Лабораторная работа № 5 «Создание SQL-запросов»

Тема: Составление и исполнение запросов на языке SQL в среде СУБД MS Access XP/2003.

Цель работы: создать SQL-запросы на создание таблицы, на выборку с параметрами, на обновление записей, на удаление записей, на добавление данных, на удаление таблицы, на создание индексов.

Основы SQL

Запрос SQL — это запрос, создаваемый при помощи инструкций SQL. Язык SQL (Structured Query Language) используется при создании запросов, а также для обновления и управления реляционными БД.

В среде MS Access, когда пользователь создает запрос в режиме конструктора запроса (с помощью построителя запросов), MS Access автоматически создает эквивалентную инструкцию SQL. Фактически, для большинства свойств запроса, доступных в окне свойств в режиме конструктора, имеются эквивалентные предложения или параметры языка SQL, доступные в режиме SQL. При необходимости, пользователь имеет возможность просматривать и редактировать инструкции SQL в режиме SQL. После внесения изменений в запрос в режиме SQL его вид в режиме конструктора может измениться.

Некоторые запросы, которые называют запросами SQL, невозможно создать в бланке запроса. Для запросов к серверу, управляющих запросов и запросов на объединение необходимо создавать инструкции SQL непосредственно в окно запроса в режиме SQL. Для подчиненного запроса пользователь должен ввести инструкцию SQL в строку Поле или Условие отбора в бланке запроса.

Синтаксиса написания SQL-предложений:

– в описании команд слова, написанные прописными латинскими буквами, являются зарезервированными словами SQL;

– фрагменты SQL-предложений, заключенные в фигурные скобки и разделенные символом «|», являются альтернативными. При формировании соответствующей команды для конкретного случая необходимо выбрать одну из них;

– фрагмент описываемого SQL-предложения, заключенный в квадратные скобки **[ ],** имеет необязательный характер и может не использоваться;

– многоточие …, стоящее перед закрывающейся скобкой, говорит о том, что фрагмент, указанный в этих скобках, может быть повторен;

Описание команд SQL

Выборка записей

Инструкция **SELECT**. При выполнении инструкции SELECT СУБД находит указанную таблицу или таблицы, извлекает заданные столбцы, выделяет строки, соответствующие условию отбора, и сортирует или группирует результирующие строки в указанном порядке в виде набора записей.

Синтаксис команды:

SELECT [предикат] { * | таблица.* | [таблица.]поле_1
[AS псевдоним_2] [, [таблица.]поле_2[AS псевдоним_2] [, ...]]}
FROM выражение [, ...]
[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...]

где предикат — один из следующих предикатов отбора: ALL, DISTINCT, DISTINCTROW, TOP. Данные ключевые слова используются для ограничения числа возвращаемых записей. Если они отсутствуют, по умолчанию используется предикат ALL;

* указывает, что результирующий набор записей будет содержать все поля заданной таблицы или таблиц. Следующая инструкция отбирает все поля из таблицы «Студенты»: SELECT * FROM Студенты;

таблица — имя таблицы, из которой выбираются записи;

поле_1, поле_2 — имена полей, из которых должны быть отобраны данные;

псевдоним_1, псевдоним_2 — ассоциации, которые станут заголовками столбцов вместо исходных названий полей в таблице;

выражение — имена одной или нескольких таблиц, которые содержат необходимые для отбора записи;

предложение GROUP BY в SQL-предложении объединяет записи с одинаковыми значениями в указанном списке полей в одну запись. Если инструкция SELECT содержит статистическую

функцию SQL, например Sum или Count, то для каждой записи будет вычислено итоговое значение;

предложение HAVING определяет, какие сгруппированные записи, выданные в результате выполнения запроса, отображаются при использовании инструкции SELECT с предложением GROUP BY. После того как записи результирующего набора будут сгруппированы с помощью предложения GROUP BY, предложение HAVING отберет те из них, которые удовлетворяют условиям отбора, указанным в предложении HAVING;

предложение ORDER BY позволяет отсортировать записи, полученные в результате запроса, в порядке возрастания или убывания

на основе значений указанного поля или полей.

