РОСЖЕЛДОР
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Ростовский государственный университет путей сообщения»
(ФГБОУ ВО РГУПС)

Гальцева А.А.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ И САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ ПО ДИСЦИПЛИНЕ

ОП.04 «Основы проектирования баз данных». Часть 2

для специальности
09.02.09 Веб-разработка

Ростов-на-Дону
РГУПС
2025

СОДЕРЖАНИЕ

Лабораторная работа № 1

Знакомство с языком SQL. Работа с консольным клиентом и графическим интерфейсом.

Цель лабораторной работы: Изучение структурированного языка запросов Transact - SQL, являющегося основой системы программирования SQL Server, и приобретение навыков применение инструментальных средств разработки и программирования объектов создаваемых баз данных.

Методические указания

Основы программирования на Transact - SQL

Система программирования SQL Server относится к классу командно-интерпретирующих систем сверхвысокого уровня. Единицами действий системы являются команды, исполняемые в режиме интерпретации сразу же по мере их поступления в сервер. Основой этой системы программирования является проблемно-ориентированный структурированный язык запросов (Structured Query Language) Transact - SQL, который расширяется и развивает возможности стандарта ANSI SQL - 92.

Transact - SQL включает следующие средства:

1. данные различного типа баз данных и переменных;

2. константы, стандартные и ограниченные идентификаторы;

3. арифметические и логические выражения, включающие следующие операнды: константы, переменные, имена столбцов таблиц, функции, подзапросы и условные выражения, а также выражения, взятые в круглые скобки;

4. SQL - команды для создания, изменения и удаления баз данных и их объектов, а также для определения запросов на ввод, обработку и извлечение данных;

5. управляющие программные структуры, определяющие условия и порядок выполнения команд в заданной последовательности или пакете команд;

6. встроенные (системные) и определяемые пользователем функции;

7. встроенные (системные) и определяемые пользователем хранимые процедуры.

В системе могут храниться, помимо функций и процедур,

последовательности (пакеты) команд, которые называются скриптами. Если скрипт описывает процесс создания базы данных, или каких-либо ее объектов, то такой скрипт называется сценарием. Сценарии позволяют переносить структуру базы данных от одного сервера к другому, а также структуру таблиц и других объектов в различные базы данных. Скрипты хранятся в текстовых файлах.

Функции и хранимые процедуры баз данных позволяют уменьшить объем запросов, передаваемых от клиента к серверу, что повышает общую производительность системы. Наличие исходного кода для этих объектов позволяет упростить сопровождение программных комплексов и внесение изменений в них.

Обычно все бизнес-правила и алгоритмы обработки данных реализуются на сервере баз данных и доступны конечному пользователю в виде набора функций и хранимых процедур, которые и представляют интерфейс обработки данных. Для обеспечения целостности данных, а также в целях безопасности, приложению обычно не предоставляется прямой доступ к данным. Вся работа ведется с помощью указанного интерфейса.

Подобный подход делает весьма простым изменение алгоритмов обработки данных и обеспечивает возможность расширения системы без внесения изменений в само приложение. Достаточно изменить хранимую процедуру на сервере баз данных, и сделанные изменения тотчас станут доступными всем пользователям сети.

В языке Transact - SQL имеются следующие виды констант:

1. битовые: 0 и 1;

2. логические: FALSE и TRUE;

3. бинарные в шестнадцатеричном представлении: 0*9E70DA;

4. символьные: 'ABC'; "ABC" (если QUOTEDIDENTIFIER = OFF); N 'ABC' (Unicode); N "ABC" (Unicode);

5. целые: 1; 2; 175;

6. с фиксированной точкой: 12.35; - 16.753;

7. с плавающей точкой: 1.75E5; 3.84E - 3;

8. для даты: " April 15.2003"; "4/15/2003"; "20031207";

9. для времени: 14:30; 14:30:20:999; 4am; 4pm;

10. денежные: $100;?200; 2.15.

Комментарии в языке бывают двух типов: сточные, начинающиеся с двух символов минуса - и блочные, заключаемые символами /* и */.

Все объекты базы данных должны иметь имена, которые используются в командах для ссылки на эти объекты. Любой объект базы данных должен быть уникально идентифицирован. Помимо программных имен сервер автоматически генерирует внутренние уникальные имена для идентификации объектов баз данных, например, PK Table X 014543FA.

Программные имена задаются идентификаторами двух типов:

1. стандартными идентификаторами: Table X; Key Col;

2. ограниченными идентификаторами: [My Table]; [Order]; "My Table"; "Order" (если QUOTEDIDENTIFIER = ON).

Длина идентификатора - от 1 до 128 символов.

Идентификатором не может быть какое-либо зарезервированное ключевое слово языка.

Стандартный идентификатор в качестве первого символа может иметь любую латинскую или русскую букву, знаки #, ##, @, @@ и знак подчеркивания _. Последующими знаками, помимо указанных, могут быть еще и десятичные цифры.

Ограниченные идентификаторы могут включать и другие символы, в том числе зарезервированные слова. В этом случае они должны заключаться в квадратные скобки или двойные кавычки.

В соответствии с идеологией SQL Server каждый объект создается определенным пользователем и принадлежит той или иной базе данных. В свою очередь база данных Комментарии в языке бывают двух типов: сточные, начинающиеся с двух символов минуса - и блочные, заключаемые символами /* и */.

Все объекты базы данных должны иметь имена, которые используются в командах для ссылки на эти объекты. Любой объект базы данных должен быть уникально идентифицирован. Помимо программных имен сервер автоматически генерирует внутренние уникальные имена для идентификации объектов баз

данных, например, PK Table X 014543FA.

Программные имена задаются идентификаторами двух типов:

1. стандартными идентификаторами: Table X; Key Col;

2. ограниченными идентификаторами: [My Table]; [Order]; "My Table"; "Order" (если QUOTEDIDENTIFIER = ON).

Длина идентификатора - от 1 до 128 символов.

Идентификатором не может быть какое-либо зарезервированное ключевое слово языка.

Стандартный идентификатор в качестве первого символа может иметь любую латинскую или русскую букву, знаки #, ##, @, @@ и знак подчеркивания _. Последующими знаками, помимо указанных, могут быть еще и десятичные цифры.

Ограниченные идентификаторы могут включать и другие символы, в том числе зарезервированные слова. В этом случае они должны заключаться в квадратные скобки или двойные кавычки.

В соответствии с идеологией SQL Server каждый объект создается определенным пользователем и принадлежит той или иной базе данных. В свою очередь база данных расположена на конкретном сервере. Из имен объекта, пользователя, базы данных и сервера создается полное имя (complete name) или полностью определенное имя (full qualified name), записываемое в следующем виде:

[[[server.].[database].[owner_name].] objectname

Варианты обращения к объектам базы данных:

A.B.C.D; A.B..D; A..C.D; A..D; B.C.D; B..D; C.D; D

Чтобы сослаться на конкретный столбец таблицы или представления, необходимо в полном имени указать пятый элемент: A.B.C.E.

Операторами выражения могут быть унарные (+ и -), бинарные арифметические операторы (+, -, *, %), оператор присваивания (=), строковая операция конкатенации (+), операторы сравнения (=, >, <, <=, >=, =, != или <>, !<, !>), логические операторы (NOT, AND, OR, ALL, ANY, BETWEEN, EXIST, IN, LIKE, SOME) и битовые операторы (&, |, ^).

Константы, переменные, операнды и выражения используются при записи

команд и программирования функций и хранимых процедур, которые, все вместе взятые, составляют основную часть системы программирования SQL Server и определяют ее выразительную мощь. Команды позволяют создавать, модифицировать и удалять базы данных и их объекты, формировать сложные запросы на ввод, обработку и извлечение данных из баз знаний, выполнять функции администрирования и обслуживания баз данных. Функции и хранимые процедуры реализуют разнообразные алгоритмы обработки данных или выполнение служебных функций сервера.

При формировании запросов очень часто используются специальные логические операторы, синтаксис которых записывают следующим образом:

1. Выражение {= | < > | ! = | > | >= | ! >|, = | !<} ALL подзапрос.

Здесь скалярное выражение вычисляется и сравнивается с каждым значением, возвращаемым подзапросом. Если сравнение дает истину для всех возвращаемых подзапросом значений, то этот оператор возвращает истину.

2. Если вместо ALL записать SOME или ANY, то результатом будет истина, если хотя в одной строке будет выполняться заданное сравнение.

3. Выражение [NOT] BETWEEN Н Выражение AND В Выражение возвращает истину, когда значение выражения лежит в диапазоне значений Н выражения и В Выражения (или не лежит).

4. Оператор EXISTS (подзапрос) возвращает значение истина, если подзапрос возвращает хотя бы одну строку.

5. Выражение [NOT] IN (подзапрос \ выражение [,.. n]) возвращает значение истина, если значение левого выражения совпадает с одним из значений подзапроса или списка значений правых выражений (или не совпадает).

6. Выражение [NOT] LIKE шаблон [ESCAPE знак] дает истину, если значение выражения соответствует или не соответствует шаблону, в котором "%" означает любое количество произвольных символов, "_" - один произвольный символ, "[символы]" - один из указанных в скобках, "[символы]" - все символы, кроме указанных. Знак после слова ESCAPE позволяет указать, что следующий за ним знак шаблона не является управляющим знаком шаблона, т.е. знаком "%", "_" и т.д., а представляет обычный знак строки.

Раздел документации сервера T - SQL Help содержит описание каждой команды языка Transact - SQL B и набор примеров их использования. Синтаксис команды определяется с помощью специального метаязыка, основанного на нормальных формах Бекуса Наура (БНФ).

Ключевые слова определения без всякого изменения переходят в саму команду; метапеременные, написанные курсивом, должны быть заменены программными именами; разделители (запятая, равно, апостроф и т.д.) также переходят в качестве разделителей в команду.

Метасинтаксические знаки имеют следующий смысл:

::= — есть по определению;

| — выбор альтернативы;

[ ] — возможное отсутствие части определения;

{ } — объединение частей определения для выюора или повторения;

[,...n] — повторение предшествующей части 1, 2, ..., n раз с разделителем запятая для этой части (разделитель может быть любой;)

<...> — метапеременная, которая имеет свое определение.

Для выполнения операций с базой данных при проведении лабораторных работ предлагается использовать программу " SQL Server Management Studio Rus", представляющую собой наиболее распространенное и удобное средство администрирования баз данных под управлением MS SQL Server (Среда Management Studio доступна для свободной загрузки из центра загрузки Майкрософт).

Среда SQL Server Management Studio — это интегрированная среда для доступа, настройки, управления, администрирования и разработки всех компонентов SQL Server. Среда SQL Server Management Studio объединяет большое число графических средств с набором полнофункциональных редакторов сценариев для доступа к SQL Server разработчиков и администраторов с любым опытом работы.

Среда SQL Server Management Studio обеспечивает следующие основные возможности:

- поддерживает большинство административных задач для SQL Server;

- единая интегрированная среда для управления SQL Server Database Engine и разработки;
- новые управляющие диалоговые окна для управления объектами в компоненте SQL Server Database Engine, службах Analysis Services, Reporting Services, Notification Services и выпуске SQL Server Compact 3.5 с пакетом обновления 1 (SP1), позволяющие выполнять действия немедленно, направлять их в редактор кода или включать эти действия в сценарий для последующего выполнения;
- экспорт и импорт регистрации сервера среды SQL Server Management Studio из одной среды Management Studio в другую;
- сохранение и печать XML-файлов плана выполнения и взаимоблокировок, созданных приложением SQL Server Profiler, просмотр их в любое время и отправка для анализа администратору;
- новые окна сообщений об ошибках и информационных сообщений, предоставляющие гораздо больше сведений и позволяющие отправлять в Майкрософт комментарии о сообщениях, копировать сообщения в буфер обмена и отправлять их по электронной почте в службу поддержки;
- встроенный веб-обозреватель для быстрого обращения к библиотеке MSDN или получения интерактивной справки;
- встроенная справка от сообществ в Интернете и т.д.

Большинство действий с базой данной MS SQL Server в среде Среда SQL Server Management Studio может быть осуществлено двумя способами: либо выполнением операторов языка SQL в окнах "Script Execute" (подключение к базе данных не обязательно) и "SQL Editor" (требуется подключение к базе данных), либо с использованием меню и диалоговых окон. В последнем случае операторы SQL, которые требуются для выполнения данного действия, будут сгенерированы и выполнены средой SQL Server Management Studio автоматически.

Задание к лабораторной работе

1. Изучите методический материал

2. Установите SQL Server Management Studio и проверьте соединение с сервером.

Лабораторная работа № 2

Создание структуры БД: операторы DDL (CREATE, ALTER, DROP).

Цель лабораторной работы: Изучить SQL-операторы для работы с таблицами и индексами. Изучить sql-команды для создания, изменения и удаления таблиц.

Команды SQL для создания, удаления и изменения таблицами

Часть языка SQL, которая управляет метаданными, называется Data Definition Language (DDL).

