ЛАБОРАТОРНАЯ РАБОТА №3

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ SQL. SELECT ДЛЯ СУБД MS SQL SERVER.

Цель: сформировать знания и умения по программированию на языке SQL, приобрести практические навыки работы со средствами языка SQL для выборки и редактирования данных в БД.

Содержание лабораторной работы:

1. Изучить теоретические сведения лабораторной работы.
2. Открыть базу данных, созданную в предыдущей лабораторной работе.
3. Создать к базе данных SELECT-запросы следующих видов:
    a. запрос, выбирающий все данные из таблицы;
    b. запрос, выбирающий данные из некоторых столбцов таблицы;
    c. запрос с использованием ключевого слова DISTINCT;
    d. запрос с использованием сортировки данных;
    e. запрос с использованием ограничения на выборку данных;
    f. запрос с использованием предикатов сравнения;
    g. запрос с использованием предиката BETWEEN;
    h. запрос с использованием предиката IN, содержащий подзапрос;
    i. запрос с использованием предиката LIKE и строковых функций;
    j. запрос с использованием предиката IS NULL;
    k. запрос с использованием агрегатных функций;
    l. запрос с использованием агрегатных функций и предложения HAVING;
    m. запрос, выбирающие данные из нескольких таблиц с использованием соединения по предикату;
    n. запрос с использованием предиката EXISTS;
    o. запрос с использованием ключевых слов ANY|ALL;
    p. запрос с использованием функции IF();
4. Выполнить задания по варианту (см. стр. 6).
5. Подготовиться к защитите лабораторной работы.

Краткий вспомогательный материал

SQL — аббревиатура выражения Structured Query Language (язык структурированных запросов). SQL основывается на реляционной алгебре и специально разработан для взаимодействия с реляционными базами данных.

SQL является, информационно-логическим языком, предназначенным для описания хранимых данных, их извлечения и модификации. SQL не является языком программирования. Конкретные реализации языка, как правило, включают различные процедурные расширения.

Язык SQL представляет собой совокупность операторов, которые можно разделить на четыре группы:

- ***DDL*** (Data Definition Language) - операторы определения данных
- ***DML*** (Data Manipulation Language) – операторы манипуляции данными
- ***DCL*** (Data Control Language) - операторы определения доступа к данным
- ***TCL*** (Transaction Control Language) - операторы управления транзакциями

SQL является стандартизированным языком. Стандартный SQL поддерживается комитетом стандартов ANSI (Американский национальный институт стандартов), и соответственно называется ANSI SQL.

Многие разработчики СУБД расширили возможности SQL, введя в язык дополнительные операторы или инструкции. Эти расширения необходимы для выполнения дополнительных функций или для упрощения выполнения определенных операций, но они привязаны к определенной СУБД и редко поддерживаются более чем одним разработчиком. Все крупные СУБД и даже те, у которых есть собственные расширения, поддерживают ANSI SQL (в большей или меньшей степени). Отдельные же реализации носят собственные имена (PL-SQL, Transact-SQL и т.д.). Transact-SQL (T-SQL) – реализация языка SQL корпорации Microsoft, используемая, в частности, и в SQL Server.

Оператор SELECT.

Для выборки данных используется команда:

```
SELECT [ ALL | DISTINCT ]
    [TOP ( выражение ) [PERCENT]]
    < список полей >
FROM <таблица> [ , <таблица2>...n ] ]
    [ WHERE <условие> ]
    [ GROUP BY <поле> | <Integer> [,...]]
    [ HAVING <условие>]
    [ ORDER BY <поле> | <Integer> [ ASC|DESC ]  [,...]]
```

Пример. Выбрать все сведения о проектах.

```
SELECT * FROM projects;
SELECT id, project_name FROM projects; /* будут выведены два поля */
```

Рассмотрим отдельные элементы синтаксиса инструкции SELECT.

ALL - указывает на то, что в результирующем наборе могут появляться повторяющиеся элементы. ALL является значением по умолчанию.