Следует отметить, что инструкции SELECT не изменяют данные в базе данных. Приведем минимальный синтаксис инструкции SELECT: SELECT поля FROM таблица.

Если несколько таблиц, включенных в предложение FROM, содержат одноименные поля, перед именем такого поля следует ввести имя таблицы и оператор « . » (точка). Предположим, что поле «Номер_группы» содержится в таблицах «Студенты» и «Группы». Следующая инструкция SQL отберет поле «Номер_группы» и «ФИО_студента» из таблицы «Студенты» и «ФИО_куратора» из таблицы «Группы» при номере группы, равном 432-1:

SELECT Группы.Номер_группы, Группы.ФИО_куратора, Студенты.ФИО_студента

FROM Группы, Студенты

WHERE Группы.Номер_группы = Студенты.Номер_группы AND

На рис. 18 приведен пример выполнения данного запроса.

Таблицы БД

СТУДЕНТЫ

Номер_зачет-ной книжки	**ФИО_студента**	**Дата рождения**	**Место рождения**	**Номер_группы**
1992412-11	Карасев А.А.	27.08.75	г. Чита	412-1
1992432-11	Данилов О. В.	27.08.75	г. Алматы	432-1
1992432-12	Раевский А. И.	20.05.75	г. Бишкек	432-1
1992432-22	Глазов О.А	04.07.75	г. Киров	432-1

ГРУППЫ

Номер_Группы	ФИО_куратора
412-1	Самойлов С.С.
432-1	Авдеев Р.М

Результат выполнения запроса

Номер_группы	ФИО_куратора	ФИО_студента
432-1	Авдеев Р.М	Данилов О. В.
432-1	Авдеев Р.М	Раевский А. И.
432-1	Авдеев Р.М	Глазов О.А

Рис. 18. Пример выполнения запроса на выборку

Помимо обычных знаков сравнения (=,<,>,<=,>=, <>) в языке SQL в условии отбора используются ряд ключевых слов:

Is not null — выбрать только непустые значения;

Is null — выбрать только пустые значения;

Between … And определяет принадлежность значения выражения указанному диапазону.

Синтаксис:

выражение [Not] Between значение_1 And значение_2 ,

где выражение — выражение, определяющее поле, значение которого проверяется на принадлежность к диапазону;

значение_1, значение_2 – выражения, задающие границы диапазона.

Если значение поля, определенного в аргументе выражение, попадает в диапазон, задаваемый аргументами значение_1 и значение_2 (включительно), то оператор Between...And возвращает значение True; в противном случае возвращается значение False. Логический оператор Not позволяет проверить противоположное условие: что выражение находится за пределами диапазона, заданного с помощью аргументов значение_1 и значение_2.

Оператор Between...And часто используют для проверки: попадает ли значение поля в указанный диапазон чисел. В

следующем примере выдается список студентов, получающих стипендию от 800 до 900 рублей:

SELECT ФИО_студента, Размер_стипендии

FROM Студенты

WHERE Размер_стипендии Between 800 And 900

На рис. 19 приведен результат выполнения запроса.

СТУДЕНТЫ

Номер_зачетной книжки	ФИО_студента	Размер_стипендии
1992412-11	Карасев А.А.	900
1992432-11	Данилов О. В.	800
1992432-12	Раевский А. И.	950
1992432-22	Глазов О.А	850

Результирующий набор данных

ФИО_студента	Размер_стипендии
Карасев А.А.	900
Данилов О. В.	800
Глазов О.А	850

Рис. 19. Результат выполнения запроса на выборку с использованием операторов Between...And

Если выражение, значение_1 или значение_2 имеет значение Null, оператор Between...And возвращает значение Null.

Оператор Like используется для сравнения строкового выражения с образцом.

