Лабораторная работа

Тема: Выборка данных.

Цель работы:Изученить операторы SQL.

SELECT

Самым часто используемым оператором языка SQL является оператор ***SELECT***.Этот оператор предназначен для выборки информации из таблиц базы данных. Упрощенный синтаксис оператора выглядит следующим образом:

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

В квадратных скобках указаны необязательные элементы.

SELECT указывает на то, что эта команда является запросом на извлечение информации.

DISTINCT - используется для исключения из результатов запроса повторяющихся записей.

FROM – в этом разделе указывается список таблиц, из которых будет извлечена информация.

Пример.

*Select * From Student*

Присутствие значка «*» в этом примере говорит о том, что из таблицы Student необходимо выбрать все столбцы.

Select Name,Surname,Kurs From Student

По данному запросу будут выбраны данные из стобцов Name,Surname,Kurs таблицы Student.

Select DISTINCT Name,Surname,Kurs From Student

В данном случае из выходного набора данных, сформированного по вышепредставленному запросу, будут удалены дублирующиеся записи.То есть каждая запись будет уникальна .

WHERE - задается условие отбора ***отдельных записей*** в выходной набор.

Пример

Select DISTINCT Name,Surname,Kurs
From Student
Where Kurs=4 AND City="Брест"

Этот запрос содержит так называемый фильтр или условие отбора, которое в свою очередь состоит из "пересечения" двух условий, соединенных логическим оператором AND. Что бы запись попала в выходной нобор, условия должны выполняться одновременно, т.е. поле Kurs записи должно содержать значение 4 и поле City записи должно содержать значение "Брест".

ORDER BY – позволяет упорядочивать выводимые данные в соответствии со значениями одного или нескольких столбцов. При этом можно задавать возрастающую **(ASC)** или убывающую **(DESC)** последовательность сортировки.

Пример

Select DISTINCT Name,Surname,Kurs
From Student
Where Kurs=4 AND City="Брест
Order by Kurs DESC

Записи в выходном наборе будут упорядочиваться по убыванию значений в поле Kurs

GROUP BY– позволяет группировать записи в подмножества, определяемые значениями каких либо полей. Позволяет применять агрегирующие функции к сформированным группам. При этом в разделе GROUP BY должны быть указаны все поля, используемые в разделе SELECT за исключением агрегирующих функций.

> **Агрегирующие функции** – особый класс функций, ориентированный на работу со множествами. Используя множество в качестве аргумента, выдают результат в виде единственного значения, являющийся некой обобщенной характеристикой для этого множества. Агрегирующие

функции не могут использоваться *непосредственно* в разделе WHERE .

COUNT – вычисляет число всех выбранных значений заданного в аргументе этой функции поля.

SUM – вычисляет арифметическую сумму всех выбранных значений заданного в аргументе этой функции поля.

AVG - вычисляет среднее арифметическое всех выбранных значений заданного в аргументе этой функции поля.

MAX – вычисляет максимальный элемент из всех выбранных значений заданного в аргументе этой функции поля.

MIN- вычисляет минимальный элемент из всех выбранных значений заданного в аргументе этой функции поля.

HAVING – определяет критерий, по которому *группы записей* следует включать в выходной набор.

Пример

Select Sum(stipend)
from Student
Group By KURS
Having Kurs >2

Данный запрос выведет сумму стипендии студентов 3,4,5 курсов

UNION

Оператор SELECT используется в сочетании с оператором объединения **UNION**. **UNION** используется для объединения выходных данных нескольких запросов в единый результирующий набор.При этом соответствующие поля должны иметь один и тот же тип и размер, если в одном из столбцов объединяемых запросов запрещены NULL -значения, то и в соответствующих столбцах других запросов NULL - значения должны быть запрещены.

Пример

Select Surname,Name,Kurs,Stipend
From Student
Where Stipend=200

UNION

Select Surname,Name,Kurs,Stipend
From Student
Where Stipend=500

Другие используемые операторы

При формировании критериев отбора используются не только агрегирующие функции, но различные операторы .

Реляционные опретароы

Это математические символы,которые задают определенный тип сравнения между двумя значениями.

=	Равно
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно
<>	Не равно

Булевы операторы

AND – берет два булевых выражения в качестве аргументов и дает в результате истину, если оба эти выражения истинны.

Пример

Select Surname,Name,Kurs
From Student
Where Univ_id =2 AND Kurs=5

Этим запросом будут выбраны данные о студентах, учащихся в университете с идентификаторами 2 не 5 курсе.

OR – оценивает результат как истину, если хотя бы один из аргументов истинен.

