**Лабораторная работа № 6.**

**Модификация базы данных.**

*Цель**работы*:

Приобретение практических навыков по созданию таблиц, индексов, вводу и обновлению данных.

*Краткие теоретические сведения.*

Кроме инструкции SELECT язык манипуляции данными DML (Data Manipulation Language) содержит три другие инструкции: INSERT, UPDATE и DELETE. Подобно инструкции SELECT эти три инструкции оперируют либо таблицами, либо представлениями.

**1. Инструкция INSERT**

Инструкция INSERT вставляет строки (или части строк) в таблицу. Существует две разные формы этой инструкции:

INSERT [INTO] tab\_name [(col\_list)]

DEFAULT VALUES | VALUES ({DEFAULT NULL expression} [ ,...n])

INSERT INTO tab\_name | view\_name [(col\_list)] {select\_statement | execute\_statement}

Первая форма инструкции позволяет вставить в таблицу одну строку (или часть ее). А вторая форма инструкции INSERT позволяет вставить в таблицу результирующий набор инструкции SELECT или хранимой процедуры, выполняемой посредством инструкции EXECUTE. Хранимая процедура должна возвращать данные для вставки в таблицу. Применяемая с инструкцией INSERT инструкция SELECT может выбирать значения из другой или той же самой таблицы, в которую вставляются данные, при условии совместимости типов данных соответствующих столбцов.

Для обеих форм тип данных каждого вставляемого значения должен быть совместимы с типом данных соответствующего столбца таблицы. Все строковые и временные данные должны быть заключены в кавычки; численные значения заключать в кавычки не требуется.

**Вставка одной строки**

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

Параметр DEFAULT VALUES вставляет значения по умолчанию для всех столбцов.

столбцы с типом данных TIMESTAMP или свойством IDENTITY по умолчанию встав-ляются значения, автоматически создаваемые системой. Для столбцов других типов данных вставляется соответствующее ненулевое значение по умолчанию, если та-ково имеется, или NULL в противном случае. Если для столбца значения NULL не раз-решены 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 sample,

демонстрируя использование инструкции INSERT для вставки небольшого объема данных в базу данных.

Пример 6.1. Загрузка данных в таблицу employee

USE sample;

INSERT INTO employee VALUES (25348, 'Matthew', 'Smith','d3');

INSERT INTO employee VALUES (10102, 'Ann', 'Jones','d3');

INSERT INTO employee VALUES (18316, 'John', 'Barrimore', 'd1');

INSERT INTO employee VALUES (29346, 'James', 'James', 'd2');

INSERT INTO employee VALUES (9031, 'Elsa', 'Bertoni', 'd2');

INSERT INTO employee VALUES (2581, 'Elke' , 'Hansel', 'd2') ;

INSERT INTO employee VALUES (28559, 'Sybill', 'Moser', 'd1');

Пример 6.2. Загрузка данных в таблицу department

USE sample;

INSERT INTO department VALUES ('d1', 'Research', 'Dallas');

INSERT INTO department VALUES ('d2', 'Accounting', 'Seattle');

INSERT INTO department VALUES ('d3', 'Marketing', 'Dallas');

Пример 6.3. Загрузка данных в таблицу project

USE sample;

INSERT INTO project VALUES ('p1', 'Apollo', 120000.00);

INSERT INTO project VALUES ('p2', 'Gemini', 95000.00);

INSERT INTO project VALUES ('p3', 'Mercury', 186500.00);

Пример 6.4. Загрузка данных в таблицу works\_on

USE sample;

INSERT INTO works\_on VALUES (10102,'p1', 'Analyst', '2006.10.1');

INSERT INTO works\_on VALUES (10102, 'p3', 'Manager', '2008.1.1');

INSERT INTO works\_on VALUES (25348, 'p2', 'Clerk', '2006.2.15');

INSERT INTO works\_on VALUES (18316, 'p2', NULL, '2006.6.1');

INSERT INTO works\_on VALUES (29346, 'p2', NULL, '2006.12.15');

INSERT INTO works\_on VALUES (2581, 'p3', 'Analyst', '2006.10.15');

INSERT INTO works\_on VALUES (9031, 'p1', 'Manager', '2006.4.15');

INSERT INTO works\_on VALUES (28559, 'p1', 'NULL', '2006.8.1');

INSERT INTO works\_on VALUES (28559, 'p2', 'Clerk', '2008.2.1');

INSERT INTO works\_on VALUES (9031, 'p3', 'Clerk', '2006.11.15');

INSERT INTO works\_on VALUES (29346, 'p1','Clerk', '2006.1.4');

Существует два разных способа вставки значений в новую строку, которые показаны в примерах 6.5—6.6.

