**Лабораторная работа №1**

**Тема:** Выполнение операций над данными с использованием операторов языка SQL.

**Цель:** научиться использовать операторы языка SQL для работы с данными БД.

**Теоретический базис**

**Описание языка**

**Язык SQL** – стандартный язык запросов по работе с реляционными базами данных. Язык SQL появился после реляционной алгебры, и его прототип был разработан в конце 70-х годов в компании IBM Research. В силу своего широкого распространения постепенно стал стандартом «де-факто» для языков манипулирования данными в реляционной СУБД.

Язык SQL предназначен для выполнения операций над таблицами (создание, удаление, изменение структуры) и над данными таблиц (выборка, изменение, добавление и удаление), а также некоторых сопутствующих операций. SQL является непроцедурным языком и не содержит операторов управления, организации подпрограмм, ввода-вывода и т.п. В связи с этим SQL автономно не используется, обычно он погружен в среду встроенного языка программирования СУБД.

Основным назначением языка SQL (как и других языков для работы с базами данных) является подготовка и выполнение запросов. В результате выборки данных из одной или нескольких таблиц может быть получено множество записей, называемое представлением.

**Оператор SELECT**

Язык запросов в SQL состоит из единственного оператора – SELECT. Синтаксис оператора SELECT имеет следующий вид:

*SELECT [ ALL| DISTINCT] <Список полей>|\**

*FROM <Список таблиц>*

*[WHERE <Предикат-условие выборки или соединения>]*

*[GROUP BY <Список полей результата>]*

*[HAVING <Предикат-условие для группы>]*

*[ORDER BY <Список полей, по которым упорядочить вывод>];*

**SELECT** – ключевое слово, которое сообщает СУБД, что эта команда – запрос. Все запросы начинаются этим словом с последующим пробелом. За ним может следовать способ выборки.

Ключевое слово **ALL** означает, что в результирующий набор строк включаются все строки, удовлетворяющие условиям запроса.

Ключевое слово **DISTINCT** означает, что в результирующий набор включаются только различные строки, то есть дубликаты строк результата не включаются в набор.

В разделе **FROM** задается перечень исходных отношений (таблиц) запроса.

**Функции MS Access:**

***Ucase*** *(выражение)* – переводит значения поля «выражение» в верхний регистр;

***Mid*** *(строка, начало\_поиска[, длина])* – возвращает строку из поля «строка», начиная с позиции «начало\_поиска» длинной – «длина». Если длина не указана, то до конца строки.

***Round*** *(выражение [,количество\_десятичных \_знаков])* – округляет «выражение» с точностью «количество\_десятичных\_знаков». Если второй аргумент не указан, то округляет до целого.

***IIF*** *(выражение, truepart, falsepart)* – возвращает одно из значений: truepart, если выражение истинно и falsepart – если ложно.

В разделе **WHERE** задаются условия отбора строк результата или условия соединения кортежей исходных таблиц, подобно операции условного соединения в реляционной алгебре. В выражении условий раздела **WHERE** могут быть использованы следующие предикаты:

**Предикаты сравнения** (=, <>, >, >=, <, <=), которые имеют традиционный смысл.

**Предикат Between A and B** – принимает значения между А и В. Предикат истинен, когда сравниваемое значение попадает в заданный диапазон, включая границы диапазона. Одновременно в стандарте задан и противоположный предикат Not Between A and B, который истинен тогда, когда сравниваемое значение не попадает в заданный интервал, включая его границы.

**Предикат вхождения в множество IN** (множество) истинен тогда, когда сравниваемое значение входит в множество заданных значений. При этом множество значений может быть задано простым перечислением или встроенным подзапросом. Одновременно существует противоположный предикат NOT IN (множество), который истинен тогда, когда сравниваемое значение не входит в заданное множество.

**Предикаты сравнения с образцом LIKE и NOT LIKE.** Предикат LIKE требует задания шаблона, с которым сравнивается заданное значение, предикат истинен, если сравниваемое значение соответствует шаблону, и ложен в противном случае. Предикат NOT LIKE имеет противоположный смысл. Шаблон может содержать % (\* для Access) для обозначения любого числа любых символов; \_ (? для Access) для обозначения любого одного символа.

**Предикат сравнения с неопределенным значением IS NULL.** Для выявления равенства значения некоторого атрибута неопределенному значению применяют специальные стандартные предикаты: <имя атрибута> IS NULL и <имя атрибута> IS NOT NULL.