Пример

Select Surname,Name,Kurs
From Student
Where Univ_id =2 OR Univ_id =5 OR Univ_id =7

Этим запросом будут выбраны данные о студентах, учащихся в университетах с идентификаторами 2 ,либо 5, либо7.

NOT – берет единственное булево выражение в качестве аргумента и изменяет его на противоположное, т.е. истину на ложь, а ложь на истину.

Пример

Select Surname,Name,Kurs
From Student
Where Univ_id NOT IN (2,5,7)

Этим запросом будут выбраны данные о студентах, не учащихся в университетах с идентификаторами 2 ,либо 5, либо7.

Специальные операторы

IN – можно прочитать как «равен любому из списка». Используется для проверки вхождения значения в данное множество,которое задается через перечисление его элементов. Результат будет истинным, если заданное значение входит в множество.

Пример

Select Surname,Name,Kurs
From Student
Where Univ_id IN (2,5,7)

Этим запросом будут выбраны данные о студентах, учащихся в университетах с идентификаторами 2 ,либо 5, либо7.

BETWEEN – сходен по логике работы с оператором IN. Вместо перечисленных элементов множества задаются границы диапазона.

Пример

Select Surname,Name,Kurs
From Student
Where Stipend BETWEEN 200 and 500

Этим запросом будут выбраны данные о студентах, чья стипендия нахоится в пределах от 200 до 500 единиц.

LIKE - осуществляет просмотр строки для выяснения: входит или не входит заданная подстрока в просматриваемую строку. Применим только к данным типа CHAR или VARCHAR. С оператром LIKE используются *шаблоны*:

«_» (подчеркивание) – заменяет один любой символ.

«%» (процент) – заменяет последовательность символов произвольной длины.

Пример

Select Surname,Name,Kurs
From Student
Where Surname 'LIKE И%'

Этим запросом будут выбраны данные о студентах, с фамилией, начинающейся на букву И.

IS NULL – результат истинен, если аргумент этого оператора есть NULL-значение(т.е. незаданное значение).

Пример

Select Surname,Name,Kurs
From Student
Where Stipend IS NULL

Этим запросом будут выбраны данные о студентах, с отсутствующими данными о стипендии.

ANY,SOME – логика работы этих операторов аналогично логике работы оператора IN.

ALL – логику работы этого оператора проще оъяснить на примерах:

> ALL(100,101,102) - «больше, чем все заданные в аргументе оператора значения»

< ALL(100,101,102) - «меньше, чем все заданные в аргументе оператора значения»

<> ALL(100,101,102) - «не равно ни одному из приведенных значений»

EXISTS – в качестве аргумента этот оператор использует подзапрос. Генерирует истину, если этот подзапрос вернул какие-либо выходные данные,а в противном случае генерируется ложь.

Пример

Select Surname,Name,Kurs
From Student
Where EXISTS (Select Mark from EXAM_MARKS
Where EXAM_MARKS.ID=Student.ID)

Этим запросом будут выбраны данные о студентах,данные об оценках которых присутствуют в таблице EXAM_MARKS.

Все вышеперечисленные операторы могут непосредственно предшесвовать оператору **NOT**, который инвертирует результат, полученный с помощью специального оператора.В этом случае получаются такие сочетания, как **IS NOT NULL,NOT IN,NOT BTWEEN, NOT LIKE,NOT EXISTS** и т.д.

Вложенные и связанные подзапросы

В SQL возможно вкладывать одни запросы внутрь других запросов. Такие внутренние запросы называются **вложенными** или **связанными подзапросами** в зависимости от организации своей структуры.

Вложенные подзапросы – это запросы, выполняемые внутри других запросов.

Предположим, что известна фамилия студента (“Петров”), надо получить список его оценок.

Пример

Select Mark From Exam_Mark
Where Student_ID= (Select Student_Id From Student
Where Surname='Петров')

Когда во вложенном запросе осуществляется ссылка на таблицы внешнего запроса, то подзапрос называется ***связанным***.

Пример.

```
Select Student_id From Student A
                    Where 5=(Select AVG(mark) From Exam_Marks B
                                        Where A.Student_ID=B.Student_Id)
```

Ход работы

1. Выполнить задания согласно полученному варианту.
2. Написанные запросы должны быть протестированы.
3. Протестированные запросы должны быть предоставлены преподавателю вместе с результатами тестов.
4. На основании выполненных заданий составить отчет.

Вариант №1

1. Напишите запрос , выполняющий вывод номера студента, фамилию студента и новую величину стипендии для этого студента, увеличенную на 20%. Выходные данные упорядочить по значению последнего столбца.