Пример 6.5. Явная вставка значения NULL

USE sample;

INSERT INTO employee VALUES (15201, 'Dave', 'Davis', NULL);

Инструкция INSERT в примере 6.5 соответствует инструкциям INSERT в примерах 6.1—6.4. Явное использование ключевого слова NULL вставляет значение NULL в соответствующий столбец.

Чтобы вставить значения в некоторые (но не во все) столбцы таблицы, обычно не-обходимо явно указать эти столбцы. Не указанные столбцы должны или разрешать значения NULL, или для них должно быть определено значение по умолчанию.

Пример 6.6. Вставка данных в часть столбцов таблицы

USE sample;

INSERT INTO employee (emp\_no, emp\_fname, emp\_lname)

VALUES (15201, 'Dave', 'Davis');

Примеры 6.5 и 6.6 равнозначны. В таблице employee единственным столбцом, разрешающим значения NULL, является столбец dept\_no, а для всех прочих столбцов это значение было запрещено предложением NOT NULL в инструкции CREATE TABLE.

Порядок значений в предложении VALUE инструкции INSERT может отличаться от порядка, указанного в инструкции CREATE TABLE. В таком случае их порядок должен совпадать с порядком, в котором соответствующие столбцы перечислены в списке столбцов.

Пример 6.6. Вставка данных в порядке, отличающемся от исходного

USE sample;

INSERT INTO employee (emp\_lname, emp\_fname, dept\_no, emp\_no) VALUES ('Davis', 'Dave', 'd1', 15201);

**Вставка нескольких строк**

Вторая форма инструкции INSERT вставляет в таблицу одну или несколько строк, выбранных подзапросом. В примере 6.8 показана вставка строк в таблицу, используя вторую форму инструкции INSERT. В данном случае выполняется запрос по выборке номеров и имен отделов, расположенных в Далласе (Dallas), и загрузка полученного результирующего набора в новую таблицу, созданную ранее.

Пример 6.8. Вставка в таблицу строк, выбранных подзапросом

USE sample;

CREATE TABLE dallas\_dept (dept\_no CHAR(4) NOT NULL, dept\_name CHAR(20) NOT NULL);

INSERT INTO dallas\_dept (dept\_no, dept\_name)

SELECT dept\_no, dept\_name

FROM department

WHERE location = 'Dallas';

Создаваемая в примере 6.8 новая таблица dallas\_dept имеет те же столбцы, что и существующая таблица department, за исключением отсутствующего столбца location. Подзапрос в инструкции INSERT выбирает в таблице department все строки, для которых значение столбца location равно 'Dallas', которые затем вставляются в созданную в начале запроса новую таблицу.

Содержимое этой таблицы можно просмотреть посредством следующей инструкции SELECT:

SELECT \* FROM dallas\_dept;

Результат выполнения этого запроса:

|  |  |
| --- | --- |
| **dept\_no** | **dept\_name** |
| d1 | Research |
| d3 | Marketing |

В примере 6.9 показан еще один способ вставки строк в таблицу, используя вторую форму инструкции INSERT. В данном случае выполняется запрос на выборку табельных номеров, номеров проектов и дат начала работы над проектом для всех сотрудников с должностью клерк (clerks), которые работают над проектом p2 с последующей загрузкой полученного результирующего набора в новую таблицу, создаваемую в начале запроса.

Пример 6.9. Вставка в таблицу строк, выбранных подзапросом с составным условием

USE sample;

CREATE TABLE clerk\_t (emp\_no INT NOT NULL, project\_no CHAR(4), enter\_date DATE);

INSERT INTO clerk\_t (emp\_no, project\_no, enter\_date)

SELECT emp\_no, project\_no, enter\_date

FROM works\_on

WHERE job = 'Clerk'

AND project\_no = 'p2';

Результатом выполнения этого запроса должна быть новая таблица clerk\_t, содержащая следующие данные:

|  |  |  |
| --- | --- | --- |
| **emp\_no** | **project\_no** | **enter\_date** |
| 25348 | p2 | 2007-02-15 |
| 28559 | p2 | 2008-02-01 |

Перед вставкой строк с помощью инструкции INSERT таблицы и clerk\_t (в примерах 6.8 и 6.9 соответственно) были пустыми. Если же таблица уже существовала и содержала строки с данными, то к ней были бы добавлены новые строки.

**Конструкторы значений таблицы и инструкция INSERT**

Конструктор значений таблицы или строки (table (row) value constructor) позволяет вставить в таблицу несколько записей (строк) посредством инструкции языка DML, такой как, например, INSERT или UPDATE. В примере 6.10 показана вставка в таблицу нескольких строк, используя такой конструктор с помощью инструкции INSERT.