К DDL относятся операторы для определения любых содержащихся в базе данных объектов, в том числе и таблиц.

Операторы, определяющие структуру таблиц в MS SQL Server, соответствуют стандарту SQL, и поэтому без изменений будут работать и во многих других СУБД.

Таблицы создаются, изменяются и удаляются соответственно командами Transact-SQL:

- CREATE TABLE
- ALTER TABLE
- DROP TABLE.

При создании новой базы данных сервер автоматически создает 18 системных таблиц для хранения информации о ее структуре и организации, доступ к которым со стороны пользователя запрещен.

Помимо основных и системных таблиц, которые, как правило, постоянно хранятся в базе данных, можно использовать временные таблицы для временного хранения информации, которые автоматически уничтожаются при закрытии соединения с базой данных.

Инструкция CREATE TABLE (Transact-SQL)

Для создания таблиц используется оператор "CREATE TABLE", который приводит к созданию пустой таблицы без строк. При создании таблиц задается имя таблицы, описание набора столбцов с их именами, типами и размерами, а также ограничения на хранящуюся в таблице информацию.

Имена таблиц в пределах базы данных должны быть уникальны.

Каждый столбец в таблице должен иметь имя, уникальное в пределах таблицы, а также либо тип данных, ограничения целостности, либо выражение для вычисления значения столбца.

Общий синтаксис

```
CREATE TABLE
[ database_name . [ schema_name ] . | schema_name . ] table_name
[ AS FileTable ]
( { <column_definition> | <computed_column_definition>
| <column_set_definition> | [ <table_constraint> ] [ ,...n ] } )
[ ON { partition_scheme_name ( partition_column_name ) | filegroup
| "default" } ]
[ { TEXTIMAGE_ON { filegroup | "default" } ]
[ FILESTREAM_ON { partition_scheme_name | filegroup
| "default" } ]
[ WITH ( <table_option> [ ,...n ] ) ]
[ ; ]
<column_definition> ::=
column_name <data_type>
[ FILESTREAM ]
[ COLLATE collation_name ]
[ NULL | NOT NULL ]
[
[ CONSTRAINT constraint_name ] DEFAULT constant_expression ]
| [ IDENTITY [ ( seed ,increment ) ] [ NOT FOR REPLICATION ]
]
[ ROWGUIDCOL ] [ <column_constraint> [ ...n ] ]
        [ SPARSE ]

<data type> ::=
[ type_schema_name . ] type_name
[ ( precision [ , scale ] | max |
[ { CONTENT | DOCUMENT } ] xml_schema_collection ) ]
<column_constraint> ::=
[ CONSTRAINT constraint_name ]
{ { PRIMARY KEY | UNIQUE }
[ CLUSTERED | NONCLUSTERED ]
[
WITH FILLFACTOR = fillfactor
| WITH ( < index_option > [ , ...n ] )
]
[ ON { partition_scheme_name ( partition_column_name )
| filegroup | "default" } ]
| [ FOREIGN KEY ]
REFERENCES [ schema_name . ] referenced_table_name [ ( ref_column ) ]
[ ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
[ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
[ NOT FOR REPLICATION ]
| CHECK [ NOT FOR REPLICATION ] ( logical_expression )
        }
```

<computed_column_definition> ::=
column_name AS computed_column_expression
[PERSISTED [NOT NULL]]
[
[CONSTRAINT constraint_name]
{ PRIMARY KEY | UNIQUE }
[CLUSTERED | NONCLUSTERED]
[
WITH FILLFACTOR = fillfactor
| WITH (<index_option> [, ...n])
]
| [FOREIGN KEY]
REFERENCES referenced_table_name [(ref_column)]
[ON DELETE { NO ACTION | CASCADE }]
[ON UPDATE { NO ACTION }]
[NOT FOR REPLICATION]
| CHECK [NOT FOR REPLICATION] (logical_expression)
[ON { partition_scheme_name (partition_column_name)
| filegroup | "default" }]
]
<column_set_definition> ::=
column_set_name XML COLUMN_SET FOR ALL_SPARSE_COLUMNS
< table_constraint > ::=
[CONSTRAINT constraint_name]
{
{ PRIMARY KEY | UNIQUE }
[CLUSTERED | NONCLUSTERED]
(column [ASC | DESC] [,...n])
[
WITH FILLFACTOR = fillfactor
|WITH (<index_option> [, ...n])
]
[ON { partition_scheme_name (partition_column_name)
| filegroup | "default" }]
| FOREIGN KEY
(column [,...n])
REFERENCES referenced_table_name [(ref_column [,...n])]
[ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }]
[ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }]
[NOT FOR REPLICATION]
| CHECK [NOT FOR REPLICATION] (logical_expression)
}
<table_option> ::=
{
[DATA_COMPRESSION = { NONE | ROW | PAGE }
[ON PARTITIONS ({ <partition_number_expression> | <range> }
[, ...n])]]
[FILETABLE_DIRECTORY = <directory_name>]
[FILETABLE_COLLATE_FILENAME = { <collation_name> | database_default }]
[FILETABLE_PRIMARY_KEY_CONSTRAINT_NAME = <constraint_name>]
[FILETABLE_STREAMID_UNIQUE_CONSTRAINT_NAME = <constraint_name>]
[FILETABLE_FULLPATH_UNIQUE_CONSTRAINT_NAME = <constraint_name>]
}

<index_option> ::=
{
PAD_INDEX = { ON | OFF }
| FILLFACTOR = fillfactor
| IGNORE_DUP_KEY = { ON | OFF }
| STATISTICS_NORECOMPUTE = { ON | OFF }
| ALLOW_ROW_LOCKS = { ON | OFF}
| ALLOW_PAGE_LOCKS ={ ON | OFF}
| DATA_COMPRESSION = { NONE | ROW | PAGE }
[ON PARTITIONS ({ <partition_number_expression> | <range> }
[, ...n])]
}
<range> ::=
<partition_number_expression> TO <partition_number_expression>

Рассмотрим и расширяем назначение ключевых слов и аргументов команды CREATE TABLE.

Аргументы:

database_name

Имя базы данных, в которой создается таблица. Параметр database_name должен указывать имя существующей базы данных. Если аргумент database_name не указан, по умолчанию список стоп-слов создается в текущей базе данных. Имя входа для текущего соединения должно быть связано с идентификатором пользователя, существующего в базе данных, указанной аргументом database_name, а этот пользователь должен обладать разрешениями CREATE TABLE.

schema_name

Имя схемы, которой принадлежит новая таблица.

table_name

Имя новой таблицы. Имена таблиц должны соответствовать правилам для идентификаторов. Аргумент table_name может состоять не более чем из 128 символов, за исключением имен локальных временных таблиц (имена с одним префиксом номера #), длина которых не должна превышать 116 символов.

AS FileTable

Создает новую таблицу FileTable. Нет необходимости указывать столбцы, так как таблица FileTable имеет фиксированное схему. Дополнительные сведения о таблицах FileTable см. в разделе Таблицы FileTable (SQL Server).

column_name

Имя столбца в таблице. Имена столбцов должны соответствовать правилам именования идентификаторов и быть уникальными в рамках таблицы. Аргумент

column_name может иметь длину не более 128 символов. Аргумент column_name может быть опущен для столбцов, создаваемых с типом данных timestamp. Если аргумент column_name не указан, столбцу типа timestamp по умолчанию присваивается имя timestamp.

computed_column_expression

Выражение, определяющее значение вычисляемого столбца. Вычисляемый столбец представляет собой виртуальный столбец, физически не хранящийся в таблице, если для него не установлен признак PERSISTED. Значение столбца вычисляется на основе выражения, использующего другие столбцы той же таблицы. Например, определение вычисляемого столбца может быть следующим:

cost AS price * qty.

Выражение может быть именем невычисляемого столбца, константой, функцией, переменной или любой их комбинацией, соединенной одним или несколькими операторами. Выражение не может быть вложенным запросом или содержать псевдонимы типов данных.

Вычисляемые столбцы могут использоваться в списках выбора, предложениях WHERE, ORDER BY и в любых других местах, в которых могут использоваться обычные выражения, за исключением следующих случаев.

Вычисляемый столбец нельзя использовать ни в качестве определения ограничения DEFAULT или FOREIGN KEY, ни вместе с определением ограничения NOT NULL. Однако вычисляемый столбец может использоваться в качестве ключевого столбца индекса или части какого-либо ограничения PRIMARY KEY или UNIQUE, если значение этого вычисляемого столбца определяется детерминистическим выражением и тип данных результата разрешен в столбцах индекса.

Вычисляемый столбец не может быть целевым столбцом инструкций INSERT или UPDATE. Примечание

Каждая строка таблицы может содержать различные значения столбцов, задействованных в вычисляемом столбце; таким образом, значение вычисляемого столбца не будет одним и тем же в каждой строке.

PERSISTED - указывает, что компонент Компонент SQL Server Database Engine будет физически хранить вычисляемые значения в таблице и обновлять их при изменении любого столбца, от которого зависит вычисляемый столбец. Указание PERSISTED для вычисляемого столбца позволяет создать индекс по вычисляемому столбцу, который

будет детерминистическим, но неточным.

ON { <partition_scheme> | filegroup | "default" }

Указывает схему секционирования или файловую группу, в которой хранится таблица. Если аргумент <partition_scheme> указан, таблица будет разбита на секции, хранимые в одной или нескольких файловых группах, указанных аргументом <partition_scheme>. Если указан аргумент filegroup, таблица сохраняется в файловой группе с таким именем. Это должна быть существующая файловая группа в базе данных. Если указано значение "default" или параметр ON не определен вообще, таблица сохраняется в файловой группе по умолчанию. Механизм хранения таблицы, указанный в инструкции CREATE TABLE, изменить в дальнейшем невозможно.

TEXTIMAGE_ON { filegroup| "default" }

Указывают, что столбцы типов text, ntext, image, xml, varchar(max), nvarchar(max), varbinary(max), а также определяемых пользователем типов данных CLR (включая geometry и geography) хранятся в указанной файловой группе.

FILESTREAM_ON { partition_scheme_name | filegroup | "default" }

Задает файловую группу для данных FILESTREAM.

Если таблица содержит данные FILESTREAM и является секционированной, необходимо включить предложение FILESTREAM_ON и указать схему секционирования файловых групп файлового потока. В этой схеме секционирования должны использоваться те же функции и столбцы секционирования, что и в схеме секционирования для таблицы; в противном случае возникает ошибка.

[type_schema_name.] type_name

Указывает тип данных столбца и схему, к которой он принадлежит. Тип данных может быть одним из следующих.

Системный тип данных.

Псевдонимы типа на основе системного типа данных SQL Server.

Определяемый пользователем тип данных CLR. Прежде чем определяемые пользователем типы данных CLR можно будет использовать в определении таблицы, их нужно создать с помощью инструкции CREATE TYPE. Для создания столбца с определяемым пользователем типом данных CLR требуется разрешение REFERENCES на этот тип.

precision

Точность указанного типа данных.

scale

Масштаб указанного типа данных.

CONTENT

Указывает, что каждый экземпляр типа данных xml в столбце **column_name** может содержать несколько элементов верхнего уровня. Аргумент **CONTENT** применим только к данным типа xml.

DOCUMENT

Указывает, что каждый экземпляр типа данных xml в столбце **column_name** может содержать только один элемент верхнего уровня. Аргумент DOCUMENT применим только к данным типа xml.

DEFAULT

Указывает значение, присваиваемое столбцу в случае отсутствия явно заданного значения при вставке. Определения **DEFAULT** могут применяться к любым столбцам, кроме имеющих тип timestamp или обладающих свойством IDENTITY. Определения DEFAULT удаляются, когда таблица удаляется из памяти. В качестве значения по умолчанию могут использоваться только константы (например, символьные строки), скалярные функции (системные, определяемые пользователем или функции CLR) или значение NULL.

constant_expression

Константа, значение NULL или системная функция, используемая в качестве значения столбца по умолчанию.

IDENTITY

Указывает, что новый столбец является столбцом идентификаторов. При добавлении в таблицу новой строки компонент Компонент Database Engine формирует для этого столбца уникальное последовательное значение. Столбцы идентификаторов обычно используются с ограничением **PRIMARY KEY** для поддержания уникальности идентификаторов строк в таблице.

Свойство **IDENTITY** присвоено столбцам типа **tinyint, smallint, int, bigint, decimal(p,0)** или **numeric(p,0).** Для каждой таблицы можно создать только один столбец идентификаторов. Ограниченные значения по умолчанию и ограничения DEFAULT не могут использоваться в столбце идентификаторов.

Необходимо указать как начальное значение, так и приращение, или же не указывать ничего. Если ничего не указано, применяется значение по умолчанию (**1,1**).

seed

Значение, используемое для самой первой строки, загружаемой в таблицу.

increment

Значение приращения, добавляемое к значению идентификатора предыдущей загруженной строки.

