**Лабораторная работа № 6.**

**Модификация данных.**

*Цель**работы*: Приобретение практических навыков по созданию таблиц, вводу и обновлению данных.

**Краткие теоретические сведения.**

Кроме инструкции SELECT, которая была рассмотрена ранее, язык манипуляции данными DML (Data Manipulation Language) содержит три другие инструкции: INSERT, UPDATE и DELETE, которые могут быть использованы для модификации содержимого таблиц баз данных.

1. **Инструкция INSERT**

Инструкция INSERT предназначена для вставки строк в таблицу. Существует две формы этой инструкции:

**INSERT [INTO] Имя\_таблицы [(Список\_столбцов)]**

**VALUES (Список значений)**

**INSERT INTO Имя\_таблицы [(Список\_столбцов)]**

**{Инструкция\_Select | Инструкция\_Execute }**

Первая форма инструкции INSERT позволяет вставить в таблицу одну строку, a вторая форма этой инструкции позволяет вставить в таблицу результирующий набор, возвращаемый инструкции SELECT или хранимой процедуры, запуск на выполнение которой осуществляется инструкцией EXECUTE. Результирующий набор, возвращаемый инструкциями SELECT или EXECUTE должен иметь строки, которые могут быть вставлены в таблицу.

Для обеих форм инструкции SELECT тип данных каждого вставляемого значения должен быть совместимым с типом данных соответствующего столбца таблицы.

**Вставка одной строки**

Для обеих форм инструкции INSERT явное указание списка столбцов не является обязательным. Отсутствие списка столбцов равнозначно указанию всех столбцов таблицы.

В столбцы с типом данных TIMESTAMP или свойством IDENTITY по умолчанию вставляются значения, автоматически создаваемые системой. Для столбцов других типов данных вставляется соответствующее ненулевое значение по умолчанию, если таковое имеется, или NULL в противном случае. Если для столбца значения NULL не разрешены и для него не определено значение по умолчанию, выполнение инструкции INSERT завершается ошибкой и выводится соответствующее сообщение.

В примерах 6.1—6.4 вставляются строки в четыре таблицы базы данных Shop, демонстрируя использование инструкции INSERT для вставки небольшого объема данных в базу данных.

Пример 6.1. Вставка строки в таблицу Employees

**USE Shop;**

**INSERT Employees VALUES ('Петренко', 'Сергей', 'Николаевич', '1997-09-30', '80293563499', 4, 5)**

В примере 6.1. отсутствует список столбцов так как после слова VALUES перечислены значения для всех столбцов таблицы, кроме empId. Столбец empId является автозаполняемым (Identity), поэтому его значение указывать не нужно. При выполнении запроса, приведенного в примере 6.1 в таблицу Employees будет вставлена строка.

Пример 6.2 Вставка части строки в таблицу Employees

**USE Shop;**

**INSERT Employees (LastName, FName, MName, BirthDate) VALUES ('Даниленко', 'Андрей', 'Борисович', '1994-04-07')**

В примере 6.2 список столбцов присутствует так как заполняются не все поля записи. При выполнении запроса, приведенного в примере 6.2 в таблицу Employees будет вставлена часть строки, а поля, не указанные в списке столбцов, будут заполнены значениями NULL, если они разрешены или значениями по умолчанию, если они определены. Если для каких-то полей не разрешены значения NULL и эти поля не указаны в списке столбцов, то запись не будет помещена в таблицу, а SQL Server выдаст соответствующее сообщение об ошибке.

**Вставка нескольких строк**

Вторая форма инструкции INSERT вставляет в таблицу столько строк, сколько их содержится в результирующем наборе, возвращаемом инструкциями SELECT или EXECUTE. В примере 6.3 показана вставка строк в таблицу, c использованием второй форму инструкции INSERT. В данном случае во вновь созданную таблицу Drivers (Водители) помещается информация о водителях компании, отбираемая инструкцией SELECT из таблицы Employees.

