**Лабораторная работа №2. Оператор DML SELECT. Простые запросы. Простые запросы с группировкой. Агрегатные функции**

**Краткие теоретические данные.**

**Простые запросы**

Для выполнения запросов в SQL существует один единственный оператор – 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. За ним может следовать способ выборки:

- **ALL -** ключевое слово, означает, что в результирующий набор строк включаются **все строки** (не столбцы), удовлетворяющие условиям запроса, т.е. в результирующий набор могут попасть одинаковые строки.

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

- список полей – это список перечисленных через запятую столбцов, которые выбираются запросом из таблиц.

- символ **\*** (звездочка) означает, что в результирующий набор включаются все столбцы из исходных таблиц запроса.

В разделе **FROM** задается перечень исходных отношений (таблиц) запроса. В случае, если указано более одного имени таблицы, неявно подразумевается, что над перечисленными таблицами осуществляется операция **декартова произведения**. Разделы SELECT и FROM являются обязательными, все другие разделы являются необязательными.

Допустимо в раздел SELECT включать не только имена полей и \*.

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 со строками. Для работы со строками могут использоваться встроенные функции, например:

DATALENGTH(<**имя атрибута>**) - возвращает длину строки в байтах;

LEN(<**имя атрибута>**) - возвращает длину строки символах;

- **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**

**- Предикаты с временными значениями**

В СУБД реализовано достаточно большое количество функций для работы с данными в формате времени и даты. Наиболее часто используемые приведены ниже:

**DAY** ( *date* )  -  целочисленное представление дня указанной даты. Эта функция эквивалентна функции **DATEPART**(*dd*, *date*).

**MONTH** ( *date* )   -  целочисленное представление месяца указанной даты. Эта функция эквивалентна функции **DATEPART**(*mm*, *date*).

**YEAR** ( *date* )   -  целочисленное представление года указанной даты. Эта функция эквивалентна функции **DATEPART**(*yy*, *date*).

**YEAR(GETDATE()) *–*** возвращает текущее значение даты и времени

**DATEDIFF (year , START\_DATE , END\_DATE) -** возвращает разницу во времени (**year - в годах, day-в днях**)

Функции даты и времени могут использоваться в выражениях SELECT при вычислении значений столбцов, в условии выборки сток, сортировки, группировки, а также условиях выборки групп.

|  |  |  |  |
| --- | --- | --- | --- |
| **tab\_1** | | | |
| **Id** | **Department** | **Name** | **Birthday** |
| 1 | Продаж | Иванов | 06-12-1985 |
| 2 | Продаж | Петрова | 04-23-1976 |
| 3 | Кадров | Сидоров | 08-27-1976 |
| 4 | Кадров | Кунин | 05-13-1983 |
| 5 | Технический | Мартынов | 11-11-1984 |

*SELECT department, name, DATEDIFF (year , Birthday , CONVERT (date,GETDATE( ))) as age*

*FROM tab\_1;*

*- возвращает выборку с названием отдела, фамилией сотрудника, возрастом сотрудника*

*SELECT department, name, birthday*

*FROM tab\_1*

*WHERE DATEDIFF (year , Birthday , CONVERT (date,GETDATE( ))) >40 ORDER BY birthday;*

*- возвращает выборку, содержащую сведения о сотрудниках старше 40 лет отсортированную по дате рождения*

*SELECT department, COUNT(\*)*

*FROM tab\_1*

*GROUP BY department*

*HAVING MAX(DATEDIFF (year , Birthday , CONVERT (date,GETDATE( ))))<40;*

*- возвращает выборку с перечнем отделов, сотрудники которых младше 40 лет;*

В части **ORDER BY** задается список полей упорядочения результата, то  
есть список полей, который определяет порядок сортировки в  
результирующем отношении. Например, если первым полем списка будет  
указан Шифр группы, а вторым Фамилия, то в результирующем отношении  
записи сначала будут расположены в порядке возрастания шифра группы, а  
затем в рамках одной группы записи будут отсортированы по фамилии в  
алфавитном порядке. Также можно определить направление порядка: по убыванию **ASC** – сортировка по возрастанию (стоит по умолчанию), **DESC** – сортировка по убыванию

**Запросы с группировкой**

Для выполнения запросов в SQL выполняющих поиск не отдельных строк, а объединение (*группировку*) строк по некоторым признакам и выборку таких 'объединений' (групп) согласно условиям также используется оператор SELECT с ***GROUP BY и HAVING*** . Полный 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>];*

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). **WHERE** определяет, какие записи должны участвовать в группировании, т.е. фильтрует до группирования