Синтаксис:

выражение Like «образец»,

где выражение — выражение SQL, используемое в предложении WHERE; образец — строка, с которой сравнивается выражение.

Оператор Like используется для нахождения в поле значений, соответствующих указанному образцу. Для аргумента образец можно задавать полное значение (например, Like «Иванов») или использовать подстановочные знаки для поиска диапазона значений (например, Like "Ив*").

Приведем перечень подстановочных символов и пример их использования в языке Jet SQL согласно документации Microsoft.

Параметры оператора Like

Тип совпадения	Образец	Совпадение (True)	Несовпадение (False)

Несколько символов	a*a	aa, aBa, aBBBa	aBC
	ab	abc, AABB, Xab	aZb, bac
Специальный символ	a[*]a	a*a	aaa
Несколько символов	ab*	abcdefg, abc	cab, aab
Одиночный символ	a?a	aaa, a3a, aBa	aBBBa
Одиночная цифра	a#a	a0a, a1a, a2a	aaa, a10a
Диапазон символов	[a-z]	f, p, j	2, &
Вне диапазона	[!a-z]	9, &, %	b, a
Не цифра	[!0-9]	A, a, &, ~	0, 1, 9
Комбинированное выражение	a[!b-m]#	An9, az0, a99	abc, aj0

Внутреннее соединение

Операция **INNER JOIN** объединяет записи из двух таблиц, если связующие поля этих таблиц содержат одинаковые значения.

Синтаксис операции:

FROM таблица_1 INNER JOIN таблица_2 ON таблица_1.поле_1 оператор таблица_2.поле_2

где таблица_1, таблица_2 — имена таблиц, записи которых подлежат объединению;

поле_1, поле_2 — имена объединяемых полей. Поля должны иметь одинаковый тип данных и содержать данные одного рода, однако эти поля могут иметь разные имена;

оператор — любой оператор сравнения: "=," "<," ">," "<=," ">=," или "<>".

Операцию INNER JOIN можно использовать в любом предложении FROM. Это самые обычные типы связывания. Они объединяют записи двух таблиц, если связующие поля обеих таблиц содержат одинаковые значения. Предыдущий пример использования команды SELECT можно записать с использованием конструкции INNER JOIN следующим образом:

SELECT Группы.Номер_группы, Группы.ФИО_куратора, Студенты.ФИО_студента

FROM Группы INNER JOIN Студенты

ON Группы.Номер_группы = Студенты.Номер_группы;

Внешнее соединение

Операции **LEFT JOIN, RIGHT JOIN** объединяют записи исходных таблиц при использовании в любом предложении FROM.

Операция LEFT JOIN используется для создания внешнего соединения, при котором все записи из первой (левой) таблицы включаются в результирующий набор, даже если во второй (правой) таблице нет соответствующих им записей.

Операция RIGHT JOIN используется для создания внешнего объединения, при котором все записи из второй (правой) таблицы включаются в результирующий набор, даже если в первой (левой) таблице нет соответствующих им записей.

Синтаксис операции:

FROM таблица_1 [LEFT | RIGHT] JOIN таблица_2

ON таблица_1.поле_1 оператор таблица_2.поле_2

Например, операцию LEFT JOIN можно использовать с таблицами «Студенты» (левая) и «Задолженность_за_обучение» (правая) для отбора всех студентов, в том числе тех, которые не являются задолжниками:

```
SELECT Студенты.ФИО_студента,
Задолженность_за_обучение.Сумма_задолженности
FROM Студенты LEFT JOIN Задолженность_за_обучение
ON Студенты.Номер_зачетной_книжки =
 Задолженность_за_обучение.Номер_зачетной_книжки;
```

Поле «Номер_зачетной_книжки» в этом примере используется для объединения таблиц, однако, оно не включается в результат выполнения запроса, поскольку не включено в инструкцию SELECT. Чтобы включить связующее поле (в данном случае поле «Номер_зачетной_книжки») в результат выполнения запроса, его имя необходимо включить в инструкцию SELECT.