Пример 6.3. Вставка в таблицу строк, выбранных инструкцией SELECT

**USE Shop;**

**CREATE TABLE Drivers (**

**DrvId INT CONSTRAINT PK\_Drv\_Id PRIMARY KEY IDENTITY(1000, 1),**

**DLName NVARCHAR(30) NOT NULL,**

**DFName NVARCHAR(30) NOT NULL,**

**DMName NVARCHAR(30),**

**DBirthDate DATE )**

**INSERT INTO Drivers**

**SELECT LName, FName, MName, BirthDate**

**FROM Employees**

**WHERE posId=6;**

Создаваемая в примере 6.3 новая таблица Drivers имеет столбцы для записи фамилий, имен, отчеств и дат рождения водителей. Инструкция SELECT, входящая в инструкцию INSERT выбирает в таблице Employees части строк, для которых значение столбца posId равно 6 (водитель), которые затем вставляются в созданную новую таблицу.

Содержимое таблицы Drivers после выполнения запроса будет иметь вид:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DrvId** | **DLName** | **DFName** | **DMName** | **DBirthDate** |
| 1008 | Жданович | Юрий |  | 21.08.1980 |
| 1009 | Бондаренко | Сергей |  | 22.03.1980 |

**Конструкторы значений таблицы и инструкция INSERT**

Конструктор значений таблицы или строки (table (row) value constructor) позволяет вставить в таблицу несколько записей (строк) c помощью одного запроса. В примере 6.4 показана вставка в таблицу нескольких строк с помощью инструкции INSERT.

Пример 6.4. Вставка нескольких строк с помощью конструктора значений таблицы

**USE Shop;**

**INSERT INTO Ledger VALUES**

**('2018-09-12', 245),**

**('2018-09-12', 112),**

**('2018-09-13', 44);**

В примере 6.4 инструкция INSERT одновременно вставляет три строки в таблицу Ledger, используя конструктор значений таблицы. Для вставки в таблицу строк с данными посредством конструктора значений таблицы нужно в круглых скобках перечислить значения каждой строки, разделяя как значения каждого списка, так и отдельные списки запятыми.

**2. Инструкция UPDATE**

Инструкция UPDATE предназначена для модификации (изменения) строк таблицы. Эта инструкция имеет следующую общую форму:

**UPDATE Имя\_таблицы**

**{SET Столбец\_1 = значение [,...n]**

**[WHERE логическое\_выражение]**

Строки таблицы Имя\_таблицы выбираются для изменения в соответствии с логическим выражением в предложении WHERE. Значения столбцов каждой модифицируемой строки изменяются с помощью предложения SET инструкции UPDATE, которое соответствующему столбцу присваивает значение. Если предложение WHERE отсутствует, то инструкция UPDATE модифицирует все строки таблицы.

В примере 6.5 приведен запрос, который сотруднику с идентификатором 1001 устанавливает идентификатор должности равный 5, то есть сотрудник с идентификатором 1001 переводится из продавцов в грузчики.

Пример 6.5. Перевод сотрудника с табельным номером 1001 на должность грузчика (posId=5)

**USE Shop;**

**UPDATE Employees**

**SET posId = 5**

**WHERE empId = 1001**

**3. Инструкция DELETE**

Инструкция DELETE предназначена для удаления строк из таблицы.

**DELETE FROM Имя\_таблицы**

**[WHERE логическое выражение];**

При выполнении этой инструкции удаляются все строки, для которых логическое выражение после слова WHERE имеет значение true (истина). Явно перечислять столбцы в инструкции DELETE не то чтобы нет необходимости, а даже не разрешается, поскольку эта инструкция оперирует строками, а не столбцами.

Использование инструкции DELETE показано в примере 6.6.

Пример 6.6. Удаление из таблицы Sales (продажи) всех записей с идентификатором продажи saleId=1 (отмена продажи).

**USE Shop;**

**DELETE FROM Sales**

**WHERE saleId =1;**

Использование предложения WHERE в инструкции DELETE не является обязательным.

Если это предложение отсутствует, то из таблицы удаляются все строки.

1. **Инструкции Truncate Table, Merge, Output**

Сервер SQL Server поддерживает следующие дополнительные инструкции и предложения для модификации таблиц:

* инструкцию TRUNCATE TABLE;
* инструкцию MERGE;
* предложение OUTPUT.