NOT FOR REPLICATION

В инструкции **CREATE TABLE** предложение **NOT FOR REPLICATION** может указываться для свойства **IDENTITY**, а также ограничений **FOREIGN KEY** и **CHECK**. Если это предложение указано для свойства **IDENTITY**, значения в столбцах идентификаторов не приращиваются, если вставку выполняют агенты репликации. Если ограничение сопровождается этим предложением, оно не выполняется, когда агенты репликации выполняют операции вставки, обновления или удаления.

ROWGUIDCOL

Указывает, что новый столбец является столбцом идентификаторов GUID строки. Только один столбец типа uniqueidentifier в таблице может быть назначен в качестве столбца ROWGUIDCOL. Применение свойства ROWGUIDCOL позволяет ссылаться на столбец с помощью ключевого слова $ROWGUID. Свойство ROWGUIDCOL может быть присвоено только столбцу типа uniqueidentifier. Ключевым словом ROWGUIDCOL нельзя обозначать столбцы определяемых пользователем типов данных.

SPARSE

Указывает, что столбец является разреженным столбцом. Хранилище разреженных столбцов оптимизируется для значений NULL. Для разреженных столбцов нельзя указать параметр NOT NULL. Дополнительные ограничения и сведения о разреженных столбцах см. в разделе Использование разреженных столбцов.

FILESTREAM

Допустимо только для столбцов типа varbinary(max). Указывает хранилище FILESTREAM для данных BLOB типа varbinary(max).

COLLATE collation_name

Задает параметры сортировки для столбца. Могут использоваться параметры сортировки Windows или параметры сортировки SQL. Параметр **collation_name** применим только к столбцам типов данных **char, varchar, text, nchar, nvarchar** и **ntext**. Если этот аргумент не указан, столбцу назначаются либо параметры сортировки определяемого пользователем типа, если столбец принадлежит к определяемому пользователем типу данных, либо установленные по умолчанию параметры сортировки для базы данных.

CONSTRAINT

Необязательное ключевое слово, указывающее на начало определения ограничения **PRIMARY KEY, NOT NULL, UNIQUE, FOREIGN KEY** или **CHECK**.

constraint_name

Имя ограничения. Имена ограничений должны быть уникальными в пределах схемы, к которой принадлежит таблица.

NULL | NOT NULL

Определяет, допустимы ли для столбца значения NULL.

PRIMARY KEY

Ограничение, которое обеспечивает целостность сущностей для указанного столбца или столбцов с помощью уникального индекса. Можно создать только одно ограничение **PRIMARY KEY** для таблицы.

UNIQUE

Ограничение, которое обеспечивает целостность сущностей для указанного столбца или столбцов с помощью уникального индекса. В таблице может быть несколько ограничений **UNIQUE.**

FOREIGN KEY REFERENCES

Ограничение, которое обеспечивает ссылочную целостность данных в этом столбце или столбцах. Ограничения **FOREIGN KEY** требуют, чтобы каждое значение в столбце существовало в соответствующем связанном столбце или столбцах в связанной таблице.

Ограничения **FOREIGN KEY** могут ссылаться только на столбцы, являющиеся ограничениями **PRIMARY KEY** или **UNIQUE** в связанной таблице или на столбцы, на которые имеются ссылки в индексе UNIQUE INDEX связанной таблицы. Внешние ключи в вычисляемых столбцах должны быть также помечены как PERSISTED.

[schema_name.] referenced_table_name]

Имя таблицы, на которую ссылается ограничение **FOREIGN KEY**, и схема, к которой она принадлежит.

(ref_column [,... n])

Столбец или список столбцов из таблицы, на которую ссылается ограничение **FOREIGN KEY**.

ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }

Определяет операцию, которая производится над строками создаваемой таблицы, если эти строки имеют ссылочную связь, а строка, на которую имеются ссылки, удаляется из родительской таблицы. Параметр по умолчанию — NO ACTION.

NO ACTION

Компонент Компонент Database Engine формирует ошибку, и выполняется откат операции удаления строки из родительской таблицы.

CASCADE

Если из родительской таблицы удаляется строка, соответствующие ей строки удаляются и из ссылающейся таблицы.

SET NULL

Все значения, составляющие внешний ключ, при удалении соответствующей строки родительской таблицы устанавливаются в NULL. Для выполнения этого ограничения внешние ключевые столбцы должны допускать значения NULL.

SET DEFAULT

Все значения, составляющие внешний ключ, при удалении соответствующей строки родительской таблицы устанавливаются в значение по умолчанию. Для выполнения этого ограничения все внешние ключевые столбцы должны иметь определения по умолчанию. Если столбец допускает значения NULL и значение по умолчанию явно не определено, значением столбца по умолчанию становится NULL.

ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }

Указывает, какое действие совершается над строками в изменяемой таблице, когда эти строки имеют ссылочную связь и строка родительской таблицы, на которую указывает ссылка, обновляется. Параметр по умолчанию — NO ACTION.

NO ACTION

Компонент Компонент Database Engine возвращает ошибку, а обновление строки родительской таблицы откатывается.

CASCADE

Соответствующие строки обновляются в ссылающейся таблице, если эта строка обновляется в родительской таблице.

SET NULL

Всем значениям, составляющим внешний ключ, присваивается значение NULL, когда обновляется соответствующая строка в родительской таблице. Для выполнения этого ограничения внешние ключевые столбцы должны допускать значения NULL.

SET DEFAULT

Всем значениям, составляющим внешний ключ, присваивается их значение по умолчанию, когда обновляется соответствующая строка в родительской таблице.

CHECK

Ограничение, обеспечивающее целостность домена путем ограничения возможных значений, которые могут быть введены в столбец или столбцы. Ограничения **CHECK** в вычисляемых столбцах должны быть также помечены как **PERSISTED**.

logical_expression

Логическое выражение, возвращающее значения TRUE или FALSE..

column

Столбец или список столбцов (в скобках), используемый в ограничениях таблицы для указания столбцов, используемых в определении ограничения.

[ASC | DESC]

Указывает порядок сортировки столбца или столбцов, участвующих в ограничениях таблицы. Значение по умолчанию — ASC.

partition_scheme_name

Имя схемы секционирования, определяющей файловые группы, которым сопоставляются секции секционированной таблицы. Эта схема секционирования должна существовать в базе данных.

[partition_column_name.]

Указывает столбец, по которому будет секционирована таблица. Столбец должен соответствовать по типу данных, длине и точности столбцу, указанному в функции секционирования, используемой аргументом partition_scheme_name. Вычисляемый столбец, участвующий в функции секционирования, должен быть явно обозначен ключевым словом PERSISTED. Важно!

WITH FILLFACTOR =fillfactor

Указывает, насколько плотно компонент Компонент Database Engine должен заполнять каждую страницу индекса, используемую для хранения данных индекса.

column_set_name XML COLUMN_SET FOR ALL_SPARSE_COLUMNS

Имя набора столбцов. Набор столбцов представляет собой нетипизированное XML-представление, в котором все разреженные столбцы таблицы объединены в структурированные выходные данные.

< table_option> ::=

Указывает один или более параметров таблицы.

DATA_COMPRESSION

Задает режим сжатия данных для указанной таблицы, номера секции или диапазона секций. Ниже приведены доступные параметры.

ON PARTITIONS ({ <partition_number_expression> | <range> } [,...n])

Указывает секции, к которым применяется параметр DATA_COMPRESSION. Если таблица не секционирована, аргумент ON PARTITIONS приведет к формированию ошибки.

Упрощенный синтаксис оператора создания таблицы:

<определение_таблицы> ::=
CREATE TABLE [имя_базы_данных.[владелец]. | владелец.]
имя_таблицы
(<элемент_таблицы>[,...n])

где

<элемент_таблицы> ::=
{<определение_столбца>}
| <имя_столбца> AS <выражение>
|>ограничение_таблицы<

Обычно владельцем таблицы (**dbo**) является тот, кто ее создал.

<Выражение> задает значение для вычисляемого столбца.

<определение_столбца> ::=
{ имя_столбца <тип_данных>}
[[DEFAULT <выражение>]
| [IDENTITY (начало, шаг) [NOT FOR REPLICATION]]]
[ROWGUIDCOL][<ограничение_столбца>][...n]]

В определении столбца обратим внимание на параметр IDENTITY, который указывает, что соответствующий столбец будет столбцом-счетчиком. Для таблицы может

быть определен только один столбец с таким свойством. Можно дополнительно указать начальное значение и шаг приращения. Если эти значения не указываются, то по умолчанию они оба равны 1. Если с ключевым словом IDENTITY указано NOT FOR REPLICATION, то сервер не будет выполнять автоматического генерирования значений для этого столбца, а разрешит вставку в столбец произвольных значений.

В качестве ограничений используются ограничения столбца и ограничения таблицы. Различие между ними в том, что ограничение столбца применяется только к определенному полю, а ограничение таблицы - к группам из одного или более полей. Различные типы ограничений рассмотрим позже.

Пример создания таблицы без ограничений.

Для выполнения данных запросов предварительно откройте среду Micrpsoft SQL Server Managmant Studio, выполните соединение с сервером и откройте базу данных Educator. Нажмите на панели инструментов команду Создать запрос.

Пример. Создание родительской таблицы Товар без ограничений.

```
use Educator
CREATE TABLE Товар
(КодТовара INT IDENTITY(1,1),
Название VARCHAR(50) ,
Цена MONEY ,
Тип VARCHAR(50) ,
Сорт VARCHAR(50) ,
Город VARCHAR(50) ,
    Остаток INT );
```

Выполните sql-код. Обновите базу данных и просмотрите созданную таблицу. Сохраните sql-запрос под именем Пример1.sql в папке ФИО_студента/Лаб2.

Автоматическая генерация значения столбца КодТовара достигается за счет использования свойства IDENTITY, по умолчанию начальное значение, генерируемое с помощью IDENTITY равно 1, так же как и его приращение. Таким образом, следующее значение будет равно 2. Значения в IDENTITY-столбцах обязательно последовательные, то есть если приращение положительное, то следующее значение всегда больше предыдущего, если приращение отрицательное, то – всегда меньше. Приращение и начальное значение могут быть заданы, однако этот механизм чрезвычайно редко используется в реальных проектах.

Изменение таблиц

Для внесения изменений в уже созданные таблицы стандартом SQL предусмотрен оператор **ALTER TABLE**, предназначенный для выполнения следующих действий:

- добавление в таблицу нового столбца;
- удаление столбца из таблицы;
- добавление в определение таблицы нового ограничения;
- удаление из определения таблицы существующего ограничения;
- задание для столбца значения по умолчанию;

- отмена для столбца значения по умолчанию.

Оператор изменения таблицы имеет следующий обобщенный формат:
<изменение_таблицы> ::=
ALTER TABLE имя_таблицы
[ADD [COLUMN]имя_столбца тип_данных
[NOT NULL][UNIQUE]
[DEFAULT <значение>][CHECK (<условие_выбора>)]]
[DROP [COLUMN] имя_столбца [RESTRICT | CASCADE]]
[ADD [CONSTRAINT [имя_ограничения]]
[{PRIMARY KEY (имя_столбца [,...n])
|[UNIQUE (имя_столбца [,...n])}
|[FOREIGN KEY (имя_столбца_внешнего_ключа [,...n])
REFERENCES имя_род_таблицы
[(имя_столбца_род_таблицы [,...n])],

[MATCH {PARTIAL | FULL}
[ON UPDATE {CASCADE| SET NULL |
SET DEFAULT | NO ACTION}]
[ON DELETE {CASCADE| SET NULL |
SET DEFAULT | NO ACTION}]
|[CHECK(<условие_выбора>)][,...n]}]
[DROP CONSTRAINT имя_ограничения
[RESTRICT | CASCADE]]
[ALTER [COLUMN] SET DEFAULT <значение>]
[ALTER [COLUMN] DROP DEFAULT]

Здесь параметры имеют то же самое назначение, что и в определении оператора CREATE TABLE.

Оператор ALTER TABLE реализован не во всех диалектах языка SQL. В некоторых диалектах он поддерживается, однако не позволяет удалять из таблицы уже существующие столбцы.

В дополнение к уже названным параметрам определим параметр {ENABLE | DISABLE } TRIGGER ALL_, предписывающий задействовать или отключить конкретный триггер или все триггера, связанные с таблицей.

Пример. Пример создания вычисляемого поля.
*ALTER TABLE Товар ADD Налог AS Цена*0.05*
Пример Пример удаления поля
ALTER TABLE Товар DROP COLUMN Остаток

Удаление таблиц

Удаление таблицы выполняется командой:

DROP TABLE имя_таблицы

Удалить можно любую таблицу, даже системную. К этому вопросу нужно подходить очень осторожно. Однако удалению не подлежат таблицы, если существуют объекты, ссылающиеся на них. К таким объектам относятся таблицы, связанные с удаляемой таблицей посредством внешнего ключа. Поэтому, прежде чем удалять родительскую таблицу, необходимо удалить либо ограничение внешнего ключа, либо дочерние таблицы. Если с таблицей связано хотя бы одно представление, то таблицу также удалить не удастся. Кроме того, связь с таблицей может быть установлена со стороны функций и процедур. Следовательно, перед удалением таблицы необходимо

удалить все объекты базы данных, которые на нее ссылаются, либо изменить их таким образом, чтобы ссылок на удаляемую таблицу не было.

