**Технологическая карта № 8**

**Наименование:** Запросы на добавление данных.

**1. Цель работы:**

1.1. Научиться выполнять запросы – действия.

1.2. Формировать компетенции: ОК 2, ОК 4, ОК 5, ОК 9, ПК 2.3, ПК 2.4.

**2. Подготовка к работе:**

2.1. Изучить предложенную литературу.

2.2. Подготовить бланк отчёта.

**3. Литература:**

3.1. Малбасарова Г.Х. «Информационные технологии и платформы разработки информационных систем». Учебное пособие, 2015

**4. Перечень необходимого оборудования:**   
4.1. Персональный IBM PC.

**5. Задание на работу:**

**Часть 1**

5.1. Создайте новую базу данных:

5.1.1 Пропишите запрос на создание БД «Фамилия\_склад» через команду CREATE DATABASE

5.1.2 Самостоятельно пропишите запросы на создание таблиц через команду CREATE TABLE. Задайте типы данных и первичные ключи.

5.1.3 Создайте диаграмму базы данных.

5.1.4 Заполните таблицы данными, приведенными в Приложении.

**Часть 2**

5.2. Добавьте в таблицу tovar новую запись.

5.3. Создайте таблицу itog, в которую добавляются записи, выбранные запросом.

5.4. Увеличьте цену товара на 10% для поставщика с кодом ‘11’.

5.5. Удалите из таблицы расход все записи за 2 октября.

5.6. В таблицу ‘поставщик’ добавьте новую запись о поставщике с кодом «17», «Лесная ягода».

5.7. В таблицу Rashod добавьте новую запись о продаже товара с кодом ‘001’. Выведите содержимое таблице до добавления и после добавления записи.

5.8. Увеличьте количество поставляемого товара на 5 единиц для номера поставки ‘04’. Выведите содержимое таблице до обновления и после обновления.

5.9. Из таблицы Rashod удалите все записи с количеством товара меньше 15. Выведите содержимое таблице до удаления и после удаления.

5.10. Создайте таблицу ‘лучший товар’ с полями kod\_tov, kol. Выполните последовательность команд: **удалите** все имеющиеся записи из таблицы ‘лучший товар’; **добавьте** в таблицу ‘лучший товар’ код товара и его количество, который больше всего был отпущен со складов (по таблице rashod, вложенный запрос на нахождение максимума); **выведите** на экран содержимое таблицы ‘лучший товар’.

**6. Порядок проведения лабораторной работы:**

6.1. Получить допуск к работе.

6.2. Выполнить задания в соответствии с приложением.

6.3. Ответить на контрольные вопросы.

6.4. Оформить отчет.

**7. Содержание отчета:**

7.1. Наименование и цель работы.

7.2. Запись результата выполненных заданий в отчёт.

7.3. Ответы контрольные вопросы.

7.4. Выводы о проделанной работе.

**8. Контрольные вопросы**:

8.1. Какие запросы называются запросами-действиями?

8.2. Как выполняется добавление данных?

8.3 Как выполняется изменение данных?

8.4. Как выполняется удаление данных?

**ПРИЛОЖЕНИЕ**

**ЧАСТЬ 1**

1. ***Создание базы данных***

Простую базу данных (без указания дополнительных параметров) можно создать, выполнив следующую команду:

**CREATE DATABASE** Имя\_таблицы

1. ***Создание таблиц***

Перед созданием таблиц БД необходимо продумать определение всех столбцов таблицы и характеристик каждого столбца (таких, как тип, длина, обязательность для ввода, ограничения, накладываемые на значения и пр.), индексов, ограничений целостности по отношению к другим таблицам.

БД, в которую будет добавлена создаваемая таблица, должна быть открыта, т.е. с ней должно быть установлено активное соединение.

**Создание таблицы БД** осуществляется оператором CREATE TABLE. Синтаксис:

**CREATE TABLE** Имя\_Таблицы (

Имя\_столбца1 тип данных,

Имя\_столбца2 тип данных,

… ,

PRIMARY KEY(Имя\_столбца)

)

Более полный синтаксис задания столбца таблицы:

имя\_столбца{тип\_данных}[DEFAULT] [NOT NULL] [<огранич\_столбца>]

[COLLATE collation]

* *DEFАULТ* определяет значение, которое по умолчанию заносится в столбец при создании записи таблицы; это значение будет присутствовать в соответствующем столбце данной записи до тех пор, пока пользователь не изменит его каким-либо образом; значения по умолчанию.
* NOT NULL — это ограничения, указывающие, что эти поля не могут содержать значение NULL при создании записей в этой таблице.
* *<огранич\_столбца>* - ограничения, накладываемые на значения столбца.
* *COLLATE collation* - определяет порядок сортировки символов.