DISTINCT - в результирующем наборе возвращаются только уникальные результаты.

ORDER BY - сортировка строк результирующей таблицы данных

```
ORDER BY { <поле> | <Integer> [ ASC|DESC ] } [,...]
```

Если ORDER BY используется внутри GROUP BY, то строки сортируются внутри каждой группы результирующих строк. Вместо имен полей могут быть использованы их порядковые номера в списке полей результирующей таблицы. ASC сортирует данные в восходящем порядке, DESC – в обратном.

Пример. Выбрать сведения о проектах, отсортировав их по названию проекта.

```
SELECT id, project_name FROM projects
ORDER BY project;
```

TOP - ограничение количества выбираемых записей в запросе

```
TOP <выражение> [percent]
```

выражение - количество записей.

Percent – наличие этого слова приведет к выводу указанного в выражении процента от общего количества записей.

Пример. Извлечь из выборки последние пять записей о проектах.

```
SELECT TOP 5 id, project_name
FROM projects ORDER BY id DESC
```

WHERE - ограничение выборки данных из указанных таблиц

```
WHERE <условие>
```

Пример. Выбрать сведения о сотрудниках, работающих в первом отделе:

```
SELECT  *
FROM employees WHERE id_depart=1;
```

BETWEEN - проверяет, попадают ли значения проверяемого выражения в диапазон, задаваемый пограничными выражениями, соединяемыми служебным словом AND.

```
[NOT]  BETWEEN  <Начальное  выражение>  AND <Конечное выражение>
```

Пример. Выбрать сведения о сотрудниках из первого, второго и третьего отделов.

```
SELECT  * FROM employees
WHERE id_depart BETWEEN 1 AND 3;
```

IN - проверяет, попадают ли значения проверяемого выражения во множество:

```
<Проверяемое_выражение> [NOT] IN (<подзапрос>)|(<выражение_для_вычисления_значе
ния>,...)
```

Пример. Выбрать сведения о сотрудниках из первого и третьего отделов.

```
SELECT  * FROM employees WHERE id_depar IN(1,3);
```

LIKE - сравнение строк с шаблоном

```
<проверяемое_значение> [NOT] LIKE <шаблон> [ESCAPE <символ>]
```

Пример. Найти все проекты, названия которых начинаются с буквы А.

```
SELECT * FROM project WHERE project LIKE 'A%';
```

Стандартом предусмотрены следующие строковые функции.

```
CONCAT()       Объединение строк
LOWER(A)       Приведение А к нижнему регистру
LEFT()         Возвращает строку символов указанной длины, отсчитывая слева
LEN()       Длина строки
SUBSTRING(A,B,C) Возвращает подстроку из А, с позиции В до позиции С
LTRIM(str)     Удаляет все начальные пробелы из строки str
RTRIM(str)     Удаляет хвостовые пробелы из строки str
REPLACE(A,B,C) Заменяет все подстроки В в строке А на подстроку С
STRCMP()       Возвращает 0, если строки одинаковые
UPPER(A)       Переводит А в верхний регистр
```

Пример. Вывести названия проектов в верхнем регистре.

```
SELECT UCASE(project_name) FROM projects;
```

IS [NOT] NULL - позволяет проверить отсутствие (наличие) значения в полях

Пример. Выбрать сведения о несданных заданиях:

```
SELECT * FROM tasks WHERE date_turn IS NULL;
```

GROUP BY - объединяет результат запроса в группы

```
GROUP BY <поле> | <Integer> [,...]
```

Пример. Посчитать, сколько заданий в каждом проекте.

```
SELECT id_project, COUNT(*) AS num_tasks /* AS   используется для
                                        назначения  псевдонима  столбцу */
FROM tasks GROUP BY id_project ;
```