**Инструкция TRUNCATE TABLE**

Инструкция TRUNCATE TABLE является более быстрой версией инструкции DELETE без предложения WHERE. Эта инструкция удаляет все строки таблицы более быстро, чем

инструкция DELETE, поскольку она удаляет содержимое постранично, тогда как

инструкция DELETE делает это построчно.

Инструкция TRUNCATE TABLE имеет следующий синтаксис:

**TRUNCATE TABLE Имя\_таблицы**

**Инструкция MERGE**

**MERGE** – операция в языке T-SQL, при которой происходит обновление, вставка или удаление данных в таблице на основе результатов соединения с данными другой таблицы или SQL запроса. Другими словами, с помощью MERGE можно осуществить слияние двух таблиц, т.е. синхронизировать их.

В операции MERGE происходит объединение по ключевому полю или полям основной таблицы (в которой и будут происходить все изменения) с соответствующими полями другой таблицы или результирующего набора запроса. В итоге, если логическое выражение, используемое для объединения, имеет значение true (истина) (**WHEN MATCHED**), то выполняются операции обновления или удаления, a если это логическое выражение имеет значение false (ложь) (**WHEN NOT MATCHED**), то выполняется операция вставки (INSERT - добавление данных), а также, если в основной таблице присутствуют данные, которое отсутствуют в таблице (или результате запроса) источника (**WHEN NOT MATCHED BY SOURCE**), то выполняется обновление или удаление таких данных.

Использование инструкции MERGE показано в примерах 6.7 и 6.8. В примере 6.7 создается таблица Bonus с информацией о премировании сотрудников, содержащая одну строку со значениями deptId=1 (идентификатор отдела) и bonus =100 (размер премии).

Пример 6.7. Создание однострочной таблицы

**USE Shop;**

**CREATE TABLE Bonus**

**(deptId int, bonus SMALLINT DEFAULT 50);**

**INSERT INTO bonus (deptId, bonus) VALUES (1, 100);**

Таблица bonus после выполнения запроса из примера 6.7 имеет вид

|  |  |
| --- | --- |
| **deptId** | **bonus** |
| 1 | 100 |

В созданную в примере 6.7 таблицу bonus в примере 6.8 будут вставлены новые данные из другой таблицы.

Пример 6.8. Вставка новых строк данных в таблицу bonus

**USE Shop;**

**MERGE Bonus B**

**USING Depts D**

**ON (B.deptId = D.deptId)**

**WHEN MATCHED THEN**

**UPDATE SET B.bonus = B.bonus \* 2**

**WHEN NOT MATCHED THEN**

**INSERT (deptId)**

**VALUES (D.deptId );**

После выполнения запроса из примера 6.8 таблица bonus будет иметь вид

|  |  |
| --- | --- |
| **deptId** | **bonus** |
| 1 | 200 |
| 2 | 50 |
| 3 | 50 |
| 4 | 50 |
| 5 | 50 |
| 6 | 50 |
| 7 | 50 |

В примере 6.8 инструкция MERGE в зависимости от значений в столбце deptId модифицирует данные в таблице Bonus. Если в столбце deptId таблицы Bonus имеется значение из столбца deptId таблицы Depts, то выполняется ветвление MATCHED и существующее значение будет обновлено (увеличено в 2 раза). В противном случае выполняется ветвление NON MATCHED и соответствующая инструкция INSERT вставляет новые строки в таблицу bonus.

**Предложение OUTPUT**

По умолчанию единым видимым результатом выполнения инструкции INSERT, UPDATE или DELETE является только сообщение о количестве модифицированных строк, например "3 rows deleted" (удалены 3 строки). Если такой видимый результат не удовлетворяет пользователя базы данных, то можно использовать предложение OUTPUT, которое выводит модифицированные, вставленные или удаленные строки.

Результаты выполненных операций соответствующих инструкций предложение OUTPUT выводит в таблицах inserted и deleted . Кроме этого, чтобы заполнить таблицы, в предложении OUTPUT требуется использовать выражение INTO. Поэтому для сохранения результата используется табличная переменная.

В примере 6.9 показано использование инструкции OUTPUT с инструкцией DELETE.

