ИСПОЛЬЗОВАНИЕ СТРУКУТРИРОВАННОГО ЯЗЫКА ЗАПРОСОВ

Лабораторный практикум по Базам данных, занятие 2 «Выбор и модификация данных таблицы»

МГУ им. А.А. Кулешова, ФМЕ, 1 курс 2 семестр
заочная форма получения образования
специальность ИПОВС, 2020-2021 учебный год
доцент Сидоренко И.Н.

1 Цель работы

Используя данные базы данных из первой лабораторной работы, подготовить и реализовать серию запросов, связанных с выборкой информации и модификацией данных таблиц.

Содержание работы

1. Изучить состав, правила и порядок использования ключевых фраз оператора SELECT:

SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY

Порядок следования фраз в команде **SELECT** должен соответствовать приведенной выше последовательности.

3. Подготовить и выполнить запросы по выборке информации из таблицы базы данных для решения нижеприведенных задач.

1. *Выбрать любого хозяина и отобразить всех его питомцев.*
2. *Выбрать любой год и отобразить животных с годом рождения позже выбранного.*
3. *Вывести количество животных любого одного вида.*
4. *Вывести клички животных, имена хозяев которых содержат букву «А».*
5. *Вывести имена животных с весом, например, от 200 до 300г. (2 варианта).*
6. *Выбрать любого хозяина, найти у него животное с наименьшим весом и вывести запись об этом животном*
7. *Вывести имена животных, относящиеся к двум разным видам (использовать IN и Or).*
8. *Вывести записи, отсортированные по году рождения.*
9. *Подсчитать количество уникальных видов (distinct) животных.*
10. *Вывести имена животных, отсортированные в убывающем порядке.*
11. *Вывести название вида и количество животных каждого вида.*
12. *Вывести название вида и количество животных этого вида с весом, например, от 200 до 300г.*

13. Вывести средний вес всех животных с использованием псевдо имени столбца «Средний вес.».

14. Удалить строку 5.

Методические указания к выполнению лабораторной работы №2.

Запрос данных из таблицы MySQL SELECT

Запрос данных выполняется с помощью команды MySQL **SELECT**.

В общем виде синтаксис оператора SELECT имеет следующий вид:

SELECT [ALL/DISTINCT] <список атрибутов>/*
FROM <список таблиц> [WHERE <условие выборки>]
[GROUP BY <список атрибутов>] [HAVING <условие>]
[ORDER BY <список атрибутов>]
[UNION<выражение с оператором SELECT>]

В квадратных скобках указываются необязательные элементы.

Ключевое слово **ALL** означает, что результатом будут все строки, удовлетворяющие условию запроса, в том числе и одинаковые строки. **DISTINCT** означает, что в результирующий набор не включаются одинаковые строки. Далее идет список атрибутов исходной таблицы, которые будут включены в таблицу-результат. Символ ***** означает, что в таблицу-результат включаются все атрибуты исходной таблицы.

Обязательным ключевым словом является слово **FROM**, за ним следуют имена таблиц, к которым осуществляется запрос.

В предложении с ключевым словом **WHERE** задаются условия выборки строк таблицы. В таблицу-результат включаются только те строки, для которых условие, указанное в предложении WHERE, принимает значение истина.

В предложении с ключевым словом **GROUP BY** задается список атрибутов группировки (разъяснение этого и последующего ключевого слова будет представлено немного позднее).

В предложении **HAVING** задаются условия, накладываемые на каждую группу.

Ключевое слово **ORDER BY** задает операцию упорядочения строк таблицы-результата по указанному списку атрибутов.

Короче можно записать:

SELECT имена_столбцов FROM имя_таблицы [WHERE ...условия];

Часть оператора с условиями является необязательной (мы рассмотрим ее позже). Оператор SELECT без условий выводит все данные из указанных столбцов.

Извлечем имена (столбец 1_name) и фамилии (столбец 2_name) всех сотрудников из таблицы Sotrudniki.

SELECT 1_name, 2_name from Sotrudniki;

Данные представлены в том порядке, в котором они были введены. Более того, последняя строка указывает число строк в таблице. Чтобы вывести всю таблицу, можно воспользоваться упрощенной формой оператора SELECT: ***SELECT * from Sotrudniki;***

Символ * в этом выражении означает 'ВСЕ столбцы'.

Выборка данных с помощью условий. SELECT имена_столбцов FROM имя_таблицы [WHERE условия];

Операторы сравнения = и != , больше и меньше, <= и >=

В условии могут использоваться операторы сравнения =, !=, <= >=

Выборка столбцов с условием для поля "имя".

SELECT name, 2_name FROM Sotrudniki WHERE name = 'Иван';

Этот оператор выводит имена и фамилии всех сотрудников, которые имеют имя Иван. Отметим, что слово Иван в условии заключено в одиночные кавычки. Можно использовать также двойные кавычки. Кавычки являются обязательными. Кроме того, сравнения MySQL не различают регистр символов, что означает, что с равным успехом можно использовать "Иван", "иван" и даже "ИвАн".