Пример 6.10. Вставка строк посредством конструктора значений таблицы

USE sample;

INSERT INTO department VALUES

('d4', 'Human Resources', 'Chicago'), ('d.5', 'Distribution', 'New Orleans'), ('d6', 'Sales', 'Chicago');

В примере 6.10 инструкция INSERT одновременно вставляет три строки в таблицу department, используя конструктор значений таблицы. Как можно видеть, синтаксис этого конструктора довольно простой. 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 используется для модифицирования строк таблицы. Эта инструкция имеет следующую общую форму:

UPDATE tab\_name

{SET column\_1 = {expression | DEFAULT | NULL} [,...n]

[FROM tab\_name1 [,...n]]

[WHERE condition]

Строки таблицы tab\_name выбираются для изменения в соответствии с условием в предложении WHERE. Значения столбцов каждой модифицируемой строки изменяются с помощью предложения SET инструкции UPDATE, которое соответствующему столбцу присваивает выражение (обычно) или константу. Если предложение WHERE отсутствует, то инструкция UPDATE модифицирует все строки таблицы.

В примере 6.11 инструкция UPDATE изменяет всего лишь одну строку таблицы works\_on, поскольку комбинация столбцов emp\_no и project\_no является первичным ключом этой таблицы и, следственно, она однозначна. В данном примере изменяется должность сотрудника, значение которого было ранее неизвестно или имело значение NULL.

Пример 6.11. Присвоение сотруднику с табельным номером 18316, который работает над проектом p2, должности менеджера (manager)

USE sample;

UPDATE works\_on

SET job = 'Manager'

WHERE emp\_no = 18316

AND project\_no = 'p2';

В примере 6.12 значения строкам таблицы присваиваются посредством выражения. Запрос пересчитывает бюджеты всех проектов с долларов н белорусские рубли. Валютный курс: 1,93 руб. за $1.

Пример 6.12. Присвоение значений посредством выражения

USE sample;

UPDATE project

SET budget = budget\*1,93;

В данном примере изменяются все строки таблицы project, поскольку в запросе отсутствует предложение WHERE. Измененное содержимое таблицы project можно просмотреть с помощью следующего запроса:

USE sample;

SELECT \* FROM project;

Результат выполнения этого запроса:

|  |  |  |
| --- | --- | --- |
| **project\_no** | **project\_name** | **budget** |
| pl | Apollo | 61200 |
| p2 | Gemini | 48450 |
| p3 | Mercury | 95115 |

В примере 6.13 в предложении WHERE инструкции UPDATE используется вложенный запрос. Поскольку применяется оператор IN, то этот запрос может возвратить более одной строки.

Пример 6.13. В связи с болезнью сотрудницы Jones во всех ее проектах в столбце ее должности присваивается значение NULL

USE sample;

UPDATE works\_on

SET job = NULL

WHERE emp\_no IN ( SELECT emp\_no FROM employee WHERE emp\_lname = 'Jones');

Запрос в примере 6.13 можно также выполнить посредством предложения FROM инструкции UPDATE. В предложении FROM указываются имена таблиц, которые обрабатываются инструкцией UPDATE. Все эти таблицы должны быть в дальнейшем соединены. Применение предложения FROM показано в примере 6.14. Логически, этот пример идентичен предыдущему примеру 6.13.

Пример 6.14. Использование предложения FROM для выполнения аналогичного запроса из примера 6.13

USE sample;

UPDATE works\_on

SET job = NULL

FROM works\_on, employee

WHERE emp\_lname = 'Jones'

AND works\_on.emp\_no = employee.emp\_no;

**3. Инструкция DELETE**

Инструкция DELETE удаляет строки из таблицы. Подобно инструкции INSERT, эта инструкция также имеет две различные формы:

DELETE FROM table\_name

[WHERE predicate];

DELETE table\_name

FROM table\_name [,...n] [WHERE condition];

Удаляются все строки, которые удовлетворяют условие в предложении WHERE. Явно перечислять столбцы в инструкции DELETE не то чтобы нет необходимости, а даже не разрешается, поскольку эта инструкция оперирует строками, а не столбцами.

Использование первой формы инструкции DELETE показано в примере 6.15.

Пример 6.15. Удаление из таблицы works\_on всех сотрудников, имеющих должность менеджера (manager)

USE sample;

DELETE FROM works\_on

WHERE job = 'Manager';

Предложение WHERE инструкции DELETE может содержать вложенный запрос, как это показано в примере 6.16.