Задание к лабораторной работе

1. Изучите методический материал

2. Самостоятельно создайте базу данных с таблицами по вариантам по следующей таблице. Заполните каждую таблицу данными (3-5 записей в каждой таблице).

Вариант №1	**БД – успеваемость студентов ВУЗА**. БД состоит из следующих таблиц: факультеты, кафедры, учебные группы, студенты, ведомости успеваемости. **Таблица факультеты** имеет следующие атрибуты: название факультета, ФИО декана, номер комнаты, номер корпуса, телефон. **Таблица кафедра** имеет следующие атрибуты: название кафедры, факультет, ФИО заведующего, номер комнаты, номер корпуса, телефон, кол-во преподавателей. **Таблица учебные группы** имеет следующие атрибуты: название группы, год поступления, курс обучения, кол-во студентов в группе, кафедра. **Таблица студенты** имеет следующие атрибуты: студента, фамилия, имя, отчество, группа, год рождения, пол, адрес, город, телефон. **Таблица ведомости успеваемости** имеет следующие атрибуты: группа, студент, предмет, оценка.
Вариант №2	**БД – информационная система супермаркета**. БД состоит из следующих таблиц: отделы, сотрудники, товары, продажа товаров, должности. **Таблица отделы** имеет следующие атрибуты: название отдела, кол-во прилавков, кол-во продавцов, номер зала. **Таблица сотрудники** имеет следующие атрибуты: фамилия, имя, отчество, отдел, год рождения, год поступления на работу, стаж, должность, пол, адрес, город, телефон. **Таблица должности** имеет следующие атрибуты: название должности, сумма ставки. **Таблица товары** имеет следующие атрибуты: название товара, отдел, страна производитель, условия хранения, сроки хранения . **Таблица продажа товаров** имеет следующие атрибуты: сотрудника являющегося продавцом, товара дата, время, кол-во, цена, сумма.
Вариант №3	**БД – информационная система военного округа**. БД состоит из следующих таблиц: места дислокации, вид войск, части, роты, личный состав. **Таблица вид войск** имеет следующие атрибуты: название. **Таблица места дислокации** имеет следующие атрибуты: страна, город, адрес, занимаемая площадь. **Таблица части** имеет следующие атрибуты: номер части, место дислокации, вид войск, кол-во рот. **Таблица роты** имеет следующие атрибуты: название роты, кол-во служащих, часть. **Таблица личный состав** имеет следующие атрибуты: фамилия, рота, должность, год рождения, год поступления на службу, выслуга лет, награды, участие в военных мероприятиях.
Вариант №4	**БД – информационная система библиотеки**. БД состоит из следующих таблиц: библиотеки, фонд библиотеки, тип литературы, сотрудники, пополнение фонда. **Таблица библиотеки** имеет следующие атрибуты: название, адрес, город. **Таблица фонд библиотеки** имеет следующие атрибуты: название фонда, библиотека, кол-во книг, кол-во журналов, кол-во газет, кол-во сборников, кол-во диссертаций, кол-во рефератов. **Таблица тип литературы** имеет следующие атрибуты: название типа. **Таблица сотрудники** имеет следующие атрибуты: фамилия сотрудника, библиотека, должность, год рождения, год поступления на работу, образование, зарплата. **Таблица пополнение фонда** имеет следующие атрибуты: фонд, сотрудник, дата, название источника литературы, тип литературы, издательство, дата издания, кол-во экземпляров.

Вариант №5	**БД – информационная система туристического агентства**. БД состоит из следующих таблиц: пансионаты, туры, клиенты, путевки, вид жилья. **Таблица пансионаты** имеет следующие атрибуты: название пансионата, адрес, город, страна, телефон, описание территории, кол-во комнат, наличие бассейна, наличие медицинских услуг, наличие спа-салона, уровень пансионата, расстояние до моря. **Таблица вид жилья** имеет следующие атрибуты: название (дом, бунгало, квартира, 1-я комната, 2-я комната и т.д.), категория жилья (люкс, полулюкс, и т.д.), пансионат, описание условий проживания, цена за номер в сутки. **Таблица туры** имеет следующие атрибуты: название тура (Европа, средняя Азия, тибет и т.д.), вид транспорта, категория жилья на ночь (гостиница, отель, палатка и т.д.), вид питания (одноразовое, двухразовое, трехразовое, завтраки), цена тура в сутки. **Таблица клиенты** имеет следующие атрибуты: фамилия, имя, отчество, паспортные данные, дата рождения, адрес, город, телефон. **Таблица путевки** имеет следующие атрибуты: клиент, тур, дата заезда, дата отъезда, наличие детей, наличие мед. страховки, кол-во человек, цена, сумма.
Вариант №6	**БД – информационная система автопредприятия города.** БД состоит из следующих таблиц: автотранспорт, водители, маршруты, обслуживающий персонал, гаражное хозяйство. **Таблица автотранспорт** имеет следующие атрибуты: название транспорта (автобусы, такси, маршрутные такси, прочий легковой транспорт, грузовой транспорт и т.д.), кол-во наработки, пробег, кол-во ремонтов, характеристика. **Таблица маршруты** имеет следующие атрибуты: название маршрута, транспорт, водитель, график работы. **Таблица водители** имеет следующие атрибуты: фамилия, имя, отчество, год рождения, год поступления на работу, стаж, должность, пол, адрес, город, телефон. **Таблица обслуживающий персонал** имеет следующие атрибуты: должность (техники, сварщики, слесари, сборщики и др.), фамилия, имя, отчество, год рождения, год поступления на работу, стаж, пол, адрес, город, телефон. **Таблица гаражное хозяйство** имеет следующие атрибуты: название гаража, транспорт на ремонте, вид ремонта, дата поступления, дата выдачи после ремонта, результат ремонта, персонал, производящего ремонт.
Вариант №7	**БД – информационная система поликлиники**. БД состоит из следующих таблиц: врачи, пациенты, история болезней, отделения, обслуживающий персонал. **Таблица отделения** имеет следующие атрибуты: название отделения (хирургия, терапия, нервология и т.д.), этаж, номера комнат, ФИО заведующего. **Таблица врачи** имеет следующие атрибуты: фамилия, имя, отчество, должность, стаж работы, научное звание, адрес, номер отделения, в котором он работает. **Таблица пациенты** имеет следующие атрибуты: фамилия, имя, отчество, адрес, город, возраст, пол. **Таблица диагнозы** имеет следующие атрибуты: название диагноза, признаки болезни, период лечения, назначения. **Таблица история болезни** имеет следующие атрибуты: пациент, врач, диагноз, лечение, дата заболевания, дата вылечивания, вид лечения (амбулаторное, стационарное).
Вариант №8	**БД – информационная система больницы**. БД состоит из следующих таблиц: врачи, пациенты, история болезней, операции, лист лечения. **Таблица врачи** имеет следующие атрибуты:, фамилия, имя, отчество, должность, стаж работы, научное звание, адрес. **Таблица пациенты** имеет следующие атрибуты: фамилия, имя, отчество, адрес, город, возраст, пол. **Таблица история болезни** имеет следующие атрибуты:, пациент, врач, диагноз, дата заболевания, дата вылечивания, вид лечения (амбулаторное, стационарное), код операции. **Таблица лист лечения** имеет следующие атрибуты: дата лечения, история болезни, лекарства, температура, давление, состояние больного (тяжелое, среднее, и т.д.). **Таблица операции** имеет следующие атрибуты: описание операции (удаление аппендицита, пластическая операция и т.д.), врач, дата операции, пациент, результат операции.
Вариант	**БД – информационная система библиотек города**. БД состоит из следующих таблиц:

№9	библиотеки, читальные залы, литература, читатели, выдача лит-ры. **Таблица библиотеки** имеет следующие атрибуты: название, адрес, город. **Таблица читальные залы** имеет следующие атрибуты: название читального зала, библиотека, кол-во единиц лит-ры, кол-во посадочных мест, время работы, этаж, кол-во сотрудников. **Таблица читатели** имеет следующие атрибуты: фамилия, имя, отчество, категория читателя, место работы или обучения, возраст, дата регистрации в библиотеке. **Таблица литература** имеет следующие атрибуты: название, категория литературы, авторы, издательство, год издательства, кол-во страниц, читальный зал. **Таблица выдача литературы** имеет следующие атрибуты: читатель, литература, дата выдачи, срок выдачи, вид выдачи, наличие залога.
Вариант №10	**БД – информационная система автосалона**. БД состоит из следующих таблиц: автомобили, марка автомобиля, сотрудники, продажа автомобилей, покупатели. **Таблица марка автомобиля** имеет следующие атрибуты: название марки, страна производитель, завод производитель, адрес. **Таблица автомобиля** имеет следующие атрибуты: название автомобиля, марка, год производства, цвет, категория, цена. **Таблица покупатели** имеет следующие атрибуты: фамилия, имя, отчество, паспортные данные, адрес, город, возраст, пол. **Таблица сотрудника** имеет следующие атрибуты: фамилия, имя, отчество, стаж, зарплата. **Таблица продажа автомобилей** имеет следующие атрибуты: дата, сотрудник, автомобиль, покупатель.
Вариант №11	**БД – успеваемость студентов кафедры**. БД состоит из следующих таблиц: кафедры, дисциплины, преподаватели, студенты, ведомости успеваемости. **Таблица кафедра** имеет следующие атрибуты: название кафедры, факультет, ФИО заведующего, номер комнаты, номер корпуса, телефон, кол-во преподавателей. **Таблица преподаватели** имеет следующие атрибуты: фамилия, имя, отчество, кафедра, год рождения, год поступления на работу, стаж, должность, пол, адрес, город, телефон. **Таблица студенты** имеет следующие атрибуты: фамилия, имя, отчество, кафедра, год рождения, пол, адрес, город, телефон. **Таблица дисциплины** имеет следующие атрибуты: название дисциплины, кафедра, читаемой эту дисциплину, кол-во часов, вид итогового контроля. **Таблица ведомости успеваемости** имеет следующие атрибуты: преподаватель, дисциплина, студент, оценка.
Вариант №12	**БД – торговая организация**. БД состоит из следующих таблиц: торговая организация, торговая точка, продавцы, поставщики, заказы поставщикам. **Таблица торговая организация** имеет следующие атрибуты: название торговой организации, адрес, ФИО директора, налоговый номер. **Таблица торговая точка** имеет следующие атрибуты: название торговой точки, тип (универмаги, магазины, киоски, лотки и т.д.), торговая организация, адрес, ФИО заведующего. **Таблица продавцы** имеет следующие атрибуты: фамилия, имя, отчество, торговая точка, должность, год рождения, пол, адрес проживания, город. **Таблица поставщики** имеет следующие атрибуты: название поставщика, тип деятельности, страна, город, адрес. **Таблица заказы поставщикам** имеет следующие атрибуты: дата заказа, торговая точка, поставщик, название товара, кол-во, цена.
Вариант №13	**БД – проектная организация**. БД состоит из следующих таблиц: отделы, сотрудники, организации, договора, проектные работы. **Таблица отделы** имеет следующие атрибуты: название отдела, этаж, телефон, начальник отдела. **Таблица сотрудники** имеет следующие атрибуты: ФИО, должность (конструкторы, инженеры, техники, лаборанты, прочий обслуживающий персонал), номер отдела, в котором работает, пол, адрес, дата рождения.