Выборка столбцов с условием для поля "возраст"

SELECT name, last_name FROM Sotrudniki WHERE age=40;

Это список имен и фамилий всех сотрудников с возрастом 40 лет. Тип столбца age задан как int, поэтому кавычки вокруг 40 не требуются.

Оператор != означает 'не равно' и является противоположным оператору равенства.

Используемые в основном с целочисленными данными операторы меньше или равно (<=) и больше или равно (>=) обеспечивают дополнительные возможности.

Поиск текстовых данных по шаблону с помощью предложения WHERE и оператора LIKE.

Как быть, если надо вывести данные о сотрудниках, имя которых начинается с буквы В? Язык SQL позволяет выполнить поиск строковых данных по шаблону. Для этого в предложении **WHERE** используется оператор **LIKE**. В условии вместо знака равенства используется LIKE и знак процент в шаблоне. **Знак % действует как символ-заместитель** (аналогично * в системе DOS). Он заменяет собой любую последовательность символов. Таким образом "В%" обозначает все строки, которые начинаются с буквы В. Аналогично "%В" выбирает строки, которые заканчиваются символом В, а "%В%" строки, которые содержат букву В.

Найдем сотрудников, имя которых начинается с буквы В

SELECT * FROM Sotrudniki WHERE name LIKE "B%"

Выведем сотрудников, которые имеют в названии должности строку "про".

SELECT * FROM Sotrudniki WHERE title LIKE '%про%';

Логические операторы AND, OR, NOT

Можно выбирать данные на основе условий SQL, представленных с помощью булевых (логических) операторов **AND, OR, NOT**

Показан оператор SELECT, который выводит имена сотрудников, которые получают более 7000, но меньше 9000.

SELECT name, last_name FROM Sotrudniki WHERE salary > 7000 AND salary < 9000;

Вывести имена и возраст сотрудников, имена которых начинаются с К или Л, и которые младше 30 лет.

SELECT name, age FROM Sotrudniki WHERE (name like 'К%' OR name like 'Л%') AND age <30;

Скобки предназначены для выделения различных логических условий и удаления двусмысленностей.

Оператор NOT поможет при поиске всех сотрудников, которые не являются программистами (поле title).

SELECT name, last_name FROM Sotrudniki WHERE title NOT LIKE "%программист%";

Операторы IN и BETWEEN

Оператор IN (в множестве) позволяет указать список значений, либо введенный явным образом, либо при помощи подзапроса и сравнить некое значение с этим списком в предложении WHERE или HAVING.

Выведем имена сотрудников, являющихся рабочими или кладовщиками.

SELECT name FROM Sotrudniki WHERE title IN ('рабочий', 'кладовщик');

Использование NOT перед IN позволяет вывести данные, которые не входят в множество, определяемое условием IN.

Оператор **BETWEEN** используется для определения **целочисленных границ**.

Выведем сотрудников, возраст которых от 32-х до 40 лет.

SELECT * FROM Sotrudniki WHERE age BETWEEN 32 AND 40

Вместо age >= 32 AND age <= 40 мы использовали age BETWEEN 32 AND 40. NOT также можно использовать вместе с BETWEEN.

Ключевое слово DISTINCT

Ключевое слово **DISTINCT** (РАЗЛИЧНЫЙ) исключает появление повторяющихся данных.

Выведем все уникальные должности из таблицы.

SELECT DISTINCT title FROM Sotrudniki

Если не использовать DISTINCT, то в списке вывода будут повторяющиеся должности.

Агрегатные функции. Поиск максимального, минимального и среднего значений, нахождение суммы и количества записей.

В MySQL есть 5 агрегатных функций для вычисления минимального, максимального и среднего значений, нахождение суммы и количества записей. **Агрегатные функции выполняют вычисление на наборе значений и возвращают одиночное значение**. Аргументами агрегатных

функций могут быть как столбцы таблиц, так и результаты выражений над ними. Агрегатные функции и сами могут включаться в другие арифметические выражения.

1. **MIN(): минимальное значение**
2. **MAX(): максимальное значение**
3. **SUM(): сумма значений**
4. **AVG(): среднее значений**
5. **COUNT(): подсчитывает число записей**

Рассмотрим, например, поиск минимальной зарплаты

SELECT MIN(salary) from Sotrudniki;

Агрегатная функция **COUNT()** подсчитывает и выводит общее число записей.

Функция COUNT имеет два формата. В первом случае возвращается количество строк входной таблицы, во втором случае — количество значений аргумента во входной таблице:

COUNT(*), COUNT([DISTINCT | ALL] выражение)

Простейший способ использования этой функции — подсчет количества строк в таблице (всех или удовлетворяющих указанному условию). Для этого используется первый вариант синтаксиса.