Пример 6.16. Поскольку сотрудница Moser уволилась, из базы данных удаляются все записи, связанные с ней

USE sample;

DELETE FROM works\_on

WHERE emp\_no IN

(SELECT emp\_no FROM employee WHERE emp\_lname = 'Moser');

DELETE FROM employee

WHERE emp\_lname = 'Moser';

Запрос из примера 6.16 можно также выполнить с помощью предложения FROM, как это показано в примере 6.16. В данном случае семантика этого предложения такая же, как и предложения FROM в инструкции UPDATE.

Пример 6.16. Удаление записей из всей базы данных посредством предложения FROM

USE sample;

DELETE works\_on

FROM works\_on, employee

WHERE works\_on.emp\_no = employee.emp\_no

AND emp\_lname = 'Moser';

DELETE FROM employee

WHERE emp\_lname = 'Moser';

Использование предложения WHERE в инструкции DELETE не является обязательным. Если это предложение отсутствует, то из таблицы удаляются все строки.

**4. Другие инструкции и предложения Transact-SQL для модификации таблиц**

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

инструкцию TRUNCATE TABLE;

инструкцию MERGE;

предложение OUTPUT.

**4.1. 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 table\_name

**4.2. Инструкция MERGE**

Инструкция MERGE объединяет последовательность инструкций INSERT, UPDATE и DELETE в одну элементарную инструкцию, в зависимости от существования записи (строки). Иными словами, можно синхронизировать две разные таблицы, чтобы модифицировать содержимое таблицы назначения в зависимости от различий, обнаруженных в таблице-источнике.

Основной областью применения для инструкции MERGE является среда хранилищ данных , где таблицы необходимо периодически обновлять, чтобы отражать новые данные, прибывающие с систем оперативной обработки транзакций OLTP (On-Line Transaction Processing). Эти данные могут содержать изменения существующих строк таблиц и/или новые строки, которые нужно вставить в таблицы. Если строка в новых данных соответствует записи, которая уже имеется в таблице, выполняется инструкция UPDATE или DELETE. В противном случае выполняется инструкция INSERT.

Альтернативно, вместо инструкции MERGE можно использовать последовательность инструкций INSERT, UPDATE и DELETE, в которых для каждой строки решается, какую операцию выполнять: вставку, удаление или обновление. Но этот подход имеет значительный недостаток, связанный с производительностью: в нем требуется выполнять несколько проходов по данным, а данные обрабатываются по принципу "запись за записью".

Использование инструкции MERGE показано в примерах 6.18 и 6.19. В примере 6.18 создается таблица bonus, содержащая одну строку со значениями p1 и 100.

Пример 6.18. Создание однострочной таблицы

USE sample;

CREATE TABLE bonus

(pr\_no CHAR(4),

bonus SMALLINT DEFAULT 100);

INSERT INTO bonus (pr\_no) VALUES ('p1');

В созданную в примере 6.18 таблицу bonus в примере 6.19 будут вставлены новые данные из другой таблицы.

Пример 6.19. Вставка новых строк данных в таблицу bonus

USE sample;

MERGE INTO bonus B

USING (SELECT project\_no, budget FROM project) E

ON (B.pr\_no = E.project\_no)

WHEN MATCHED THEN

UPDATE SET B.bonus = E.budget \* 0.1

WHEN NOT MATCHED THEN

INSERT (pr\_no, bonus)

VALUES (E.project\_no, E.budget \* 0.05);

В примере 6.19 инструкция MERGE в зависимости от значений в столбце pr\_no модифицирует данные в таблице bonus. Если в столбце pro\_no таблицы bonus имеется значение из столбца project\_no таблицы project, то выполняется ветвление MATCHED и существующее значение будет обновлено. В противном случае выполняется ветвление NON MATCHED и соответствующая инструкция INSERT вставляет новые строки в таблицу bonus.

После выполнения этой инструкции содержимое таблицы bonus будет выглядеть следующим образом:

|  |  |
| --- | --- |
| **pr\_no** | **bonus** |
| Pl | 12000 |
| p2 | 4750 |
| p3 | 9325 |

По этому результату можно видеть, что значение столбца bonus представляет 10% исходного значения при использовании инструкции UPDATE и 5% при использовании инструкции INSERT.

**4.3. 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 вашим требованиям, 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 6.20 показано использование инструкции OUTPUT с инструкцией DELETE.