Пример 6.9. Применение инструкции OUTPUT

**USE Shop;**

**DECLARE @del\_table TABLE (recordId int, goodId int, quantity decimal(5, 3), price decimal(5, 3), saleId int);**

**DELETE FROM sales**

**WHERE saleId =1**

**OUTPUT DELETED.recordId, DELETED.goodId, DELETED.quantity, DELETED.price, DELETED.saleId INTO @del\_table**

**WHERE saleId =1;**

**SELECT \* FROM @del\_table**

При условии, что содержимое таблицы находится в исходном состоянии, выполнение запроса в примере 6.9 дает следующий результат:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **recordId** | **goodId** | **quantity** | **price** | **saleId** |
| 1 | 1 | 3,00 | 3,00 | 1 |
| 2 | 3 | 2,00 | 8,50 | 1 |
| 3 | 5 | 2,50 | 5,50 | 1 |

В примере 6.9 сначала объявляется табличная переменная @del\_table с пятью столбцами: recordId int, goodId int, quantity decimal(5, 3), price decimal(5, 3), saleId int. В этой таблице будут сохранены удаленные строки. Синтаксис инструкции DELETE расширен предложением OUTPUT:

**OUTPUT DELETED.recordId, DELETED.goodId, DELETED.quantity, DELETED.price, DE:ETEDsaleId INTO @del\_table**

Посредством этого предложения система сохраняет удаленные строки в таблице deleted, содержимое которой потом копируется в переменную @del\_table.

Аналогично предложение OUTPUT может быть применено к инструкции UPDATE. При изменении записи сначала удаляется существующая запись и помещается в таблицу DELETED, а затем вставляется измененная запись и помещается в таблицу INSERTED.

Поэтому при изменении записи с помощью инструкции UPDATE, используя предложение OUTPUT можно вывести изменяемые записи как до изменения так и после.

**Задание на лабораторную работу**

**Вариант 1**

1. Создать новую таблицу «Должники» и с помощью запроса внести в нее данные о читателях, у которых истек срок сдачи книг. Вывести таблицу «Должники»

2. В таблицу «Книги» добавить поле «Год поступления» и с помощью запроса внести в это поле данные, считая, что годом поступления является следующий после года издания год. Вывести таблицу «Книги».

3. Проиндексировать таблицу «Книги» по полям «Название» и «Автор».

4. Удалить из таблицы «Книги» данные о книгах старше 40 лет.

5. Удалить из таблицы «Читатели», читателей, которые не брали книг более 3-х лет.

**Вариант 2**

1. Создать таблицу карточек с истекшим сроком действия и с помощью запроса внести в нее данные о карточках, срок действия которых истек. Вывести созданную таблицу.
2. В таблицу «Карточки» добавить поле «Код безопасности» и с помощью запроса внести в это поле данные, считая, что кодом безопасности последние три цифры номера карточки. Вывести таблицу «Карточки».
3. Проиндексировать таблицу «Клиенты» по полям «Фамилия» и «Адрес».
4. Удалить из таблицы «Карточки» данные о карточках с истекшим сроком действия.
5. Удалить из таблицы «Клиенты» клиентов, у которых нет действующих карточек.

**Вариант 3**

1. Создать таблицу с данными уволенных сотрудников и с помощью запроса внести в нее данные. Вывести созданную таблицу.
2. В таблицу «Должности» добавить поле «Премия» и с помощью запроса внести в это поле данные, считая, что премия составляет 20% от оклада. Вывести таблицу «Должности».
3. Проиндексировать таблицу «Сотрудники» по полям «Фамилия» и «Табельный номер».
4. Удалить из таблицы «Сотрудники» данные об уволенных сотрудниках.
5. Удалить из таблицы «Штатное расписание» данные об уволенных сотрудниках.

**Вариант 4**

1. Создать таблицу «Студенты ФИТР» и с помощью запроса внести в нее данные о студентах факультета ИТР. Вывести созданную таблицу.
2. В таблицу «Факультеты» добавить поле «Аудитория» и с помощью запроса внести в это поле данные, считая, что номером аудитории три последние цифры телефона. Вывести таблицу «Факультеты».
3. Проиндексировать таблицу «Студенты» по полям «Фамилия» и «Имя».
4. Удалить из таблицы «Студенты» данные о студентах факультета ИТР.
5. Удалить из таблицы «Приказы» приказы, изданные более пяти лет назад.