Стандартом предусмотрены следующие агрегатные функции:

```
COUNT(*)        Возвращает количество строк источника записей.
COUNT(<имя поля>) Возвращает количество значений в указанном столбце.
```

```
SUM(<имя_поля>) Возвращает сумму значений в указанном столбце.
AVG(<имя_поля>) Возвращает среднее значение в указанном столбце.
MIN(<имя_поля>) Возвращает минимальное значение в указанном столбце.
MAX(<имя_поля>) Возвращает максимальное значение в указанном столбце.
```

HAVING - используется вместе с GROUP, для того чтобы выбирать только определенные группы строк данных, которые удовлетворяют указанному условию.

```
HAVING <условие>
```

Пример. Вывести сведения о проектах, в которых больше трех заданий.

```
SELECT id_project FROM tasks GROUP BY id_project
HAVING COUNT(*)>3;
```

JOIN - часто для решения задач необходимо выбирать данные, находящиеся в разных, связанных логически между собой таблицах. Синтаксис соединения таблиц по предикату имеет вид:

```
FROM <таблица1> [AS <псевдоним_табл1>] [INNER | LEFT | RIGHT | FULL] JOIN <табл
ица2> [AS <псевдоним_табл2>] [ON <предикат>]
```

Пример. Вывести номер телефона сотрудника Василькова.

```
SELECT   d.phone
FROM employees JOIN departments as d
ON(employees.id_depart = d.id_depart) WHERE   employees.surname="Васильков";
```

UNION - объединяет выходные строки каждого из запросов в один результирующий набор.

```
<запрос1> UNION [ALL] <запрос2>;
```

Пример. Вывести время выдачи зарплаты сотрдников первого отдела и работающих над проектом «P1»

```
SELECT id_employee, salary_hour FROM employees WHERE id_depart=1
UNION
SELECT distinct id_employee, salary_hour
FROM employees as e join tasks as t on (e.id_employee=t.id_empl) JOIN
projects ON (t.id_project=projects.id_project) WHERE project= "P1"
ORDER BY salary_hour;
```

EXISTS - принимает значение TRUE, если подзапрос возвращает любое количество строк, иначе его значение равно FALSE.

```
EXISTS <подзапрос1>
```

Пример. Выбрать проекты, которые не выполняются в данный момент.

```
SELECT id_project FROM projects WHERE NOT EXISTS
      (SELECT id FROM tasks
       WHERE id_project= projects.id_project)
```

ANY(SOME) - проверяемое значение поочередно сравнивается с каждым элементом, который возвращает вложенный запрос. Если хотя бы одно из сравнений истинно, то строчка выводится на экран.

```
<выражение> <оператор_сравнения> ALL|ANY(SOME) (<подзапрос>)
```

Пример. Извлечь из списка сотрудников, сдавших хотя бы одно задание в этом месяце.

```
SELECT   id_employee, surname, id_depart FROM employees as e
WHERE id_employee = ANY (SELECT id_empl FROM tasks WHERE
```

```
                    month(date_turn)=month(NOW())and id_empl=e.id_employee );
```

ALL - проверяемое значение поочередно сравнивается с каждым элементом, который возвращает вложенный запрос, но строка выводится на экран, когда все сравнения истинны.

Пример. Вывести фамилии сотрудников, которые не сдавали задания в текущем месяце.

```
SELECT  id_employee, surname, id_depart FROM employees as e WHERE
id_employee != ALL (SELECT id_empl FROM tasks WHERE
                    month(date_turn)=month(NOW())and id_empl=e.id_employee );
```

Функция IIF() - функция предназначена для логического выбора. Если expr1 истинно, то функция возвращает expr2, иначе expr3.

```
IIF(expr1, expr2, expr3);
```

Пример. Указать занятость сотрудников

```
SELECT id_employee, iif(count(tasks.id)<1,"free", "busy") as busyness
FROM employees left join tasks on(employees.id_employee=tasks.id_empl)
WHERE date_turn is NULL GROUP BY id_employee;
```

Результату выполнения функции iif() в запросе присвоится новый псевдоним busyness.