	Таблица организации имеет следующие атрибуты: название организации, тип деятельности, страна, город, адрес, ФИО директора. **Таблица договора** имеет следующие атрибуты: номер договора, дата заключения договора, организация, стоимость договора. **Таблица проектные работы** имеет следующие атрибуты: дата начала проектной работы, дата завершения проектной работы, номер договора, отдел, осуществляющий разработку.
Вариант №14	**БД – информационная система военно-морского флота**. БД состоит из следующих таблиц: базы, части, личный состав, корабли, учения. **Базы военно-морского флота** имеет следующие атрибуты: название базы, георгафическое расположение, кол-во частей. **Таблица части** имеет следующие атрибуты: номер части, база флота, место базирования, вид войск (морская авиация, морская пехота и т.д.). **Таблица личный состав** имеет следующие атрибуты: фамилия, часть, должность, год рождения, год поступления на службу, выслуга лет, награды, **Таблица корабли** имеет следующие атрибуты: идентификационный номер корабля, название корабля, тип корабля, дата создания, наработка, кол-во посадочных мест, устройство двигателя (парусное, гребное, пароход, теплоход, турбоход, и т.д.), тип привода (самоходное, несамоходное), размещение корпуса (подводная лодка, ныряющее, полупогружное, и т.д.) **Таблица учения**: часть, корабль, дата учения, место проведения, оценка.
Вариант №15	**БД – туристическая фирма**. БД состоит из следующих таблиц: туристы, туристическая группа, состав групп, гостиницы, ведомости продажей. **Таблица туристы** имеет следующие атрибуты: ФИО, паспортные данные, пол, возраст, дети. **Таблица туры** имеет следующие атрибуты: название, страна, города, тип передвижения, тип питания, цена тура, тип проживания. **Таблица туристическая группа** имеет следующие атрибуты: название, дата отправления, дата прибытия, тур, кол-во туристов. **Таблица состав групп** имеет следующие атрибуты: дата продажи, турист, группа, цена билета. **Таблица гостиницы** имеет следующие атрибуты: название гостиницы, страна, город, адрес, кол-во мест, тип гостиницы. **Таблица ведомость продажей** имеет следующие атрибуты: дата, туристическая группа, гостиница, общая стоимость.
Вариант №16	**БД – цирк**. БД состоит из следующих таблиц: работники цирка, представления, расписание гастролей, труппа цирка, программа цирка. **Таблица работники цирка** имеет следующие атрибуты: фамилия, имя, отчество, год рождения, год поступления на работу, стаж, должность (акробат, клоун, гимнаст, музыкант, постановщик, служащий и т.д.), пол, адрес, город, телефон. **Таблица представления** имеет следующие атрибуты: название, режиссер-постановщик, художник-постановщик, дирижер-постановщик, автор, жанр, тип. **Таблица расписание гастролей** имеет следующие атрибуты: представление, дата начала, дата окончания, места проведения гастроли. **Таблица труппа представления цирка** имеет следующие атрибуты: представление, актер цирка, название роли. **Таблица программа цирка** имеет следующие атрибуты: представление, дата премьеры, период проведения, дни и время, цена билета.

Вариант №17	**БД – аптека**. БД состоит из следующих таблиц: лекарства, покупатели, продавцы, рецепты, продажа лекарств. **Таблица лекарства** имеет следующие атрибуты: название, тип (готовое, изготовляемое), вид (таблетки, мази, настойки), цена. **Таблица покупатели** имеет следующие атрибуты: фамилия, имя, отчество, адрес, город, телефон. **Таблица продавцы** имеет следующие атрибуты: фамилия, имя, отчество, дата поступления, дата рождения, образование. **Таблица рецепты** имеет следующие атрибуты: номер рецепта, дата выдачи, ФИО больного (покупатель), ФИО врача, диагноз пациента. **Таблица продажа лекарств** имеет следующие атрибуты: дата, лекарство, кол-во, рецепт, продавец.
Вариант №18	**БД – городская телефонная сеть**. БД состоит из следующих таблиц: АТС, абонент, ведомость звонков, прайс АТС, ведомость абонентской платы. **Таблица АТС** имеет следующие атрибуты: название АТС, вид (городские, ведомственные и учрежденческие), адрес, город, кол-во абонентов. **Таблица абоненты** имеет следующие атрибуты: фамилия, имя, отчество, вид телефона (основной, параллельный или спаренный), номер телефона, межгород (открыт/закрыт), льгота (да/нет), адрес: индекс, район, улица, дом, квартира, АТС. **Таблица ведомость звонков** имеет следующие атрибуты: абонент, дата звонка, время начала, время окончания, межгород (да/нет). **Таблица прайс АТС** имеет следующие атрибуты: АТС, цена на городские, цена на межгород. **Таблица ведомость абонентской платы** имеет следующие атрибуты: абонент, месяц, год, кол-во минут на городские, кол-во минут на межгород, стоимость, сумма льготы, общая стоимость.
Вариант №19	**БД – аэропорт**. БД состоит из следующих таблиц: работники аэропорта, расписание вылетов, самолеты, бригады самолетов, ведомость продаж билетов. **Таблица работники аэропорта** имеет следующие атрибуты: фамилия, имя, отчество, год рождения, год поступления на работу, стаж, должность (пилотов, диспетчеров, техников, кассиров, работников службы безопасности, справочной службы и других,), пол, адрес, город, телефон. **Таблица расписание вылетов** имеет следующие атрибуты: самолет, дата вылета, время вылета, место выбытия, место прибытия, маршрут (начальный и конечный пункты назначения, пункт пересадки), стоимость билета. **Таблица самолеты** имеет следующие атрибуты: номер, год выпуска, кол-во посадочных место, грузоподъемность. **Таблица бригады самолетов** имеет следующие атрибуты: номер бригады, самолет, работник аэропорта (пилоты, техники и обслуживающий персонал)ю **Таблица ведомость продажи билетов** имеет следующие атрибуты: дата и время продажи, ФИО пассажира, паспортные данные, номер рейса, кол-во билетов, наличие льгот (пенсионеры, дети-сироты и т.д.), багаж (да/нет), стоимость.
Вариант №20	**БД – театр**. БД состоит из следующих таблиц: работники театра, спектакли, расписание гастролец, труппа спектакля, репертуар театра. **Таблица работники театра** имеет следующие атрибуты: фамилия, имя, отчество, год рождения, год поступления на работу, стаж, должность (актеров, музыкантов, постановщиков и служащих), пол, адрес, город, телефон. **Таблица спектакли** имеет следующие атрибуты: название, режисер-постановщик, художник-постановщик, дирижер-постановщик, автор, жанр (музыкальная комедия, трагедия, оперетта и пр), тип (детские, молодежные и пр.). **Таблица расписание гастролей** имеет следующие атрибуты: название, дата начала, дата окончания, места проведения гастроли, спектакль.

	Таблица труппа спектакля имеет следующие атрибуты: спектакль, актер, название роли. **Таблица репертуар театра** имеет следующие атрибуты: спектакль, дата премьеры, период проведения, дни и время, цена билета.
Вариант №21	**БД – железнодорожный вокзал**. БД состоит из следующих таблиц: работники ж.д.вокзала, расписание движения поездов, поезда, бригады поездов, ведомость продаж билетов. **Таблица работники ж.д.вокзала** имеет следующие атрибуты: фамилия, имя, отчество, год рождения, год поступления на работу, стаж, должность (машинист, диспетчеров, проводник, ремонтников подвижного состава, путей, кассиров, работников службы подготовки составов, справочной службы и других,), пол, адрес, город, телефон. **Таблица расписание движения поездов** имеет следующие атрибуты: поезд, дата отправления, время отправления, место отправления, дата прибытия, время прибытия, место прибытия, маршрут ((начальный и конечный пункты назначения, основные узловые станции), стоимость билета. **Таблица поезда** имеет следующие атрибуты: номер, год выпуска, кол-во вагонов, тип поезда (общий, скоростной, высокоскоростной). **Таблица бригады поездов** имеет следующие атрибуты: номер бригады, поезд, работник ж.д.вокзала (машинисты, техники, проводники и обслуживающий персонал). **Таблица ведомость продажи билетов** имеет следующие атрибуты: дата и время продажи, ФИО пассажира, паспортные данные, номер рейса, кол-во билетов, наличие льгот (пенсионеры, дети-сироты и т.д.), стоимость.
Вариант №22	**БД – информационная система ВУЗА**. БД состоит из следующих таблиц: факультеты, кафедры, преподаватели, дисциплины, учебная нагрузка. **Таблица факультеты** имеет следующие атрибуты: название факультета, ФИО декана, номер комнаты, номер корпуса, телефон. **Таблица кафедра** имеет следующие атрибуты: название кафедры, ФИО заведующего, номер комнаты, номер корпуса, телефон, кол-во преподавателей. **Таблица дисциплины** имеет следующие атрибуты: название дисциплины, кол-во часов, цикл дисциплин. **Таблица преподаватели** имеет следующие атрибуты: фамилия, имя, отчество, кафедра, год рождения, год поступления на работу, стаж, должность, пол, город. **Таблица учебная нагрузка** имеет следующие атрибуты: преподаватель, дисциплина, учебный год, семестр, группы, кол-во студентов, вид итогового контроля.
Вариант №23	**БД – информационная система военного округа**. БД состоит из следующих таблиц: места дислокации, вид войск, части, роты, личный состав. **Таблица вид войск** имеет следующие атрибуты: название вида войск. **Таблица места дислокации** имеет следующие атрибуты: страна, город, адрес, занимаемая площадь, кол-во сооружений. **Таблица части** имеет следующие атрибуты: номер части, место дислокации, вид войск, кол-во рот, кол-во техники, кол-во вооружений. **Таблица техника** имеет следующие атрибуты: название техники, часть, характеристики. **Таблица вооружения** имеет следующие атрибуты: название вооружения, часть, характеристики.
Вариант №24	**БД – информационная система супермаркета**. БД состоит из следующих таблиц: отделы, клиенты, товары, продажа товаров, поставщики. **Таблица отделы** имеет следующие атрибуты: название отдела, кол-во прилавков, кол-во продавцов, номер зала. **Таблица клиенты** имеет следующие атрибуты: название клиента, адрес, вид оплаты. **Таблица поставщики** имеет следующие атрибуты: название поставщика, адрес, страна, вид транспорта, вид оплаты. **Таблица товары** имеет следующие атрибуты: название товара, отдел, поставщик, условия хранения, сроки хранения . **Таблица продажа товаров** имеет следующие атрибуты: клиент, товар, дата, время, кол-во, цена, сумма.

Вариант №25	**БД – информационная система больницы**. БД состоит из следующих таблиц: врачи, пациенты, история болезней, отделения, лист лечения. **Таблица отделения** имеет следующие атрибуты: название отделения (хирургия, терапия, нервология и т.д.), этаж, номера комнат, ФИО заведующего. **Таблица врачи** имеет следующие атрибуты: фамилия, имя, отчество, должность, стаж работы, научное звание, адрес, отделение. **Таблица пациенты** имеет следующие атрибуты: фамилия, имя, отчество, адрес, город, возраст, пол. **Таблица история болезни** имеет следующие атрибуты: пациент, врач, диагноз, дата заболевания, дата вылечивания, вид лечения (амбулаторное, стационарное). **Таблица лист лечения** имеет следующие атрибуты: дата лечения, история болезни, лекарства, температура, давление, состояние больного (тяжелое, среднее, и т.д.).
Вариант №26	**БД –Мебельная фабрика**. БД состоит из следующих таблиц: **Таблица Склад** имеет следующие атрибуты: название, адрес, город **Таблица Мебель** имеет следующие атрибуты: наименование, модель, производитель, цена, склад, на котором он находится. **Таблица Покупатели** имеет следующие атрибуты: фамилия, имя, отчество, адрес, город, возраст, пол. **Таблица продавцы** имеет следующие атрибуты: фамилия, имя, отчество, адрес проживания, стаж работы, паспортные данные. **Таблица Заказы** имеет следующие атрибуты: дата заказа, время заказа, продавец, покупатель, мебель, кол-во, стоимость.
Вариант №27	**БД –Детский образовательный центр**. БД состоит из следующих таблиц: **Таблица Преподаватели** имеет следующие атрибуты: фамилия, имя, отчество, образование, категория, дата рождения, возраст, адрес, дата поступления на работу, стаж, пол. **Таблица Предметы** имеет следующие атрибуты: наименование, общее количество часов. **Таблица Группы** имеет следующие атрибуты: название, учебный год, семестр, стоимость обучения. **Таблица Учащиеся** имеет следующие атрибуты: фамилия, имя, отчество, адрес, возраст, пол, образовательное учреждение, в котором он учится, телефон родителя, группа. **Таблица расписание** имеет следующие атрибуты: год, день недели, время начала, время окончания, преподаватель, предмет, группа.
Вариант №28	**БД –Автошкола**. БД состоит из следующих таблиц: **Таблица Автомобили** имеет следующие атрибуты: номер автомобиля, марка, модель, цвет, год выпуска, дата регистрации в ГАИ. **Таблица Инструкторы** имеет следующие атрибуты: фамилия, имя, отчество, образование, категория, дата рождения, возраст, адрес, дата поступления на работу, стаж вождения, пол. **Таблица Группы** имеет следующие атрибуты: номер, учебный год, номер квартала, стоимость обучения. **Таблица Обучающиеся** имеет следующие атрибуты: фамилия, имя, отчество, паспортные данные, адрес, город, возраст, пол, группа. **Таблица Результат обучения** имеет следующие атрибуты: Обучающийся, Инструктор, Сдано (да/нет), Присвоенная категория
Вариант №29	**БД –Водно-оздоровительный комплекс**. БД состоит из следующих таблиц: **Таблица Инструкторы** имеет следующие атрибуты: фамилия, имя, отчество, образование, категория, дата рождения, возраст, адрес, дата поступления на работу, стаж, пол. **Таблица Спортивные группы** имеет следующие атрибуты: название, вид спорта, учебный год, семестр. **Таблица Участники** имеет следующие атрибуты: фамилия, имя, отчество, адрес, возраст, пол, образовательное учреждение, в котором он учится, телефон, телефон родителя, спортивная группа. **Таблица Расписание тренировок** имеет следующие атрибуты: год, день недели, время начала, время окончания, инструктор, группа. **Таблица Соревнования** имеет следующие атрибуты: дата, адрес проведения соревнования, участник.