**Пример**

|  |  |  |  |
| --- | --- | --- | --- |
| **tab\_1** | | | |
| **Id** | **Department** | **Name** | **Birthday** |
| 1 | Продаж | Иванов | 06-12-1985 |
| 2 | Продаж | Петрова | 04-23-1976 |
| 3 | Кадров | Сидоров | 08-27-1976 |
| 4 | Кадров | Кунин | 05-13-1983 |
| 5 | Технический | Мартынов | 11-11-1984 |

*SELECT department*

*FROM tab\_1*

*GROUP BY department;*

*- возвращает выборку с названием отделов*

|  |
| --- |
| **Department** |
| Продаж |
| Кадров |
| Технический |

**Кроме того, группировка позволяет применять агрегатные функций.** В SQL добавлены дополнительные функции, которые позволяют вычислять обобщенные групповые значения -- *агрегатные функции*. Для применения агрегатных функций предполагается предварительная операция группировки. При отсутствии группировки вся выборка является одной группой.

Некоторые агрегатные функции представлены в таблице 3.1.

**Таблица 3.1**

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

Агрегатные функции могут быть использованы как вычисляемый столбец

*SELECT Сount(\*) FROM <table1>;*

*- возвращает количество всех строк*

*SELECT < поле1>, MAX(< поле2>) as maximum FROM <table1> GROUP BY < поле1>;*

*- в результате запроса поля таблицы будут сгруппированы по полю 1 и в каждой группе найдено максимальное значение поля 2*

**ПРИМЕР**

*SELECT department, count(id) FROM tab\_1 GROUP BY department;*

*- возвращает выборку с названием отделов*

|  |  |
| --- | --- |
| **Department** |  |
| Продаж | 2 |
| Кадров | 2 |
| Технический | 1 |

Ключевое слово**DISTINCT** возможно использовать совместно с НЕКОТОРЫМИ агрегатными функциями, например с **COUNT()**

*SELECT COUNT(DISTINCT Department) FROM tab\_1;*

*- возвращает количество отделов компании.*

!!!! При наличии в запросе группировки (GROUP BY) перечень столбцов результирующей выборки может содержать только те столбцы, по которым выполняется группировка, а также столбцы к которым применена одна из агрегативных функций.

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

|  |
| --- |
| !!! WHERE фильтрует до группирования.  !!! HAVING определяет, какие из получившихся в результате группировки записей будут включены в результирующую выборку, т.е. фильтрует записи после группирования. |

**Предикаты-условия могут быть такими же как и в разделе WHERE:**

**- Предикаты сравнения (=, <>, >, >=, <, <=)**.

**- Предикат Between A and B**

- **Предикат вхождения в множество IN**

и прочее.

**ПРИМЕР**

*SELECT department, count(id) FROM tab\_1 GROUP BY department HAVING count(id)>1;*

|  |  |
| --- | --- |
| **Department** |  |
| Продаж | 2 |
| Кадров | 2 |

Тип NULL, его влияние на значения агрегатных функции

 Тип NULL особым образом обрабатывается как в условиях оператора SELECT, так и агрегатными функциями.

Для типа **AVG()**, **MIN()**, **MAX()**, **SUM()**, **COUNT()**, если значение столбца будет содержать **NULL то такая строка игнорируется**

Если какие либо из значений содержащихся в столбце, равны **NULL**, при вычислении результата функции они исключаются!

Если все значения в столбце равны **NULL**, то функции **AVG()**, **SUM()**, **MIN()**, **MAX()** возвращают значения **NULL.** Функция **COUNT()** возвращает ноль.

Если в столбце нет значений (т.е. столбец пуст), то функции **AVG()**, **SUM()**, **MIN()**, **MAX()** возвращают значения **NULL.** Функция **COUNT()** возвращает ноль!

Функция **COUNT(\*)** подсчитывает количество строк и не зависит от наличия или отсутствия в столбце значений **NULL.** Если строк в столбце нет, то эта функция возвращает ноль

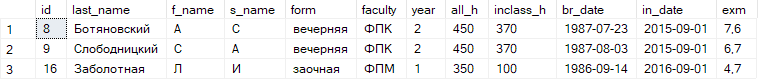
**SUM(значение столбца - NULL)** возвращает **NULL.**

**Задание к лабораторной работе №2**

!!! Выполнять по БД UNIVER

1. Вывести фамилии студентов, в которых присутствуют буквы "б" и/или "о" в любых комбинациях и порядке

select \* from stud where last\_name like '%б%' or last\_name like '%о%'



1. Вывести сведения о студентах, фамилии которых начинаются с буквы К, при этом они являются иностранцами (NULL в поле отчества)

Select \* from stud where last\_name like 'К%' And s\_name is null



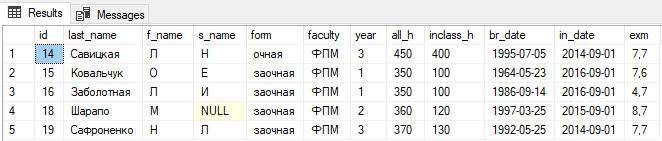
1. Вывести сведения о студентах, фамилии которых содержат 8 символов и более

select \* from stud where last\_name like '\_\_\_\_\_\_\_\_%'