2. Напишите запрос, выводящий персональные данные студента, у которых стипендия совпадает с максимальным значением стипендии для города, в котором живет студент.

3. Напишите запрос с оператором EXISTS, выбирающий сведения обо всех студентах, для которых в том же городе, где живет студент, существуют университеты, в которых он не учится.

4. Напишите запрос, использующий операторы ANY или ALL, выполняющий выборку данных о студентах, у которых в городе их постоянного места жительства нет университета.

5. Создайте объединение двух запросов, которые выдают значения полей UNIV_NAME, CITY, RATING для всех университетов. Те из них, у которых рейтинг равен или выше 300, должен иметь комментарий ‘Высокий’, все остальные – ‘Низкий’.

6. Написать запрос, выполняющий вывод количества студентов, имеющих только отличные оценки.

7. Написать запрос, выводящий список студентов, средняя оценка у которых не превышает 4 баллов.

8. Написать вывод фамилий преподавателей, учебная нагрузка которых (количество учебных часов, проводимых преподавателем в совокупности) превышает нагрузку преподавателя **Н**.

9. Написать запрос, выполняющий вывод количества часов занятий, проводимых преподавателем **И**.

10. Написать запрос, выполняющий вывод фамилий преподавателей, в названии предметов которых присутствует сочетание букв **ма**.

Вариант №2

1. Напишите запрос, который выводит суммы баллов всех студентов для каждой даты сдачи экзаменов и представляет результаты в порядке убывания этих сумм.

2. Напишите запрос, который позволяет вывести имена и идентификаторы всех студентов, для которых точно известно, что они проживают в городе, где нет ни одного университета.

3. Напишите запрос, который выбирает данные о названиях университетов, рейтинг которых равен либо превосходит рейтинг такого-то университета.

4. Выведите список студентов и преподавателей, живущих в Москве, с соответствующими комментариями ‘Студент’, ’Преподаватель’

5. Написать запрос, выполняющий вывод данных об именах и фамилиях студентов, не получивших ни одной отличной оценки.

6. Написать запрос, выполняющий вывод имен и фамилий преподавателей, читающих два и более различных предмета обучения.

7. Написать запрос, выполняющих вывод данных о фамилиях преподавателей, проводящих занятия у студентов, обучающихся в университетах с рейтингом , превышающим 200.

8. Написать запрос, выполняющий выборку значений идентификаторов студентов, имеющих такие же оценки, что и студент с идентификатором 12.

9. Написать запрос, выполняющий вывод данных об именах и фамилиях студентов, получивших хотя бы одну отличную оценку.

10. Написать запрос, выполняющий вывод количества часов занятий, проводимых преподавателями в фамилиях которых присутствует сочетание букв **ир**.

Учебная база данных STUDENTS

Student_id	Surname	Name	Stipend	Kurs	City	Birthday	Univ_id
Student_id – числовой код, идентифицирующий студента Surname – фамилия студента Name – имя студента Stipend – стипендия , получаемая студентом Kurs – курс, на котором учится студент City – город, в котором живет студент Birhday – его день рождения Univ_id – числовой код университета, в котором учится студент							

LECTURER

Lecturer_id	Surname	Name	City	Univ_id
Lecturer_id – числовой код, идентифицирующий преподавателя Surname – фамилия преподавателя Name – имя преподавателя City – город, в котором живет преподаватель Univ_id – идентификатор университета, в котором работает преподаватель				

SUBJECT

Subj_id	Subj_name	Hour	Semester
Subj_id – идентификатор предмета обучения Subj_name – наименование предмета обучения Hour – количество часов, отводимых на изучение предмета Semester – семестр, в котором изучается данный предмет Kurs – курс, на котором читается предмет Univ_Id - идентификатор университета, в котором читается предмет			

UNIVERSITY

Univ_id	Univ_name	Rating	City
Univ_id – идентификатор университета Univ_name – название университета Rating – рейтинг университета City – город, в котором расположен университет			

EXAM_MARKS

Exam_id	Student_id	Subj_id	Mark	Exam_date
Exam_id – идентификатор экзамена Student_id – идентификатор студента Subj_id – идентификатор предмета обучения Mark – экзаменационная оценка Exam_date – дата экзамена				

SUBJ_LECT

Lecturer_id	Subj_id
Lecture_id – идентификатор преподавателя Subj_id – идентификатор предмета обучения	

Литература

1. Мартин Грабер Введение в SQL. - Москва:“Лори”,1994.-348 с.
2. Мамаев Е.,Вишневский А. - СПБ.:Питер,2000.-894с.