Пример 6.20. Применение инструкции OUTPUT

USE sample;

DECLARE @del\_table TABLE (emp\_no INT, emp\_lname CHAR(20));

DELETE employee

OUTPUT DELETED.emp\_no, DELETED.emp\_lname INTO @del\_table

WHERE emp\_no > 15000;

SELECT \* FROM @del\_table

При условии, что содержимое таблицы находится в исходном состоянии, выполнение запроса в примере 6.20 дает следующий результат:

|  |  |
| --- | --- |
| **emp\_no** | **emp\_lname** |
| 25348 | Smith |
| 18316 | Barrimore |
| 29346 | James |
| 28559 | Moser |

В примере 6.20 сначала объявляется табличная переменная @del\_table с двумя столбцами: emp\_ no и emp\_lname. В этой таблице будут сохранены удаленные строки. Синтаксис инструкции DELETE расширен предложением OUTPUT:

OUTPUT DELETED.emp\_no, DELETED.emp\_lname INTO @del\_table

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

В примере 6.21 показано использование предложения OUTPUT в инструкции UPDATE.

Пример 6.21. Инструкция UPDATE с предложением OUTPUT

USE sample;

DECLARE @update\_table TABLE

(emp\_no INT, project\_no CHAR(20),old\_job CHAR(20),new\_job CHAR(20));

UPDATE works\_on

SET job = NULL

OUTPUT DELETED.emp\_no, DELETED.project\_no,

DELETED.job, INSERTED.job INTO @update\_table

WHERE job = 'Clerk'; SELECT \* FROM @update\_table

Результат выполнения этого запроса:

|  |  |  |  |
| --- | --- | --- | --- |
| **emp\_no** | **project\_no** | **Oldjob** | **newjob** |
| 25348 | p2 | Clerk | NULL |
| 28559 | p2 | Clerk | NULL |
| 9031 | p3 | Clerk | NULL |
| 29346 | pi | Clerk | NULL |

В следующих примерах приводится использование предложения OUTPUT с инструкцией MERGE.

Допустим, что маркетинговый отдел решил предоставить клиентам скидку в 20% на все велосипеды ("Bikes") стоимостью более $500. В примере 6.22 инструкция SELECT выбирает все продукты стоимостью свыше $500 и вставляет их во временную таблицу temp\_PriceList. А последующая инструкция UPDATE выбирает из всех отобранных товаров велосипеды и уменьшает их цену.

Пример 6.24. Уменьшение цены велосипедов на 20%

USE AdventureWorks;

SELECT ProductID, Product.Name as ProductName, ListPrice

INTO temp\_PriceList

FROM Production.Product

WHERE ListPrice > 500;

UPDATE temp\_PriceList

SET ListPrice = ListPrice \* 0.8

WHERE ProductID IN (SELECT ProductID

FROM AdventureWorks.Production.Product

WHERE ProductSubcategoryID IN (SELECT ProductCategoryID

FROM AdventureWorks.Production.ProductSubcategory

WHERE ProductCategoryID IN (SELECT ProductCategoryID

FROM AdventureWorks.Production.ProductCategory

WHERE Name = 'Bikes')));

В примере 6.23 инструкция CREATE TABLE создает новую таблицу, temp\_Difference, в которой будет сохраняться результирующий набор инструкции MERGE. После этого инструкция MERGE сравнивает полный список продуктов с новым списком (находящимся в таблице temp\_PriceList) и вставляет модифицированные цены для всех велосипедов, используя предложение UPDATE SET. (Кроме вставки новых цен для всех велосипедов, эта инструкция также изменяет значения столбца ModifiedDate для всех продуктов на текущую дату.) Предложение OUTPUT в примере 6.23 записывает старые и новые цены во временную таблицу temp\_Difference. Это в дальнейшем (при необходимости) позволит подсчитать агрегатные различия.

Пример 6.23. Инструкция MERGE с предложением UPDATE

USE AdventureWorks;

CREATE TABLE temp\_Difference

(old DEC (10,2), new DEC(10,2));

GO

MERGE INTO Production.Product

USING temp\_PriceList ON Product.ProductID = temp\_PriceList.ProductID WHEN MATCHED AND Product.ListPrice <> temp\_PriceList.ListPrice THEN UPDATE SET ListPrice = temp\_PriceList.ListPrice, ModifiedDate = GETDATE()

WHEN NOT MATCHED BY SOURCE THEN UPDATE SET ModifiedDate = GETDATE()

OUTPUT DELETED.ListPrice, INSERTED.ListPrice INTO temp\_Difference1;

В примере 6.24 показано вычисление общих различий, полученных в результате предыдущих модификаций.

Пример 6.24. Вычисление общих различий

USE AdventureWorks;

SELECT SUM(old) — SUM(new) AS diff

FROM AdventureWorks.dbo.temp\_Difference;

Результат выполнения этого запроса:

|  |
| --- |
| **Diff** |
| 10773.60 |

**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?