Важно отметить, что операции LEFT JOIN или RIGHT JOIN могут быть вложены в операцию INNER JOIN, но операция INNER JOIN не может быть вложена в операцию LEFT JOIN или RIGHT JOIN.

Перекрестные запросы

В некоторых СУБД (в частности в MS Access) существует такой вид запросов как перекрестный. В перекрестном запросе отображаются результаты статистических функций — суммы, средние значения и др., а также количество записей. При этом подсчет выполняется по данным из одного полей таблицы.

Результаты группируются по двум наборам данных, один из которых расположен в левом столбце таблицы, а другой в заголовке таблицы. Например, при необходимости вычислить средний балл студентов за семестр, обучающихся на разных кафедрах, необходимо реализовать перекрестный запрос, в результате выполнения которого будет создана таблица, где заголовками строк будут служить номер семестра, заголовками столбцов — названия кафедр, а в полях таблицы будет рассчитан средний балл.

Для создания перекрестного запроса необходимо использовать следующую инструкцию:

TRANSFORM статистическая_функция

инструкция_SELECT

PIVOT поле [IN (значение_1[, значение_2[, ...]])],

где статистическая_функция — статистическая функция SQL, обрабатывающая указанные данные;

инструкция_SELECT — запрос на выборку;

поле — поле или выражение, которое содержит заголовки столбцов для результирующего набора;

значение_1, значение_2 — фиксированные значения, используемые при создании заголовков столбцов.

Составим SQL-запрос, реализующий описанный выше пример. В качестве исходного набора данных используется таблица Успеваемость (рис. 20).

Успеваемость : таблица

НОМЕР_ЗАЧЕТНОЙ_КНИЖКИ	Семестр	Оценка	Кафедра
102	1	3	АСУ
102	2	3	АСУ
102	3	4	АСУ
102	4	4	АСУ
110	1	5	АОИ
110	2	3	АОИ
110	3	3	АОИ
110	4	4	АОИ
110	5	4	АОИ

Рис. 20. Таблица УСПЕВАЕМОСТЬ

В результате выполнения нижеприведенного перекрестного SQL-запроса формируется следующая таблица (рис. 21):

```
TRANSFORM AVG(Успеваемость.Оценка) AS Сред_балл
SELECT Успеваемость.Семестр
FROM Успеваемость
```

GROUP BY Успеваемость.Семестр
PIVOT Успеваемость.Кафедра


SQL_запрос : перекрестный запрос

Семестр	АОИ	АСУ
1	5	3
2	3	3
3	3	4
4	4	4
5	4	

Рис. 21. Результат выполнения перекрестного запроса

Таким образом, когда данные сгруппированы с помощью перекрестного запроса, можно выбирать значения из заданных столбцов или выражений и определять как заголовки столбцов. Это позволяет просматривать данные в более компактной форме, чем при работе с запросом на выборку.

Подчиненные запросы

Часто возникает ситуация, когда желаемый результат нельзя получить с помощью одного SQL-запроса. Одним из способов решения такой задачи является использование подчиненных запросов в составе главного SQL-запроса. Подчиненным SQL-запросом называют инструкцию SELECT, включаемую в инструкции SELECT, SELECT...INTO, INSERT...INTO, DELETE или UPDATE или в другой подчиненный запрос. Подчиненный запрос может быть создан одним из трех способов, синтаксис которых представлен ниже:

1) сравнение [ANY | ALL | SOME] (инструкцияSQL)

2) выражение [NOT] IN (инструкцияSQL)

3) [NOT] EXISTS (инструкцияSQL),

где сравнение — выражение и оператор сравнения, который сравнивает выражение с результатами подчиненного запроса;

выражение — выражение, для которого проводится поиск в результирующем наборе записей подчиненного запроса;

инструкцияSQL — инструкция SELECT, заключенная круглые скобки.