***Первичный ключ***

При создании таблицы желательно, чтобы она имела уникальный столбец или же совокупность столбцов, которая уникальна для каждой ее строки – по данному уникальному значению можно однозначно идентифицировать запись. Такое значение называется первичным ключом таблицы.

Создать первичный ключ можно командой PRIMARY KEY ( ) в момент создания таблиц.

***Типы данных***

Тип данных – характеристика, которая говорит о том какого рода данные может хранить столбец таблицы.

Основные типы данных используемые в MS SQL:

|  |  |  |
| --- | --- | --- |
| **Значение** | **Обозначение в MS SQL** | **Описание** |
| Строка переменной длины | varchar(N) | При помощи числа N, мы можем указать максимально возможную длину строки для соответствующего столбца. Например, если мы хотим сказать, что значение столбца «ФИО» может содержать максимум 30 символов, то необходимо задать ей тип varchar(30). |
| Целое число | int | Данный тип позволяет нам использовать в столбце только целые числа, как положительные, так и отрицательные. Для справки (сейчас это не так актуально для нас) – диапазон чисел который позволяет тип int от -2 147 483 648 до 2 147 483 647. Обычно это основной тип, который используется для задания идентификаторов. |
| Вещественное или действительное число | float | Если говорить простым языком, то это числа, в которых может присутствовать десятичная точка (запятая). |
| Дата | date | Если в столбце необходимо хранить только Дату, которая состоит из трех составляющих: Числа, Месяца и Года. Например, 15.02.2014 (15 февраля 2014 года). Данный тип можно использовать для столбца «Дата приема», «Дата рождения». |
| Флаг | bit | Данный тип удобно применять для хранения значений вида «Да»/«Нет», где «Да» будет сохраняться как 1, а «Нет» будет сохраняться как 0. |

*Задание 1.*

1.1 Создайте базу данных «Фамилия\_склад».

1.2 Создайте следующие таблице в этой БД:

* Post (kod\_post, name),
* Tovar (kod\_tov, zena),
* Postavka (n\_post, data, kol, kod\_tov, n\_sklad),
* Rashod (n\_rash, data, kol, kod\_tov, n\_sklad)

Типы данных для полей:

* kod\_post, kod\_tov, n\_post, n\_rash -  тип данных varchar(5) not null
* name - тип данных varchar(50)
* zena - тип данных float
* data - тип данных varchar(50)
* kol - тип данных int
* n\_sklad - тип данных int или varchar(5)

Пример создания таблицы:

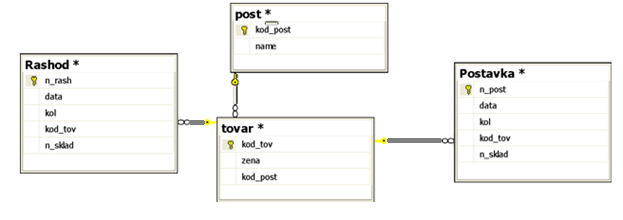
CREATE TABLE Post (

kod\_post varchar(10) NOT NULL,

name varchar(50),

PRIMARY KEY(kod\_post))

1.3 Создайте диаграмму базы данных:



1.4 Заполните таблицы следующими данными:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tovar** | | |  | **post** | |
| *kod\_tov* | *kod\_post* | *zena* |  | *kod\_post* | *name* |
| 001 | 12 | 120 |  | 11 | «Весёлый молочник» |
| 002 | 11 | 300 |  | 12 | «Сластёна» |
| 003 | 14 | 340 |  | 13 | «Колокольчик» |
| 004 | 11 | 100 |  | 14 | «Южная ночь» |

**Postavka**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n\_post | data | kol | kod\_tov | n\_sklad |
| 01 | 11 сентября | 100 | 003 | 3 |
| 02 | 11 сентября | 200 | 001 | 2 |
| 03 | 13 сентября | 50 | 003 | 1 |
| 04 | 16 сентября | 130 | 002 | 1 |
| 05 | 16 сентября | 230 | 004 | 2 |
| 06 | 18 сентября | 70 | 003 | 2 |
| 07 | 20 сентября | 200 | 001 | 2 |

**Rashod**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n\_rash | data | kol | kod\_tov | n\_sklad |
| 111 | 1 октября | 30 | 003 | 1 |
| 112 | 1 октября | 10 | 004 | 2 |
| 113 | 2 октября | 50 | 001 | 2 |
| 114 | 3 октября | 20 | 004 | 2 |
| 115 | 4 октября | 15 | 002 | 1 |

**ЧАСТЬ 2**

Добавление, изменение и удаление записей выполняется запросами действиями. Эти запросы не выводят данные на экран, а производят их изменение. Если необходимо просмотреть измененные данные, то после запроса – действия, выполните запрос выборку.