**Вариант 5**

1. Создать таблицу «Дальние поездки» и с помощью запроса внести в нее данные о выездах автомобилей на срок более 3-х дней. Вывести созданную таблицу.
2. В таблицу «Автомобили» добавить поле «Год поступления» и с помощью запроса внести в это поле данные, считая, что автомобиль поступил на предприятие через 2 года после выпуска. Вывести таблицу «Автомобили».
3. Проиндексировать таблицу «Автомобили» по полям «Год выпуска» и «Номер».
4. Удалить из таблицы «Автомобили» данные об автомобилях со сроком эксплуатации более 20 лет.
5. Удалить из таблицы «Водители» данные о водителях старше 65 лет.

**Вариант 6**

1. Создать таблицу «Архив» и с помощью запроса внести в нее данные о заселении номеров за все годы, предшествующие текущему. Вывести созданную таблицу.
2. В таблицу «Номера» добавить поле «Количество окон» и с помощью запроса внести в это поле данные, считая, что в номерах на четных этажах 2 окна, а в номерах на нечетных этажах – одно окно. Вывести таблицу «Номера».
3. Проиндексировать таблицу «Клиенты» по полям «Фамилия» и «Номер паспорта».
4. Удалить из таблицы «Состояние » данные о заселении номеров за все годы, предшествующие текущему.
5. Удалить из таблицы «Клиенты» клиентов, которых не проживали в отеле в этом году.

**Вариант 7**

1. Создать таблицу «Архив» и с помощью запроса внести в нее данные о посещении поликлиники за все годы, предшествующие текущему. Вывести созданную таблицу.
2. В таблицу «Пациенты» добавить поле «Возраст» и с помощью запроса внести в это поле число исполнившихся пациенту лет. Вывести таблицу «Карточки».
3. Проиндексировать таблицу «Пациенты» по полям «Фамилия» и «Дата рождения».
4. Удалить из таблицы «График» данные о посещении поликлиники за все годы, предшествующие текущему.
5. Удалить из таблицы «Пациенты» данные пациентов, не посещавших поликлинику более 3-х лет.

**Вариант 8**

1. Создать таблицу «Архив заказов» и с помощью запроса внести в нее данные о заказах за все годы, предшествующие текущему. Вывести созданную таблицу.
2. В таблицу «Заказы» добавить поле «Сумма» и с помощью запроса внести в это поле суммарную стоимость заказа. Вывести таблицу «Заказы».
3. Проиндексировать таблицу «Клиенты» по полям «Фамилия» и «Адрес».
4. Удалить из таблицы «Заказы» данные о заказах за все годы, предшествующие текущему.
5. Удалить из таблицы «Клиенты» данные клиентов, не делавших заказов в этом году.

**Вариант 9**

1. Создать таблицу «Архив» и с помощью запроса внести в нее данные о продажах за все годы, предшествующие текущему. Вывести созданную таблицу.
2. В таблицу «Клиенты» добавить поле «Сумма» и с помощью запроса внести в это поле сумму всех покупок клиентом. Вывести таблицу «Клиенты».
3. Проиндексировать таблицу «Клиенты» по полям «Фамилия» и «Адрес».
4. Удалить из таблицы «Журнал продаж» данные о продажах за все годы, предшествующие текущему.
5. Удалить из таблицы «Клиенты» данные клиентов, не покупавших билеты в этом году.

**Вариант 10**

1. Создать таблицу «Архив» и с помощью запроса внести в нее данные из Журнала за все годы, предшествующие текущему. Вывести созданную таблицу.
2. В таблицу «Преподаватели» добавить поле «Категория» и с помощью запроса внести в это поле число 1 для преподавателей с четным кодом преподавателя и число 2 для преподавателей с нечетным кодом преподавателя. Вывести таблицу «Преподаватели».
3. Проиндексировать таблицу «Ученики» по полям «Фамилия» и «Адрес».
4. Удалить из таблицы «Журнал» данные за все годы, предшествующие текущему.
5. Удалить из таблицы «Ученики» данные об учениках, не имеющих оценок в этом году.