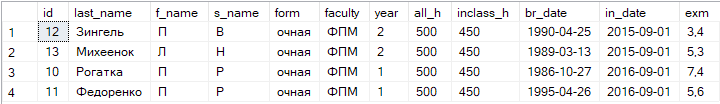
1. Вывести сведения о студентах, фамилии которых содержат или больше, или меньше семи символов (но не ровно семь символов), при этом одна из букв фамилии 'a'

Select \* from stud where last\_name like '%а%' AND last\_name not like '\_\_\_\_\_\_\_'



1. Найти всех студентов учащихся на ФПМ очном 1 или 2 курсе, отсортировать результаты по полю отчества

select \* from stud where faculty='ФПМ' and form='очная' and (year = '1' or year = '2') order by s\_name



1. Найти всех студентов учащихся на ФПК заочном со средним балом успеваемости больше 6, отсортировать результаты по оценки в убывающем порядке

Select \* from stud where faculty='ФПК' AND form = 'заочная' and exm>(select avg(exm) from stud)



1. Вывеси список всех преподавателей, которые работают на ФПК (не зависимо от формы обучения) отсортировать по алфавиту фамилии в пределах каждой формы обучения

select last\_name,form from teach where faculty='ФПК' group by form,last\_name order by form, last\_name



1. Вывести список всех преподавателей, которые работают на ФПМ, первом курсе и читают дисциплины более 100 часов.

Select \* from teach where faculty='ФПМ' and year=1 and hours>100



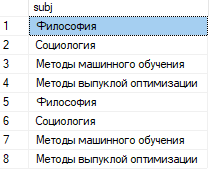
1. Вывести список преподавателей иностранцев, работающих в университете более трех лет на текущий момент

select \* from teach where s\_name is null and (DATEDIFF (year,YEAR(GETDATE()),start\_work\_date) > 3)



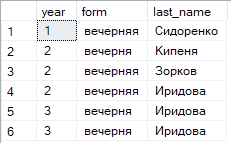
1. Вывести информацию о дисциплинах, читаемых для студентов третьего курса ФПМ

Select subj from teach where faculty='ФПМ' and year=3



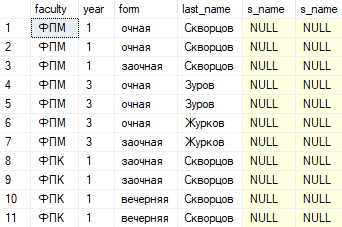
1. Вывести информацию о дисциплинах (курс, форма обучения, Фио преподавателя), читаемых на ФПК, число часов по которым больше 100.

select year, form,last\_name from teach where faculty='ФПК' and (hours > 100)



1. Вывести информацию о дисциплинах (факультет, курс, форма обучения, Фио преподавателя), которые ведут преподаватели иностранцы.

Select faculty, year, form, last\_name, s\_name, s\_name from teach where s\_name is null



1. Вывести список преподавателей старше 30 лет на начало текущего года

select DISTINCT last\_name,br\_date,(year(getdate()) - year(br\_date)) as age from teach where ((year(getdate()) - year(br\_date)) > 30)



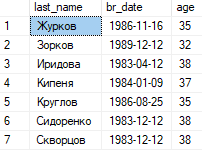
1. Вывести список преподавателей, от 35 до 40 лет на настоящий момент, отсортировать их по алфавиту фамилии.

select distinct last\_name, br\_date, YEAR(getdate()) - YEAR(br\_date) as age from teach

where YEAR(getdate()) - YEAR(br\_date) > 30

And YEAR(getdate()) - YEAR(br\_date) < 40

order by last\_name



1. Вывести список преподавателей, день рождения у которых в октябре, Вывести в порядке возрастания даты рождения.

select DISTINCT last\_name,br\_date from teach where (MONTH(br\_date) = 10) order by br\_date asc



1. Определить среднее значение средних баллов (то значение, которое в таблице) за экзамены для студентов заочной форме обучения каждого факультета.

select faculty, avg(exm) as avg\_exm from stud where form='заочная' group by faculty



1. Найти значения максимальных средних баллов для каждого факультета с учетом курса

select faculty,year, max(exm) from stud group by faculty, year order by faculty



1. Определить список факультетов, среднее значение средних баллов успеваемости на которых больше 7

select faculty, avg(exm) as avg\_exm from stud

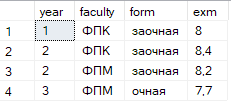
group by faculty

Having avg(exm)>7



1. Определить список курсов с указанием факультета и формы обучения, средний балл успеваемости на которых больше 7.5