**Предикат существования EXIST и не существования NOT EXIST.** Применяется во вложенных запросах для определения непустого или пустого множества, являющегося результатом выборки.

В разделе GROUP BY задается список полей группировки. GROUP BY группирует записи данных и объединяет в одну запись все записи данных, которые содержат идентичные значения в указанном поле (или полях).

В разделе HAVING задаются предикаты-условия, накладываемые на каждую группу. HAVING используется для фильтрации записей, полученных в результате группировки.

**Применение агрегатных функций**

|  |  |
| --- | --- |
| **Функция** | **Результат** |
| COUNT | Количество строк или непустых значений полей, которые выбрал запрос |
| SUM | Сумма всех выбранных значений данного поля |
| AVG | Среднеарифметическое значение всех выбранных значений данного поля |
| MIN | Наименьшее из всех выбранных значений данного поля |
| MAX | Наибольшее из всех выбранных значений данного поля |

**Применение объединения (JOIN)**

Внутреннее объединение (INNER JOIN) возвращает записи из двух таблиц, если значение первичного ключа первой таблицы соответствует значению внешнего ключа второй таблицы, связанной с первой.

**Перекрестные запросы и подзапросы**

**Перекрестный запрос** – способ группировки данных по двум измерениям, позволяющий отображать итоги в компактном результирующем наборе. В перекрестном запросе группировка выполняется по одному полю, а итоговая функция применяется к другому полю. Структура перекрестного запроса следующая: в конструкции **TRANSFORM** указывается поле и групповая функция, применяемая к нему, данное поле выводится на пересечении строк и столбцов; в конструкции **GROUP** **BY** указывается поле, по которому проводится группировка и которое выводится в качестве заголовков строк; в конструкции **PIVOT** указывается поле, значения которого выводятся в качестве заголовков столбцов.

**Задание на лабораторную работу**

**Часть I Простые запросы с параметром**

1. Написать запрос для выбора автомобилей определенного цвета. Цвет задается в виде параметра в условии WHERE (например, ‘белый’).

SELECT AUTO.\*

FROM AUTO

WHERE COLOR LIKE 'Blue';

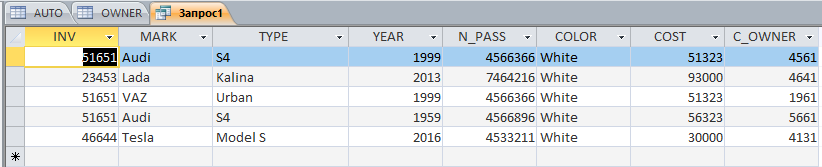


Рис. 1. Результат запроса\_1\_1

**Часть II Использование агрегирования и подзапросов**

1. Определить количество автомобилей, у которых номер тех. паспорта начинается на "1" и не заканчивается на "4".

SELECT AUTO.\*

FROM AUTO

WHERE MID (INV, 1, 2) LIKE '46' AND MID (INV, 4, 2) NOT LIKE '55';

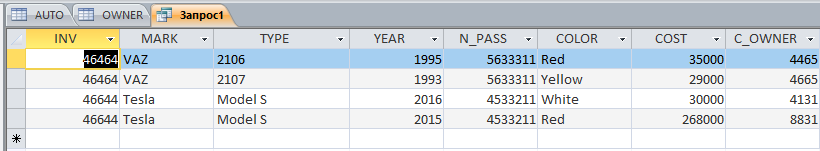
**

Рис. 2. Результат запроса\_2\_1

2. По каждой штатной группе а/м определить, сколько а/м каждой марки было выпущено в заданном году. Вывести названия групп и названия марок на экран.

SELECT AUTO.MARK, AUTO.TYPE, AUTO.YEAR, Count(AUTO.MARK) AS QUANTITY

FROM AUTO

WHERE (((AUTO.YEAR)=2013))

GROUP BY AUTO.MARK, AUTO.TYPE, AUTO.YEAR;

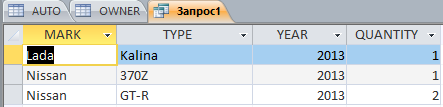


Рис. 3. Результат запроса\_2\_2

3. Определить, какие а/м данного класса переданы в подразделения после указанной даты. Указать также номер автомобиля и дату документа передачи.