**Вариант 11**

1. Создать таблицу «Архив» и с помощью запроса внести в нее данные о продажах за все годы, предшествующие текущему. Вывести созданную таблицу.
2. В таблицу «Афиша» добавить поле «Коэффициент» и с помощью запроса внести в это поле число 1.5 для спектаклей, идущих в вечернее время и 1.3 для спектаклей, идущих в дневное время. Вывести таблицу «Афиша».
3. Проиндексировать таблицу «Спектакли» по полю «Спектакль».
4. Удалить из таблицы «Продажи» данные о продажах за все годы, предшествующие текущему.
5. Удалить из таблицы «Спектакли» данные о спектаклях, информация о которых отсутствует в таблице «Афиша».

**Вариант 12**

1. Создать таблицу «Архив» и с помощью запроса внести в нее данные о размещении студентов за все годы, предшествующие текущему. Вывести созданную таблицу.
2. В таблицу «Комнаты» добавить поле «Количество окон» и с помощью запроса внести в это поле число 2 для комнат с числом мест больше 3 и число 1 для комнат с числом мест меньше или равно. Вывести таблицу «Комнаты».
3. Проиндексировать таблицу «Студенты» по полю «Фамилия».
4. Удалить из таблицы «Размещение» данные о размещении студентов за все годы, предшествующие текущему.
5. Удалить из таблицы «Студенты» данные о студентах, не проживающих в общежитии.

**Вариант 13**

1. Создать таблицу «Старые дисциплины» и с помощью запроса внести в нее данные о дисциплинах, информация о которых отсутствует в таблице «План». Вывести созданную таблицу.
2. В таблицу «План» добавить поле «Практические занятия» и с помощью запроса внести в это поле число часов, равное числу часов на лабораторные работы. Вывести таблицу «План».
3. Проиндексировать таблицу «Дисциплины» по полю «Дисциплина».
4. Удалить из таблицы «Дисциплины» данные о дисциплинах, информация о которых отсутствует в таблице «План».
5. Удалить из таблицы «Кафедры» данные о кафедрах, информация о которых отсутствует в таблице «Дисциплины».

**Вариант 14**

1. Создать таблицу «Элитная недвижимость» и с помощью запроса внести в нее данные об объектах недвижимости стоимостью свыше 120000 . Вывести созданную таблицу.
2. В таблицу «Недвижимость» добавить поле «Этажность» и с помощью запроса внести в это поле число этажей, равное сумме кода типа и кода района. Вывести таблицу «Недвижимость».
3. Проиндексировать таблицу «Клиенты» по полям «Фамилия» и «Адрес».
4. Удалить из таблицы «Недвижимость» данные об объектах недвижимости стоимостью свыше 120000 .
5. Удалить из таблицы «Типы недвижимости» данные о типах недвижимсоти, информация о которых отсутствует в таблице «Недвижимость».

**Вариант 15**

1. Создать таблицу «Архив» и с помощью запроса внести в нее данные об оказанных услугах за все месяцы, предшествующие текущему. Вывести созданную таблицу.
2. В таблицу «Услуги» добавить поле «Продолжительность» и с помощью запроса внести в это поле число 30 для услуг с четным кодом услуги и число 20 для услуг с нечетным кодом услуги. Вывести таблицу «Афиша».
3. Проиндексировать таблицу «Клиенты» по полям «Фамилия» и «Адрес».
4. Удалить из таблицы «Расписание» данные об оказанных услугах за все месяцы, предшествующие текущему.
5. Удалить из таблицы «Услуги» данные об услугах, информация о которых отсутствует в таблице «Расписание».

***Контрольные вопросы.***

1. Какие операторы языка SQL используются для изменения содержания таблиц?
2. Каким образом с помощью одного запроса можно вставить в таблицу несколько строк?
3. Что произойдет с полями строки, если их значения не указаны в операторе вставки строки?
4. Для чего нужны индексы?
5. Какие бывают индексы?
6. Как можно очистить содержание таблицы?