Язык SQL ориентирован на выполнение операций над группами записей, хотя в некоторых случаях операция может проводиться и над отдельной записью. Поэтому неудивительно, что операторы добавления, изменения и удаления записей в общем случае вызывают соответствующие операции над группами записей.

***1.Оператор INSERT.*** Оператор INSERТ применяется для добавления записей в объект. В качестве объекта может выступать таблица БД

Формат оператора INSERT:

**INSERT INTO <объект> [(столбец1[(, столбец2 ...] )]**

**{VALUES «значение1> (, <значение2> ...)) | <оператор SELECT> }**

Список столбцов указывает *столбцы,* которым будут присвоены *значения* в добавляемых записях. Список столбцов может быть опущен. В этом случае подразумеваются все столбцы объекта, причем в том порядке, в котором они определены в данном объекте.

Поставить в соответствие столбцам списки значений можно двумя способами. Первый состоит в явном указании значений после слова VALUES, второй - в формирован значений при помощи оператора SELECT.

Явное указание списка значений. В этом случае оператор INSERТ при меняется для добавления одной записи и имеет формат:

INSERT INTO <объект> (столбец1 (, столбец2 ...)) VALUES «значение1> (, <значение2> ...)

*Значения* назначаются *столбцам* по порядку следования тех и других в операторе: первому по порядку столбцу назначается первое значение, второму столбцу - второе значение и Т.д.

*Задание 2.* Добавить в таблицу tovar новую запись:

INSERT INTO tovar (kod\_tov, zena, kod\_post) VALUES (‘005’, 350, ‘12’)

Поскольку столбцы таблицы RASHOD указаны в полном составе и именно в том порядке, в котором они перечислены при создании таблицы tovar оператором CREAТ TЕABLE, оператор можно упростить:

INSERT INTO tovar VALUES (‘005’, 350, ‘12’)

Указание значений при помощи оператора SELECT:

**INSERT INTO <объект> (столбец1 (, столбец2 ...)) <оператор SELECT>**

При этом *значениями,* которые присваиваются *столбцам,* являются значения, возвращаемые оператором SELECT. Порядок их назначения столбцам аналогичен предыдущей форме оператора INSERТ: значение первого по порядку столбца результирующего набора данных оператора SELECT присваивается первому столбцу оператора INSERT, второй - второму и Т.Д. Следует обратить внимание на важную особенность: поскольку оператор SELECT в общем случае возвращает множество записей, то и оператор INSERT в данной форме приведет к добавлению в *объект* аналогичного количества новых записей.

*Задание 3.* Создайте таблицу itog, в которую добавляются записи, выбранные запросом.

Создание таблицы:

use sklad

*create table* itog(

data varchar(15),

kol int )

В эту таблицу добавляется записи за каждую дату с суммарным количеством поступившего товара.

use sklad

*INSERT INTO* itog

*SELECT* data,sum (kol)

*FROM* postavka

*group by* data

***1.Оператор UPDAТЕ.*** Оператор UPDAТЕ применяется для изменения значения в группе записей или - в частном случае - в одной записи объекта. В качестве объекта могут выступать ТБД или просмотр, созданный оператором CREAТЕ VIEW. В последнем случае могут изменяться значения записей из нескольких таблиц.

Формат оператора UPDAТЕ:

**UPDAТЕ <объект>**

**SET столбец1 = <значение1> (,столбец2 = <значение2>...)**

**(WНЕRE <условие поиска > )**

При корректировке каждому из перечисленных *столбцов* присваивается соответствующее *значение.* Корректировка выполняется для всех записей, удовлетворяющих *условию поиска.* Условие поиска задается так же, как в операторе SELECT

*Задание 4.* Увеличьте цену товара на 10% для поставщика с кодом ‘11’

use sklad

*select* \*

*from* tovar

*where* kod\_post='11'

*update* tovar

*set* zena=zena+50

*where* kod\_post='11'

*select* \*

*from* tovar

*where* kod\_post='11'

Перед выполнением запроса – действия запускается запрос на выборку, чтобы посмотреть состав данных перед обновлением. После выполнения обновления снова выведем данные для просмотра.

***3.Оператор DELETE.*** Оператор DELETE предназначен для удаления группы записей из объекта. В качестве объекта могут выступать ТБД или просмотр, созданный оператор CREATE VIEW..

Формат оператора DELETE:

**DELETE FROM <объект> [WHERE <условие поиска>];**

Удаляются все записи из *объекта,* удовлетворяющие *условию* Условие поиска задается так же, как в операторе SELECT.

*Задание 5.*Удалите из таблицы расход все записи за 2 октября.

use sklad

*select* \*

*from* rashod

*delete from* rashod

*where* data = '2 октября'

*select* \*

*from* rashod

Перед выполнением запроса – действия запускается запрос на выборку, чтобы посмотреть состав данных перед обновлением. После выполнения обновления снова выведем данные для просмотра.