Вариант	Задание
1-5	1. Из таблицы ORЕDERS выбрать заказы со сроком даты заказа более ранним чем 20-03-2018. Список отсортировать по номеру заказа в обратном порядке.
	2. Получить информацию о покупателях (фамилия, имя, адрес, телефон, город), которые совершили заказ со статусом «оплата». Список отсортировать по городу и фамилиям.
	3. Выбрать все сведения о покупателях из первой и третьей компании.
	4. Получить информацию о количестве покупателей, оплативших заказ в каждом штате.
6-10	1. Получить информацию о заказе: id заказа, фамилию, имя, адрес, дата заказа, дата доставки. Список отсортировать по дате доставки в обратном порядке
	2. Получить информацию о покупателях (фамилия, имя, адрес, телефон), которые совершили заказ со статусом «оплата». Список отсортировать по фамилиям.
	3. Выбрать всю информацию о покупателях, которые совершили заказы.
	4. Подсчитать количество покупателей в каждом штате, которые не сделали ни одного заказа
11-15	1. Получить список заказов от компании «НАЗВАНИЕ КОМПАНИИ». Список отсортировать по дате заказа.
	2. Получить информацию о покупателях (фамилия, имя, адрес, телефон, город), которые совершили заказ со статусом «в кредит». Список отсортировать по городу и фамилиям
	3. Выбрать всю информацию о покупателях, которые не совершили ни одного заказа.
	4. Подсчитать сумму сделок каждого покупателя из определенного штата
16-20	1. Получить информацию о покупателях, которые не сделали ни одного заказа. Список отсортировать по фамилии.
	2. Получить информацию о покупателях (фамилия, имя, адрес, телефон), которые совершили заказ со статусом «оплата». Список сгруппировать по штату и отсортировать по фамилиям
	3. Вывести список товаров (ID, описание, количество, сумма, дата доставки), которые были заказаны в заказах под номерами 5 и 12.
	4. Подсчитать количество каждого товара в заказах в определенном штате
21-25	1. Получить список заказов, фамилии, телефоны и адреса покупателей, которые совершили заказ с 11-01-2017 по 20-03-2017. Список отсортировать по дате заказа.
	2. Получить информацию о покупателе (фамилия, адрес, телефон, город, дата заказа) с максимальной суммой заказа.
	3. Получить сгруппированный по штату список с информацией (№заказа, дата заказа, дата доставки). Отсортировать список по дате доставки.
	4. Подсчитать количество заказов по штатам

26-30	1. Получить информацию о товарах, которые находятся на складе и цена которых от 90 до 1000. Список отсортировать по цене.
	2. Получить информацию о покупателе (фамилия, адрес, телефон, город, дата заказа) с минимальной суммой заказа.
	3. Получить сгруппированный по штату список с информацией (№заказа, дата заказа, дата доставки), причем покупатели этих заказов сделали больше 3 заказов.
	4. Подсчитать среднюю цену заказов по штатам

Контрольные вопросы

1. Что такое SQL? Предназначение?
2. Какие существуют группы операторов в языке SQL?
3. Каково назначение команды SELECT?
4. Опишите структуру команды SELECT.
5. Как осуществляется сортировка? Использование сортировки вместе с группировкой. Восходящий и обратный порядок сортировки.
6. Как осуществляется ограничение количества выбираемых записей?
7. Как реализуется ограничение выбора данных с помощью конструкции WHERE и предикатов BETWEEN, IN, LIKE, IS?
8. Какие строковые функции предусмотрены стандартом? Их назначение.
9. Как осуществляется группировка данных?
10. Какие агрегатные функции предусмотрены стандартом? Их назначение.
11. Как применяется конструкция HAVING BY?
12. Опишите синтаксис соединения таблиц по предикату.
13. Объясните разлдичия между INNER JOIN, LEFT JOIN, RIGHT JOIN.
14. Опишите структуру и принцип работы оператора UNION.
15. Каково назначение предиката EXIST?
16. Опишите структуру и назначение команд ALL, ANY, SOME.
17. Каково назначении функции IF?