SELECT OWNER.C\_OWNER, OWNER.D\_PASS

FROM OWNER

WHERE OWNER.D\_PASS>#1/5/2014#

GROUP BY OWNER.C\_OWNER, OWNER.D\_PASS;

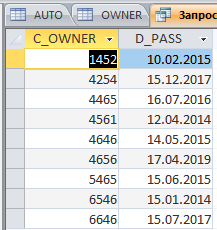


Рис. 4. Результат запроса\_2\_3

**Часть III Использование объединений**

1. Произвести выборку автомобилей из двух полей «номер авто», «класс авто» (подставлять название из отношения MENU). Если поле «класс» в таблице MENU не существует, то выводить строку «Класс средства неизвестен» с помощью функции iif.

SELECT AUTO.MARK, AUTO.TYPE, OWNER.OWNER\_SN, OWNER.OWNER\_FN

FROM OWNER INNER JOIN AUTO

ON OWNER.C\_OWNER=AUTO.C\_OWNER;

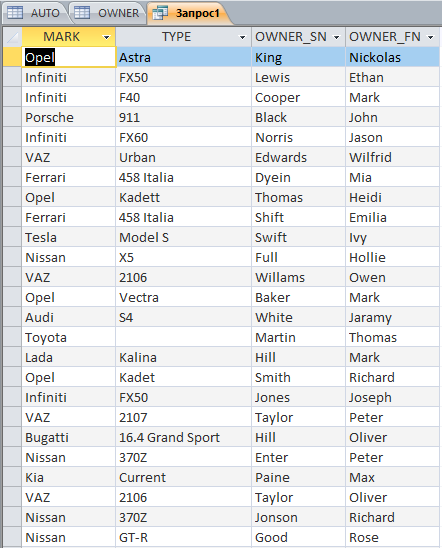


Рис. 5. Результат запроса\_3\_1

**Часть IV Использование перекрестных запросов и подзапросов**

1. Определить, сколько а/м каждой марки имеют год выпуска меньший, чем округленный до целого средний год выпуска а/м заданной пользователем марки.

SELECT AUTO.MARK, Count(AUTO.MARK) AS QUANTITY

FROM AUTO

WHERE AUTO.YEAR < (

SELECT Round(Avg(AUTO.YEAR))

FROM AUTO

WHERE AUTO.MARK LIKE 'Tesla')

GROUP BY AUTO.MARK;

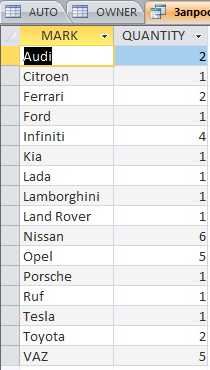


Рис. 6. Результат запроса\_4\_1

2. Определить какое количество а/м каждой марки в каком году было произведено (перекрестный запрос: марки а/м на год производства).

TRANSFORM COUNT(AUTO.MARK)

SELECT AUTO.MARK

FROM AUTO

GROUP BY AUTO.MARK

PIVOT YEAR;

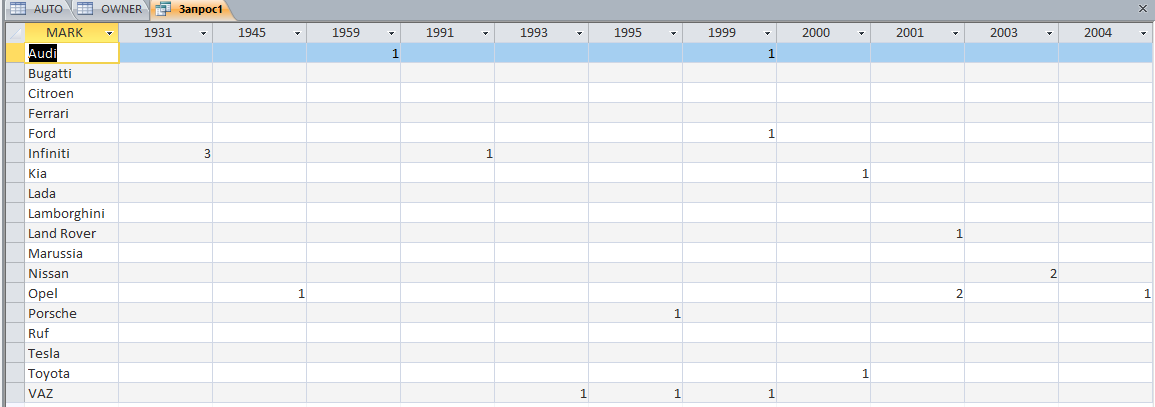


Рис. 7. Результат запроса\_4\_2

**Вывод.** Выполнив лабораторную работу я освоил основные функции языка SQL для работы с Access.