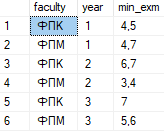
select year,faculty,form,avg(exm) as exm from stud group by year,faculty,form Having (avg(exm)>7.5)



1. Определить минимальное значение средних баллов успеваемости на каждом факультете с привязкой к курсу

select faculty,year, min(exm) as min\_exm from stud

group by faculty , year



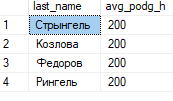
1. Определить перечень факультетов с привязкой к форме минимальное значение средних баллов успеваемости для которых больше 6

select faculty,form from stud group by faculty,form having min(exm) > 6



1. Определить, какое количество часов один студент ФПК 3-го курса заочной формы обучения тратит на самостоятельную подготовку

select last\_name, (all\_h + inclass\_h)/2 as avg\_podg\_h from stud where form like 'заочная' and faculty like 'ФПК'



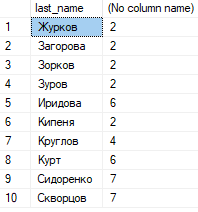
1. Вывести список факультетов, курсов, форм обучения, учась на которых студент тратит более 150 часов на самостоятельную подготовку

select faculty,year,form from stud where inclass\_h > 150



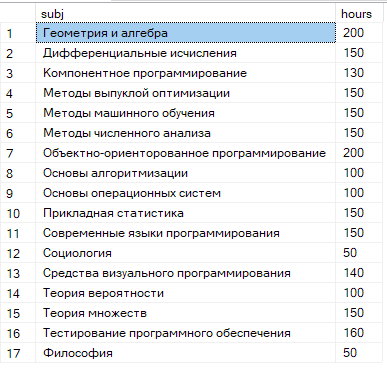
1. Вывести количество предметов, читаемых каждым преподавателем

select last\_name, COUNT(subj) from teach group by last\_name



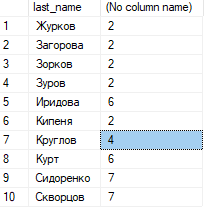
1. Вывести список предметов с указанием числа часов являющихся максимальным для данного предмета

select subj,max(hours) as hours from teach group by subj



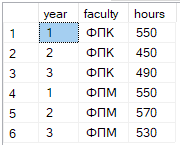
1. Вывести список преподавателей, читающих более одного предмета.

select last\_name, COUNT(subj) from teach group by last\_name having COUNT(subj)>1



1. Вывести номера курсов с привязкой к факультету и указанием суммы часов по всем предметам, проводимых на них

select year,faculty, SUM(hours) as hours from teach group by year,faculty



1. Вывести количество предметов, читаемых на каждом факультете преподавателями иностранцами

Select faculty, count(subj) from teach where s\_name is null group by faculty



Задания под вопросом (старая нумерация - 25 и 29)

1. Вывести количество преподавателей, работающих на каждом факультете

select faculty, count(distinct (last\_name )) as 'saze' from teach group by faculty



1. Вывести количество предметов, читаемых на каждом факультете на втором курсе, отсортировать по факультетам в убывающем порядке, по курсам в возрастающем;

Вопрос исправлен на: Вывести количество предметов, читаемых на каждом факультете на втором курсе

Select faculty, count(subj) from teach where year=2 group by faculty



***Контрольные вопросы.***

**1) Какие 2 раздела являются обязательными при выполнении запроса в SQL**

*Что вывести SELECT*

*и откуда FROM*

**2) за что отвечает GROUP BY, HAVING и WHERE в запросах с группировкой?**

*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). WHERE определяет, какие записи должны участвовать в группировании, т.е. фильтрует до группирования. A**HAVING используется для фильтрации записей, полученных в результате группировки и отбора WHERE.*

**3) Что из WHERE и HAVING применяется для фильтрации записей до группирования, а что- после?**

*WHERE фильтрует до группирования.*

*HAVING определяет, какие из получившихся в результате группировки  
записей будут включены в результирующую выборку, т.е. фильтрует записи после группирования.*

**4) Дать определения агрегатным функциям. Привести 3-4 примера названий функций**

*Агрегатные функции рассчитывают результат функции по переданным значениям. Например select max(exm) from stud рассчитает максимальную оценку среди всех в таблице stud. Агрегатные функции возвращают одно результирующее значение или с группировкой по полям в каждой группе. Для расчета функции по группе используем GROUP BY.*

*Например, AVG(), MIN()*

**5) О чем говорит значение 0, полученное от функции COUNT(\*)?**

*Это значит, что ни одна строка таблицы не соответствует нашему запросу.*

*Например, select count(\*) from stud where exm>9, к нашей базе UNIVER вернет 0, т.к. нет ни одной записи, где оценка больше 9. К примеру вот select count(\*) from stud where exm>7 вернет 11, т.к. этому запросу будут соответствовать 11 записей.*