Во втором варианте синтаксиса функции COUNT в качестве аргумента может быть использовано имя отдельного столбца. В этом случае подсчитывается количество либо всех значений в этом столбце входной таблицы, либо только неповторяющихся (при использовании ключевого слова DISTINCT).

Например, чтобы подсчитать общее число записей в таблице, нужно выполнить следующую команду. Знак * означает "все данные".

SELECT COUNT(*) FROM Sotrudniki;

Именование столбцов. AS

MySQL позволяет задавать имена для выводимых столбцов с помощью оператора **AS**.

Вывод средней зарплаты с использованием псевдо-имен столбцов

SELECT AVG(salary) AS 'Средняя зарплата' FROM Sotrudniki;

Предложения GROUP BY и HAVING

Предложение GROUP BY позволяет группировать аналогичные данные и используется для определения групп выходных строк, к которым могут применяться агрегатные функции (**COUNT**, **MIN**, **MAX, AVG** и **SUM**). Все выходные строки запроса разбиваются на группы, характеризуемые одинаковыми комбинациями значений в столбцах группировки. После чего к каждой группе применяются агрегатные функции. Следует иметь в виду, что для **GROUP BY** все значения **NULL** трактуются как равные, то есть при группировке по полю, содержащему **NULL**-значения, все такие строки попадут в одну группу. Если при наличии предложения **GROUP BY**, в предложении **SELECT** отсутствуют агрегатные функции, то запрос просто вернет по одной строке из каждой группы.

1.Вывести все уникальные должности в таблице:

SELECT title FROM Sotrudniki GROUP BY title;

Аналогичный результат будет при использовании **DISTINCT.**

2.Подсчитать количество сотрудников на каждой должности.

SELECT title, COUNT(*) FROM Sotrudniki GROUP BY title;

В этой команде MySQL сначала создает группы различных должностей, а затем выполняет подсчет в каждой группе.

3.Выведем среднюю зарплату сотрудников для каждой должности.

SELECT title, AVG(salary) FROM Sotrudniki GROUP BY title;

Предложение **HAVING** применяется после группировки для наложения ограничений на групповые значения. Оно необходимо для проверки значений, которые получены с помощью агрегатной функции не из отдельных строк источника записей, определенного в предложении **FROM**, а из групп таких строк. Поэтому такая проверка не может содержаться в предложении **WHERE**.

Выведем только те подразделения, где средняя зарплата более 1000. Выполним это с помощью предложения **HAVING.**

SELECT title, AVG(salary) FROM Sotrudniki GROUP BY title HAVING AVG(salary) > 1000;

Упорядочивание данных. ORDER BY

SQL позволяет сортировать извлеченные данные с помощью предложения ORDER BY на основе какого-либо столбца. Предложение ORDER BY может сортировать в возрастающем порядке (ASCENDING или ASC) или в убывающем порядке (DESCENDING или DESC) в зависимости от указанного аргумента.

Выведем имена сотрудников с упорядоченными по алфавиту фамилиями сотрудников (в возрастающем порядке).

SELECT name, last_name FROM Sotrudniki ORDER BY last_name;

Теперь найдем и выведем число сотрудников, имеющих различные должности, и отсортируем их с помощью ORDER BY, используя псевдо-имя для столбца количество.

SELECT title, COUNT(*)AS Number FROM Sotrudniki GROUP BY title ORDER BY Number;

Ограничение количества извлекаемых данных.

Ограничить число записей, выводимых оператором SELECT. можно с помощью **предложения LIMIT.** Общая форма оператора LIMIT имеет следующий вид:

SELECT (что-то) FROM имя таблицы LIMIT начальная строка, извлекаемое число записей;

LIMIT можно использовать также для **извлечения подмножества данных**, используя дополнительные аргументы.

Выведем три записи, начиная с шестой.

SELECT name FROM Sotrudniki LIMIT 6,3;

Контрольные вопросы

1. Как узнать число строк в таблице с помощью оператора SELECT
2. Что делает оператор SELECT без условий?
3. Какие служебные слова обязательно присутствуют в операторе SELECT?
4. Какой оператор задает имена для выводимых столбцов?
5. Для чего используется оператор BETWEEN?
6. Извлеките 5 записей из таблицы table, начиная с 10 строки?
7. Какой оператор группирует аналогичные данные?
8. Какой оператор позволяет наложить ограничение на группу?
9. Что делает оператор UPDATE без условий?
10. Зачем нужно ключевое слово DISTINCT?

Содержание отчета (в электронном виде загрузить в Moodle)

1. Тема работы
2. Цель работы
3. Выполненное задание с кодом и скриншотами.
4. Ответы на контрольные вопросы

Оформление отчета должно удовлетворять требованиям предъявляемым к курсовым работам (правила оформления курсовых смотрите на сайте факультета)