Вариант №30	**БД – Страховая организация**. БД состоит из следующих таблиц: **Таблица Клиенты** имеет следующие атрибуты: фамилия, имя, отчество, паспортные данные, дата рождения, адрес, город, телефон. **Таблица Страховые агенты** имеет следующие атрибуты: фамилия, имя, отчество, отдел, год рождения, год поступления на работу, стаж, образование, пол, адрес, город, телефон. **Таблица Виды страхования** имеет следующие атрибуты: наименование, область применения, стоимость в год. **Таблица Страховые договоры** имеет следующие атрибуты: дата договора, агент, клиент, вид страхования, объект страхования, период страхования (кол-во лет). **Таблица Статистика страховых случаев** имеет следующие атрибуты: дата страхового случая, клиент, страховой договор, сумма компенсации.

Лабораторная работа № 3

Манипулирование данными: операторы INSERT, UPDATE, DELETE.

Цель лабораторной работы: Изучить используемый в реляционных СУБД оператор извлечения данных из таблиц. Изучить синтаксис языка модификации данных.

Методические указания

Новые данные добавляются оператором INSERT. Наименьшей единицей информации, которую можно добавить в реляционную базу данных, является одна строка таблицы.

Немного упрощенный синтаксис оператора INSERT имеет вид:

INSERT INTO Имя_Таблицы [(Колонка [, Колонка ...])]
{VALUES(<величина> [, <величина> ...]) | <оператор SELECT>};
<величина> = {:Переменная | <константа> | <выражение>
| <функция> | udf([<величина> [, <величина> ...]])
| NULL | USER}
<константа> = Число | 'Строка'
<функция> = CAST(<величина> AS <тип данных>)
UPPER(<величина>)
GEN_ID(Имя_Генератора, <величина>)
<выражение> = SQL выражение, возвращающее единичное значение

В этом описании можно выделить два варианта оператора:

1. Вставка одной строки. Для этого после ключевого слова VALUES в круглых скобках указывают вставляемые величины.

2. Вставка в таблицу нескольких строк, выбранных с помощью оператора SELECT * .

В этой лабораторной работе рассматривается только первый вариант оператора INSERT.

Пример, когда в качестве вставляемых величин применены константы:

INSERT INTO Person(Pr_ID, Pr_LastName, Pr_FirstName)
VALUES(150, 'Иванов', 'Петр');

2. Удаление существующих данных

Для удаления строк из таблицы используется оператор DELETE. Вот его

упрощенный синтаксис:

DELETE FROM Имя_Таблицы
[WHERE <условие поиска>];

<условие поиска> = как в операторе SELECT
Если не использовать предложение WHERE, то будут удалены все строки в таблице.
Пример. Удаление всех служащих:
DELETE FROM Employee;
Пример. Удаление всех людей с номерами 150 и больше:
DELETE FROM Person WHERE Pr_ID >= 150;

Отбирать строки для удаления не обязательно только на основании содержимого этих строк. Можно составить условие для удаляемых строк, опираясь на данные из других таблиц. Для составления таких условий необходимо сначала изучить оператор SELECT.

3. Обновление существующих данных

Оператор UPDATE обновляет значения одного или нескольких столбцов в выбранных строках одной таблицы. Строки для обновления указываются в предложении WHERE. Если пропустить предложение WHERE, то изменятся все строки таблицы.

UPDATE Имя_Таблицы
SET Колонка = <величина>[,
Колонка = <величина>...]
[WHERE <условие поиска>]
<величина> = { Колонка | :Переменная | <константа>
| <выражение> | <функция>
| udf([<величина> [, <величина> ...]]) | NULL | USER}
<выражение> = SQL выражение, возвращающее единичное значение
<условие поиска> = как в операторе SELECT

Примеры:

-- Увеличить зарплату всем служащим на 10%:

```
UPDATE Employee
SET Salary = 1.1*Salary;
```

/* Увеличить зарплату всем служащим, которые имеют

зарплату меньше 10000 на 15%: */

```
UPDATE Employee
SET Salary = 1.15*Salary;
WHERE Salary <= 10000;
```

Отбирать строки для изменения, как и для удаления, можно с использование подчиненного запроса SELECT, который позволит учитывать в условии поиска изменяемых строк данные из других таблиц.

Например, можно выполнить такой запрос: увеличить зарплату на 10% всем служащим, работающим в отделе продаж, которые обслужили за последний месяц клиентов больше чем в полтора раза, чем в среднем по их отделу.

Задания к лабораторной работе

1. Изучите методический материал.
2. Вставьте в любую таблицу своей БД новую строку данных, используя INSERT.
3. Обновите данные в таблице своей БД, используя UPDATE.
4. Удалите любую строку в свой БД, используя DROP.

Лабораторная работа № 4

Простые запросы на выборку (SELECT). Фильтрация строк (WHERE).

Цель лабораторной работы: Изучить используемый в реляционных СУБД оператор извлечения данных из таблиц.

Методические указания

В SQL имеется единственный оператор, который предназначен для выборки данных из базы данных. Оператор относится к подмножеству **DML**.

Ниже приведен почти полный синтаксис оператора **SELECT**.

SELECT [DISTINCT | ALL]
{ | <величина> [, <величина> ...]}*
[INTO :Переменная [, :Переменная ...]]
FROM <tableref> [, <tableref> ...]
[WHERE <условие поиска>]
[GROUP BY Колонка [, Колонка ...]]
[HAVING <условие поиска>]
[UNION [ALL] <select_expr>]
[ORDER BY <список сортировки>];
<величина> = {Колонка | :Переменная | <константа>
| <выражение> | <функция>
| udf ([<величина> [, <величина> ...]])
| NULL | USER} [AS Псевдоним]
<константа> = Число | 'Строка'
<выражение> = SQL выражение, возвращающее единичное значение
<функция> =
COUNT (| [ALL] <величина> | DISTINCT <величина>)*
| SUM ([ALL] <величина> | DISTINCT <величина>)
| AVG ([ALL] <величина> | DISTINCT <величина>)
| MAX ([ALL] <величина> | DISTINCT <величина>)
| MIN ([ALL] <величина> | DISTINCT <величина>)
| CAST(<величина> AS <тип данных>)
| UPPER (<величина>)
| GEN_ID (Имя_Генератора, <величина>)
<tableref> = {<joined_table> | table | view
| procedure[(<величина> [, <величина> ...])]}

[Псевдоним]
<joined_table> = <tableref> <join_type> JOIN <tableref>
ON <условие поиска> | (<joined_table>)
<join_type> = [INNER] | {LEFT | RIGHT | FULL } [OUTER]
<условие поиска> =
<величина> <оператор сравнения>
{<величина> | (<select_one>)}
| <величина> [NOT] BETWEEN <величина> AND <величина>
| <величина> [NOT] LIKE <величина>
| <величина> [NOT] IN
(<величина> [, <величина> ...] | <select_list>)
| <величина> IS [NOT] NULL
| <величина> {>= | <=} <величина>
| <величина> [NOT] {= | < | >} <величина>
| {ALL | SOME | ANY} (<select_list>)

| EXISTS (<select_expr>)
| SINGULAR (<select_expr>)
| <величина> [NOT] CONTAINING <величина>
| <величина> [NOT] STARTING [WITH] <величина>
| (<условие поиска>)
| NOT <условие поиска>
| <условие поиска>OR <условие поиска>
| <условие поиска>AND <условие поиска>
<оператор сравнения> =
{= | < | > | <= | >= | !< | !> | <> | !=}
<select_one> = оператор SELECT, выбирающий одну колонку и возвращающий ровно одно значение
<select_list> = оператор SELECT, выбирающий одну колонку, возвращающий ноль или много значений
<select_expr> = оператор SELECT, выбирающий несколько величин и возвращающий ноль или много значений
<список сортировки> =
{Колонка | Номер}
[ASC | DESC]
[, <список сортировки> ...]

Как видно из синтаксиса оператора **SELECT,** обязательными являются только предложение **SELECT** с перечнем выдаваемых колонок и предложение **FROM**.

Пример простейшего оператора SELECT:

-- *Выдать перечень всех служащих:*

SELECT * FROM Employee;

Примеры создания запросов с отбором строк по условию.

SQL дает возможность определить критерии отбора необходимых строк во фразе WHERE предложения SELECT. В этом случае строки исходных таблиц будут включены в результирующую только если строка соответствует указанным критериям. *Условие* - это выражение, которое может быть истинным или ложным (логическое выражение или предикат), то есть принимать логические значения TRUE или FALSE соответственно. В результирующую таблицу включаются только те строки, для которых указанное во фразе WHERE условие равно TRUE (иными словами, которые удовлетворяют заданному условию).

В случае одной таблицы механизм работы предложения SELECT с фразой WHERE следующий.

1. Из таблицы, указанной во фразе FROM, выбирается очередная строка.
2. Она проверяется на соответствие условию во фразе WHERE.
3. Если результат равен TRUE, строка включается в результирующую таблицу и форматируется в соответствии с фразой SELECT, а если он равен FALSE, строка пропускается.

Далее будут рассмотрены основные выражения, допустимые для условия во фразе WHERE.

Использование простейших условий

Простейшими считаются условия, в которых используются операторы сравнения и логические операторы.

Хотя такие условия являются простейшими в смысле семантики конструкций, они могут иметь довольно сложную структуру из многих операторов сравнений и вложенных друг в друга логических связок.

Операторы сравнения

Особенностью операторов сравнения является то, что независимо от типов операндов их результатом являются логические значения. Предположим, вы хотите получить список всех профессоров.

Пример. Вывести фамилии профессоров.

```
SELECT NAME_TEACHER AS ‘Список профессоров’
FROM TEACHER
    WHERE DOLGNOST = 'профессор';
```

Задание к лабораторной работе

1. Изучите методический материал

2. Для созданной базы данных, согласно номеру варианта, самостоятельно создать на языке Transact-SQL 5 запросов с использованием операторов сравнения.

Лабораторная работа № 5

Работа с функциями и выражениями в SELECT. Сортировка (ORDER BY).

Цель лабораторной работы: Изучить используемый в реляционных СУБД оператор извлечения данных из таблиц SELECT и выполнение группировки и сортировки данных.

Методические указания

Сортировка результирующих строк

Как мы уже отмечали, строки в таблицах базы данных неупорядочены. Также неупорядочены и строки результирующей таблицы запроса, однако для их упорядочения в предложении SELECT можно воспользоваться фразой ORDER BY. Она сортирует по значению указанных в ней столбцов (и выражений над столбцами) строки результирующей таблицы запроса. Синтаксис этой фразы следующий:

ORDER BY спецификация_сортировки[. спецификация_сортировки]...

где спецификация_сортировки имеет такой синтаксис:

выражение_сортировки [направление_сортировки] [положение_NULL]

Сортировать можно по столбцам (выражениям) тех типов, для которых определены операции сравнения. Это относится, в частности, к символьным строкам, числам и временным значениям. Можно указывать направление сортировки и место расположения строк, имеющих значение NULL для выражений сортировки.

Далее в этом уроке мы рассмотрим общие способы упорядочения результирующих строк.

Сортировка по столбцу или выражению

Сортировать строки результирующей таблицы запроса можно по отдельным столбцам, совокупности столбцов, а также по одному или нескольким выражениям над столбцами. Ниже рассматриваются все эти варианты.

Сортировка по столбцу

Простейший вариант сортировки - это сортировка по одному из столбцов результирующей таблицы.

Пример. Вывести алфавитный список фамилий профессоров и доцентов.

```
SELECT NAME_TEACHER
FROM TEACHER
wheRE LOwER(Dolgnost ) =’профессор’ OR
LOWER(Dolgnost ) = 'доцент'
ORDER BY NAME_TEACHER;
```

Сортировка по выражению над столбцами

Упорядочивать строки можно не только по значению столбца, но и по значению выражения над столбцами.

Пример. Вывести фамилии ассистентов и их зарплату по ее возрастанию.

```
SELECT Name_teacher, Salary + Rise
FROM TEACHER
WHERE LOWER(Dolgnost ) = 'ассистент’
ORDER BY Salary + Rise;
```

Направление сортировки

Во всех до сих пор приводимых примерах сортировка производилась в порядке возрастания значений. В SQL такой порядок определен по умолчанию. Однако есть возможность и явно указать направление сортировки с помощью ключевых слов ASC (по возрастанию) и DESC (по убыванию), которые следует располагать после имени сортируемого столбца (выражается).

Пример. Вывести фамилии ассистентов и дату их приема на работу по возрастанию даты.

```
SELECT Name_teacher, Data_hire
FROM TEACHER
WHERE LOWER(Dolgnost) = 'ассистент'
ORDER BY Data_hire ASC;
```

Задание к лабораторной работе

1. Изучите методический материал.

