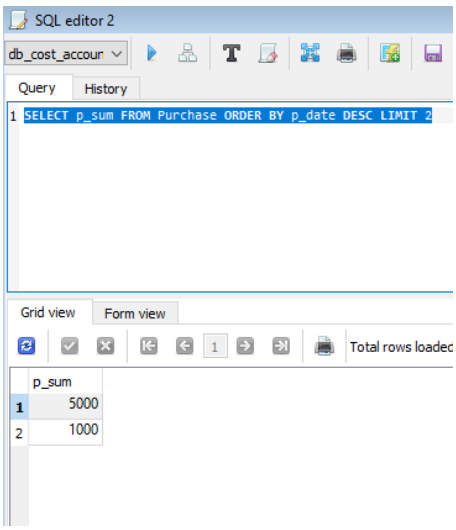

Рисунок 5 – Результат выборки стоимости двух наиболее свежих заказов

Выведем максимальный и минимальный денежный лимит из таблицы Expenses с помощью команды SELECT MAX(exp_limit) AS max, MIN(exp_limit) AS min FROM Expenses (рис. 6).

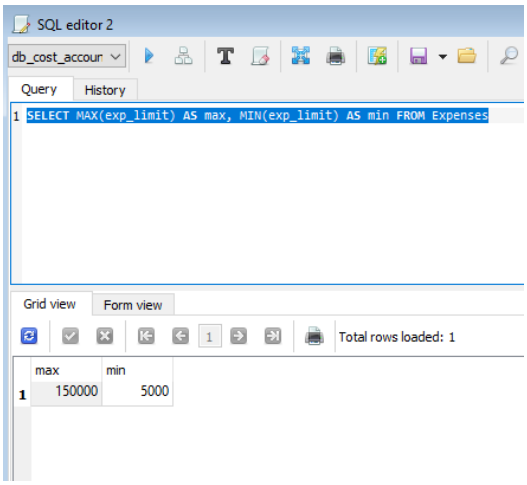

Рисунок 6 – Выборка макс. и мин. денежного лимита

Выводы

В ходе лабораторной работы были произведены простые запросы на выборки с использованием операторов и функций MySQL на основе заполненной базы данных «db_cost_accounting» в предыдущей лабораторной работе №3.