Подчиненный запрос можно использовать вместо выражения в списке полей инструкции SELECT или в предложениях WHERE и HAVING. Инструкция SELECT используется в подчиненном запросе для задания набора конкретных значений, вычисляемых в выражениях предложений WHERE или HAVING.

Предикаты ANY или SOME, являющиеся синонимами, используются для отбора записей в главном запросе, которые удовлетворяют сравнению с записями, отобранными в подчиненном запросе. В следующем примере отбираются все студенты, средний балл которых за семестр больше 4.

```
SELECT * FROM Студенты
WHERE Номер_зачетной_книжки = ANY
(SELECT Номер_зачетной_книжки FROM Успеваемость
WHERE оценка > 4)
```

Предикат ALL используется для отбора в главном запросе только тех записей, которые удовлетворяют сравнению со всеми записями, отобранными в подчиненном запросе. Если в предыдущем примере предикат ANY заменить предикатом ALL, результат запроса будет включать только тех студентов, у которых средний балл больше 4. Это условие является значительно более жестким.

Предикат IN используется для отбора в главном запросе только тех записей, которые содержат значения, совпадающие с одним из отобранных подчиненным запросом. Следующий пример возвращает сведения обо всех студентах, средний балл которых за семестр был больше 4.

```
SELECT * FROM Студенты
WHERE Номер_зачетной_книжки in
(SELECT Номер_зачетной_книжки FROM Успеваемость
WHERE оценка > 4)
```

Предикат NOT IN используется для отбора в главном запросе только тех записей, которые содержат значения, не совпадающие ни с одним из отобранных подчиненным запросом.

Предикат EXISTS (с необязательным зарезервированным словом NOT) используется в логическом выражении для определения того, должен ли подчиненный запрос возвращать какие-либо записи.

В подчиненном запросе можно использовать псевдонимы таблиц для ссылки на таблицы, перечисленные в предложении FROM, расположенном вне подчиненного запроса. В следующем примере отбираются фамилии и имена студентов, чья стипендия равна или больше средней стипендии студентов, обучающихся в той же группе. В данном примере таблица СТУДЕНТЫ получает псевдоним С1:

```
SELECT Фамилия,
Имя, Номер_группы, Стипендия
FROM СТУДЕНТЫ AS C1
WHERE Стипендия >=
(SELECT Avg(Стипендия)
FROM СТУДЕНТЫ
WHERE C1.Номер_группы = СТУДЕНТЫ.Номер_группы)
Order by Номер_группы;
```

В последнем примере зарезервированное слово AS не является обязательным.

Некоторые подчиненные запросы можно использовать в перекрестных запросах как предикаты (в предложении WHERE). Подчиненные запросы, используемые для вывода результатов (в списке SELECT), нельзя использовать в перекрестных запросах.

Порядок выполнения работы

Все запросы, создаваемые в рамках данной лабораторной работы, должны быть реализованы на языке SQL без помощи построителя запросов.

Для создания нового SQL-запроса в окне базы данных нажмите кнопку Создание запроса в режиме конструктора и не выбирая таблицы для запроса нажмите кнопку Вид  на панели инструментов.

1. Используя команду SELECT, создайте запрос на выборку записей из двух (или более) таблиц, используя правила внешнего и внутреннего соединения, а также различные условия отбора и сортировки.
2. Используя команду UPDATE, создайте запрос на обновление данных в созданных ранее таблицах.
3. Используя команду INSERT INTO, создайте запросы на добавление группы записей и одной записи в существующую таблицу.
4. Используя команду CREATE INDEX, создайте запрос на создание нового индекса, используя различные условия на значения индексов (IGNORE NULL, DISSALLOW NULL, PRIMARY), а также типы сортировки.

5. Используя команду DROP, создайте запросы на удаление таблицы и индекса, созданных ранее в БД.
   Сохраните все созданные запросы в базе данных.