2. Для созданной базы данных, согласно номеру варианта, самостоятельно создать на языке Transact-SQL 3 многотабличных запросов:

- 1 запрос с использованием группировки по одному столбцу;

- 1 запрос на использование группировки по нескольким столбцам;

- 1 запрос с использованием сортировки по столбцу;

Лабораторная работа № 6

Соединение таблиц: INNER JOIN, табличные псевдонимы (алиасы).

Цель лабораторной работы: Изучить используемый в реляционных СУБД оператор извлечения данных из таблиц. Получить навыки работы с оператором SELECT в программе ‘SQL Server Managmant Studio’.

Методические указания

При проектировании стремятся создавать таблицы, в каждой из которых содержалась бы информация об одном и только одном типе сущности. Это облегчает модификацию базы данных и поддержку ее целостности. Именно так мы поступили, создавая учебную базу данных. Однако сущности могут быть взаимосвязанными. Кафедры связаны с факультетами по признаку вхождения в их состав, преподаватели работают на кафедрах, студенты учатся на кафедрах и т. д.

Связь между таблицами устанавливается за счет размещения специального столбца первичного ключа одной таблицы, которая называется родительской, в другой таблице, которая называется дочерней. Столбец (или совокупность столбцов) дочерней таблицы, определенный для связи с родительской таблицей, называется внешним ключом.

Наличие внешних ключей является основной для инициирования поиска по многим таблицам.

Одна из наиболее важных особенностей предложения SELECT — это способность использования связей между различными таблицами, а также вывода содержащейся в них информации. Операция, которая приводит к соединению из двух таблиц всех пар строк, для которых выполняется заданное условие, называется соединением таблиц. Для того чтобы указать соединяемые таблицы, их следует перечислить через запятую во фразе FROM.

Упрощенный синтаксис внутреннего соединения (стандарт SQL-92):

```
SELECT Колонка [, Колонка ...] | *
FROM <tableref_left> [INNER] JOIN <tableref_right>
[ON <условие поиска>]
    [WHERE <условие поиска>];
```

Синонимы таблиц

Синонимы таблиц часто используются для задания более лаконичного имени таблицы, по которому можно сослаться на нее в любых других местах запроса. Приведем пример.

Пример. Вывести названия кафедр, на которых имеются студенты со стипендией >200.

```
SELECT DISTINCT k.Name_Kafedru
FROM KAFEDRA k, STUDENT s
WHERE k.Kod_kafedru = s. Kod_kafedru AND s.Stipend > 400;
```

Задания к лабораторной работе

1. Изучите методический материал.

2. Для созданной базы данных, согласно номеру варианта, самостоятельно создать

на языке Transact-SQL многотабличный запрос с использованием внутреннего соединения.

Лабораторная работа № 7

Внешние соединения: LEFT/RIGHT JOIN. Перекрестное соединение (CROSS JOIN).

Цель лабораторной работы: Изучить используемый в реляционных СУБД оператор извлечения данных из таблиц. Получить навыки работы с оператором SELECT в программе ‘SQL Server Managmant Studio’.

Методические указания

При проектировании стремятся создавать таблицы, в каждой из которых содержалась бы информация об одном и только одном типе сущности. Это облегчает модификацию базы данных и поддержку ее целостности. Именно так мы поступили, создавая учебную базу данных. Однако сущности могут быть взаимосвязанными. Кафедры связаны с факультетами по признаку вхождения в их состав, преподаватели работают на кафедрах, студенты учатся на кафедрах и т. д.

Связь между таблицами устанавливается за счет размещения специального столбца первичного ключа одной таблицы, которая называется родительской, в другой таблице, которая называется дочерней. Столбец (или совокупность столбцов) дочерней таблицы, определенный для связи с родительской таблицей, называется внешним ключом.

Наличие внешних ключей является основной для инициирования поиска по многим таблицам.

Одна из наиболее важных особенностей предложения SELECT — это способность использования связей между различными таблицами, а также вывода содержащейся в них информации. Операция, которая приводит к соединению из двух таблиц всех пар строк, для которых выполняется заданное условие, называется соединением таблиц. Для того чтобы указать соединяемые таблицы, их следует перечислить через запятую во фразе FROM.

Внешние соединения. Внешние соединения бывают левыми, правыми и полными.

Если внешние соединения задаются в предложении FROM, они указываются с одним из следующих наборов ключевых слов.

LEFT JOIN или LEFT OUTER JOIN

Результирующий набор левого внешнего соединения включает все строки из левой таблицы, заданной в предложении LEFT OUTER, а не только те, в которых соединяемые столбцы соответствуют друг другу. Если строка в левой таблице не имеет совпадающей строки в правой таблице, результирующий набор строк содержит значения NULL для всех столбцов списка выбора из правой таблицы.

RIGHT JOIN или RIGHT OUTER JOIN

Правое внешнее соединение является обратным для левого внешнего соединения. Возвращаются все строки правой таблицы. Для левой таблицы возвращаются значения

NULL каждый раз, когда строка правой таблицы не имеет совпадающей строки в левой таблице.

FULL JOIN или FULL OUTER JOIN

Полное внешнее соединение возвращает все строки из правой и левой таблицы. Каждый раз, когда строка не имеет соответствия в другой таблице, столбцы списка выбора другой таблицы содержат значения NULL. Если между таблицами имеется соответствие, вся строка результирующего набора содержит значения данных из базовых таблиц.

Перекрестные с соединения

Перекрестное соединение возвращает все строки из левой таблицы. Каждая строка из левой таблицы соединяется со всеми строками из правой таблицы. Перекрестные соединения называются также декартовым произведением.

Таблицы или представления в предложении FROM могут указываться в любом порядке с внутренним соединением или полным внешним соединением. Однако важен порядок таблиц или представлений, заданных при использовании левого или правого внешнего соединения.

Задание к лабораторной работе

1. Изучите методический материал.

2. Для созданной базы данных, согласно номеру варианта, самостоятельно создать **на языке Transact-SQL 3 многотабличных запросов:**

- 1 запрос с использованием левого внешнего соединения;
- 1 запрос на использование правого внешнего соединения;
- 1 запрос с использованием симметричного соединения и удаление избыточности.

Лабораторная работа № 8

Агрегирование данных: GROUP BY и агрегатные функции (COUNT, SUM, AVG).

Цель лабораторной работы: Изучить используемый в реляционных СУБД оператор извлечения данных из таблиц SELECT и выполнение группировки и сортировки данных.

Методические указания

Агрегатные функции

Аргументами агрегатных функций могут быть как столбцы таблиц, так и результаты выражений над ними. Агрегатные функции и сами могут включаться в другие арифметические выражения. В стандарте SQL определены следующие виды агрегатных функций: унарные, бинарные, инверсного распределения, гипотетические функции множеств.

Мы будем рассматривать только определенные в стандарте SQL унарные агрегатные функции. Конкретные СУБД расширяют этот список.

AVG - среднее
MIN - минимум
CHECKSUM_AGG - Возвращает контрольную сумму значений в группе. Значения NULL не учитываются.
SUM - сумма
COUNT - количество
STDEV – среднее квадратическое отклонение
COUNT_BIG - Возвращает количество элементов в группе.
STDEVP - Возвращает статистическое стандартное отклонение всех значений в указанном выражении.
GROUPING - Указывает, является ли указанное выражение столбца в списке GROUP BY статистическим или нет. В результирующем наборе функция GROUPING возвращает 1 (статистическое выражение) или ноль (нестатистическое выражение).
VAR - дисперсия
GROUPING_ID - Представляет собой функцию, которая вычисляет уровень группирования.
VARP - Возвращает статистическую дисперсию для заполнения всех значений в указанном выражении.
MAX - максимум
Общий формат унарной агрегатной функции следующий:
имя_функции([ALL | DISTINCT] выражение) [FILTER (WHERE условие)]
где **DISTINCT** указывает, что функция должна рассматривать только различные значения аргумента, а ALL — все значения, включая повторяющиеся (этот вариант используется по умолчанию). Фраза FILTER позволяет дополнительно отобрать строки таблицы, столбец которой используется в качестве аргумента функции.

Агрегатные функции применяются во фразах SELECT и HAVING. Здесь мы рассмотрим их использование во фразе SELECT. В этом случае выражение в

аргументе функции применяется ко всем строкам входной таблицы фразы SELECT. Кроме того, во фразе SELECT нельзя использовать и агрегатные функции, и столбцы таблицы (или выражения с ними) при отсутствии фразы GROUP BY.

Функция COUNT

Функция COUNT имеет два формата. В первом случае возвращается количество строк входной таблицы, во втором случае — количество значений аргумента во входной таблице:

C0UNT(*)

C0UNT([DISTINCT | ALL] выражение)

Простейший способ использования этой функции - подсчет количества строк в таблице (всех или удовлетворяющих указанному условию). Для этого используется первый вариант синтаксиса.

Функция SUM

Эта агрегатная функция подсчитывает сумму значений аргумента для всех строк входной таблицы. Аргумент должен иметь числовой тип или быть временным промежутком. В качестве аргумента может выступать имя столбца или выражение над столбцами входной таблицы. В этой функции также допускается использовать ключевые слова DISTINCT и ALL. Приведем примеры.

Запрос. Какая суммарная ставка всех ассистентов?

```
SELECT SUM(Salary)
FROM TEACHER
WHERE LOWER(DOLGNOST) = 'ассистент';
```

Функция AVG

Агрегатная функция AVG подсчитывает среднее значение аргумента для всех строк входной таблицы. Аргумент должен иметь числовой тип или быть временным промежутком. В качестве аргумента может выступать имя столбца или выражение над столбцами входной таблицы. Допускается использовать ключевые слова DISTINCT и ALL. Приведем ряд примеров.

Запросы с группировкой строк

Часто при создании отчетов появляется необходимость в формировании промежуточных итоговых значений, то есть относящихся к данным не всей таблицы, а ее частей.

Именно для этого предназначена фраза GROUP BY. Она позволяет все множество строк таблицы разделить на группы по признаку равенства значений одного или нескольких столбцов (и выражений над ними).

Фраза GROUP BY должна располагаться вслед за фразой WHERE (если она отсутствует, то за фразой FROM).

Общий синтаксис фразы GROUP BY следующий:

GROUP BY выражение[, выражение]...

При наличии фразы GROUP BY фраза SELECT применяется к каждой группе, сформированной фразой группировки. В этом случае и действие агрегатных функций, указанных во фразе SELECT, будет распространяться не на всю результирующую таблицу, а только на строки в пределах каждой группы. Каждое выражение в списке фразы SELECT должно принимать единственное значение для группы, то есть оно может быть:

– константой;

– агрегатной функцией, которая оперирует всеми значениями аргумента в пределах группы и агрегирует их в одно значение (например, в сумму);

– выражением, идентичным стоящему во фразе GROUP BY;

– выражением, объединяющим приведенные выше варианты.

Рассмотрим возможности фразы GROUP BY, переходя от простых вариантов ее использования к более сложным.

Группировка по одному столбцу

Группировка по значениям одного столбца является самым простым вариантом использования фразы GROUP BY. Приведем примеры.

Для каждого корпуса подсчитать количество находящихся в нем кафедр.

```
SELECT NUM_KORPUSA AS “Корпус”,
COUNT(*) AS "К-во кафедр"
FROM KAFEDRA
GROUP BY NUM_KORPUSA ;
```

Группировка по нескольким столбцам

SQL позволяет группировать строки таблицы и по нескольким столбцам. В этом случае имена столбцов перечисляются во фразе GROUP BY через запятую.

Запрос. Для каждого факультета, расположенного в корпусе 1, вывести сколько учится студентов по каждой группе.

```
SELECT f.Name_faculteta,
s."GROUP", count(s."GROUP") AS "Кол-во студентов в группе"
FROM FACULTET f, KAFEDRA d, STUDENT s
WHERE f.KOD_FACULTETA = d.KOD_FACULTETA AND
d.KOD_kafedru = s.KOD_kafedru AND
d.NUM_KORPUSA = '1'
GROUP BY f.Name_faculteta,s."GROUP";
```

Задания для выполнения лабораторной работы

1. Изучите методический материал.

2. Для созданной базы данных, согласно номеру варианта, самостоятельно создать **на языке Transact-SQL 4 многотабличных запроса:**

- с использованием AVG

- с использованием SUM

- с использованием COUNT

- с использованием GROUP BY

Лабораторная работа № 9

Фильтрация групп. Условие HAVING. Сравнение WHERE и HAVING.

Цель лабораторной работы: Изучить используемый в реляционных СУБД оператор извлечения данных из таблиц SELECT и выполнение группировки и сортировки данных.

Методические указания

Условие отбора групп

Предположим, что нужно вывести номера кафедр, у которых суммарное количество работающих профессоров более 1. Приведенная ниже формулировка запроса является неверной:

```
SELECT KOD_kafedru
FROM TEACHER
WHERE count(dolgnost) > 1 and dolgnost='профессор'
GROUP BY KOD_kafedru;
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
WHERE count(dolgnost) > 3 and dolgnost='профессор';
*
```

Ошибка в строке 3;

CRA-00934: групповая функция здесь не разрешена

Дело в том, что фраза WHERE проверяет на соответствие условию строки исходных таблиц, а мы указали в ней агрегатную функцию. Для отбора строк среди полученных групп следует применять фразу HAVING. Она играет такую же роль для групп, что и фраза WHERE для исходных таблиц, и может использоваться лишь при наличии фразы GROUP BY. В предложении SELECT фразы WHERE, GROUP BY и HAVING обрабатываются в следующем порядке.

Фразой WHERE отбираются строки, удовлетворяющие указанному в ней условию.

Фраза GROUP BY группирует отобранные строки.

Фразой HAVING отбираются группы, удовлетворяющие указанному в ней условию. В связи с вышесказанным, предыдущий запрос необходимо записать так.

Перепишем тогда запрос так:

Запрос. Вывести номера кафедр, у которых суммарное количество работающих профессоров более 1.

```
SELECT KOD_kafedru as "Номер кафедры" ,Count(*) as "Кол-во профессоров на кафедре"
FROM TEACHER
WHERE dolgnost='профессор'
GROUP BY KOD_kafedru
  having count(dolgnost) > 1 ;
```

Задания для выполнения лабораторной работы

1. Изучите методический материал.

2. Для созданной базы данных, согласно номеру варианта, самостоятельно создать на языке Transact-SQL многотабличный запрос с использованием конструкции HAVING.

Лабораторная работа № 10

Простые подзапросы в условиях WHERE и HAVING.

Цель лабораторной работы: Изучить используемый в реляционных СУБД оператор извлечения данных из таблиц SELECT и выполнение группировки и сортировки данных.

Методические указания

Использование столбцов группировки во фразе HAVING

Рассмотрим использование во фразе HAVING условий отбора, заданных для группируемых столбцов (или выражений над ними). Для этого усложним предыдущий запрос.

Запрос. Вывести названия кафедр факультета математики и информатики, на которых работают один и более профессоров. Указать также количество профессоров и их суммарную зарплату.

```
SELECT d.Name_kafedru, Count(*), SUM(t.salary + t.Rise)
FROM FACULTET f, KAFEDRA d, TEACHER t
WHERE f.KOD_FACULTETA = d.KOD_FACULTETA AND
d.KOD_kafedru = t.KOD_kafedru AND
LOWER(f.Name_faculteta) = 'математики и информатики' AND
LOWER(t.Dolgnost ) = 'профессор'
GROUP BY d.Name_kafedru
HAVING COUNT(*) > 0;
```

Фраза HAVING без фразы GROUP BY

Выше мы указали, что фраза HAVING может использоваться лишь при наличии фразы GROUP BY. Из этого правила синтаксис SQL допускает только одно исключение: когда вся таблица интерпретируется как одна группа. В этом случае в списке фразы SELECT можно использовать только константы, агрегатные функции и выражения над ними. Приведем примеры.

Запрос. Если суммарная зарплата всех преподавателей превышает 15 000, вывести их минимальную ставку, максимальную надбавку и суммарную зарплату.

```
SELECT MIN(Salary), MAX(Rise), SUM(Salary + Rise)
FROM TEACHER
HAVING SUM(Salary + Rise) > 15000;
```

При наличии фразы WHERE сначала производится отбор строк согласно ее условию, и только после этого применяется условие фразы HAVING.

Запрос. Если суммарная зарплата всех ассистентов превышает 2500, вывести их среднюю ставку, среднюю надбавку и суммарную зарплату.

```
SELECT AVG(Salary), AVG(Rise), SUM(Salary + Rise)
FROM TEACHER
WHERE LOWER(Dolgnost ) = 'ассистент'
HAVING SUM(Salary + Rise) > 2500;
```

На практике фраза HAVING очень редко используется без фразы GROUP BY, из-за чего такая возможность предоставляется не во всех СУБД.

Задания для выполнения лабораторной работы

1. Изучите методический материал.

2. Для созданной базы данных, согласно номеру варианта, самостоятельно создать на языке Transact-SQL многотабличный запрос с использованием конструкции HAVING.

Лабораторная работа № 11

Сложные подзапросы: коррелированные подзапросы, подзапросы в FROM.

Цель лабораторной работы: Создавать подзапросы в MS SQL Server, изучить виды операторов, использованные в подзапросах. Применять в подзапросах функции и выражения.

Методические указания

Подзапросы представляют собой запрос внутри предиката другого запроса.

Подзапросы, внутренние или вложенные запросы – есть не что иное, как запрос внутри запроса. Обычно, подзапрос используется в конструкции WHERE. И, в большинстве случаев, подзапрос используется, когда вы можете получить значение с помощью запроса, но не знаете конкретного результата. Обычно, внутренний запрос генерирует значение, которое проверяется в предикате внешнего запроса, определяющего верно оно или нет.

Подзапросы являются альтернативным путем получения данных из множества таблиц.

Наряду с операторами сравнения такими, как =, <, >, >=, <= и др.,

Подзапрос иногда называется вложенным запросом. Утверждение, содержащее подзапрос называется родительским выражением. Строки, возвращенные подзапросом, используются родительским выражением.

Подзапросы могут использоваться для следующих целей:

1. Для определения множества строк, вставляемых в целевую таблицу выражениями INSERT или CREATE TABLE
2. Для определения одного или более значений, назначаемых существующим строкам в утверждении UPDATE
3. Для обеспечения необходимых условий в выражениях WHERE, HAVING утверждений SELECT, UPDATE, и DELETE

Чтобы определить таблицу, обрабатываемую запросом, подзапрос располагается после оператора FROM запроса вместо имени таблицы.

Можно использовать подзапросы вместо таблиц таким же образом и в утверждениях INSERT, UDPATE и DELETE. Подзапросы, используемые таким образом, могут применять переменные корреляции, но только если эти переменные определены внутри самого запроса и не содержат внешних ссылок

Существует два типа подзапросов: независимые и связанные.

В независимых подзапросах вложенный запрос логически выполняется ровно один раз.

Связанный запрос отличается от независимого тем, что его значение зависит от переменной, получаемой от внешнего запроса.

Таким образом, вложенный запрос связанного подзапроса выполняется каждый раз, когда система получает новую строку от внешнего запроса. В этом разделе приводится несколько примеров независимых подзапросов. Связанные

подзапросы рассматриваются далее в следующей статье совместно с оператором соединения JOIN.

Независимый подзапрос может применяться со следующими операторами:

- операторами сравнения;
- оператором IN;
- операторами ANY и ALL.

Подзапросы и операторы сравнения

Использование оператора равенства (=) в независимом подзапросе показано в примере ниже.

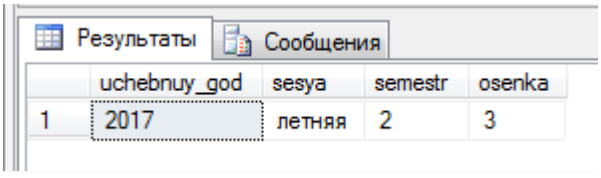
Запрос. Например, предположим, что мы знаем имя студента - Комаров, но не знаем значение его ключевого поля, но хотим извлечь все его оценки. Для этого выполним следующий запрос:

```
SELECT uchebnuy_god, sesya, semestr, osenka
  FROM SchoolReport
  WHERE kod_student =
   ( SELECT student_id
     FROM Student
     WHERE sutfname = 'Комаров');
```

Чтобы оценить внешний (основной) запрос, SQL сначала должен оценить внутренний запрос (или подзапрос) внутри предложения WHERE. Он делает это так как и должен делать запрос, имеющий единственную цель - отыскать через таблицу Студентов (Student) все строки, где поле sutfname равно значению Комаров, и затем извлечь все строки из таблицы Ведомости успеваемости (SchoolReport) соответствующие найденному номеру. Единственной найденной строкой естественно будет kod_student = 16. Однако SQL, не просто выдает это значение, а помещает его в предикат основного запроса вместо самого подзапроса, так чтобы предиката прочитал что

```
WHERE kod_student = 16
```

Результат выполнения запроса:



Значения, которые могут выдавать подзапросы

Скорее всего было бы удобнее, чтобы наш подзапрос в предыдущем примере возвращал одно и только одно значение.

Если бы мы выбирали поле **kod_student** с помощью условия "**WHERE city = 'Ростов-на-Дону'**" вместо " **WHERE sutfname = 'Комаров'**", то можно получить несколько различных значений. Это приводит к ситуации, когда в условии

предиката основного запроса невозможно применить для оценки верности или неверности, и запрос выдаст ошибку.

При использовании подзапросов в предикатах основанных на реляционных операторах (равенствах или неравенствах), вы должны убедиться, что использовали **подзапрос, который будет выдавать одну и только одну строку вывода.**

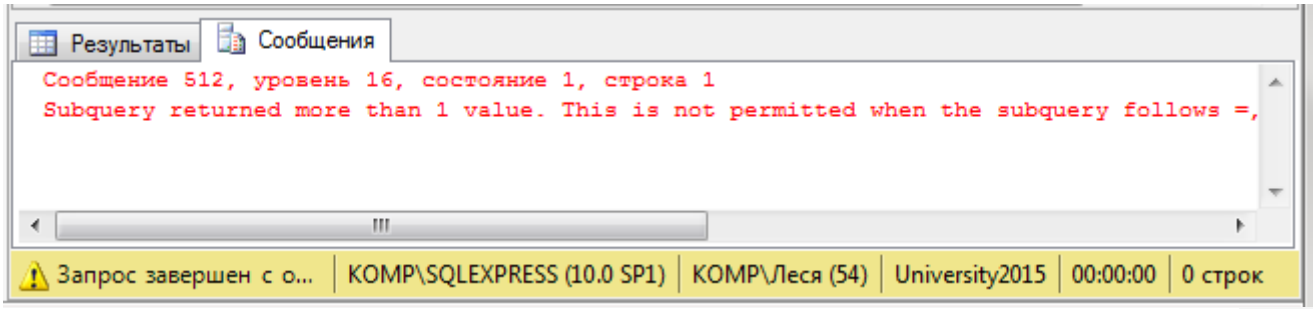
Если вы используете подзапрос, который не выводит никаких значений, то это расценивается верным запросом не имеющего ошибок. Но основной запрос не выведет никаких значений.

Подзапросы, которые не производят никакого вывода (или нулевой вывод) вынуждают рассматривать предикат ни как верный и ни как неверный, а как **неизвестный**. Однако, неизвестный предикат имеет тот же самый эффект, что и неверный: никакие строки не выбираются основным запросом.

Запрос. Вывести все оценки студентов, проживающих в городе Ростове-на-Дону:

```
SELECT uchebnuy_god, sesya, semestr, osenka
  FROM SchoolReport
  WHERE kod_student =
   ( SELECT student_id
     FROM Student
     WHERE city = 'Ростов-на-Дону');
```

Поскольку студентов, проживающих в городе Ростове-на-Дону в таблице Студентов несколько, то внутренний подзапрос выдаст несколько значений. А это значит, что условие внешнего запроса – неопределенно. Значит, СУБД отклонит выполнение данного запроса и выдаст ошибку.



Задания для выполнения лабораторной работы

1. Изучите методический материал.

2. Для созданной базы данных, согласно номеру варианта, самостоятельно создать на языке Transact-SQL сложный подзапрос.

Лабораторная работа № 12

Предикаты IN, EXISTS, ANY/SOME, ALL. Операция UNION.

Цель лабораторной работы: Создавать подзапросы в MS SQL Server, изучить виды операторов, использованные в подзапросах. Применять в подзапросах функции и выражения.

Методические указания

Проверка на принадлежность множеству

Оператор IN позволяет проверить, входит ли значение в указанное множество значений. В простейшем случае этот оператор имеет следующий синтаксис:

имя_столбца [NOT] IN (список_значений)

Здесь список значений представляет собой перечень разделенных запятыми констант, тип которых должен соответствовать типу столбца, чье имя приведено слева. Семантика этого предиката такова: он принимает значение TRUE, если значение столбца соответствует одной из констант списка. Приведем пример.

Запрос. Вывести названия и номер корпуса кафедр, расположенных в корпусах 1, 3, 12.

```
SELECT Name_Kafedru, NUM_KORPUSA
FROM  KAFEDRA
WHERE NUM_KORPUSA IN ('1', '3', '12');
```

Использование отрицания

Так как предикат IN возвращает истинностное значение, к нему можно применить логическое отрицание. Для этого следует воспользоваться нотацией NOT IN. В этом случае предикат будет истинным, если значение столбца не входит в указанный список.

Использование выражений над столбцами

В левой части оператора IN вместо имени столбца можно использовать любое допустимое над столбцами таблицы выражение языка.

Более того, элементами списка в правой части оператора IN тоже могут быть выражения над столбцами, как это показано в следующем примере:

Запрос.

```
SELECT NAME_TEACHER, Salary, Salary + Rise
FROM  TEACHER
WHERE Salary + Rise IN (Salary + 100, Salary + 200, Salary + 300, Salary + 400, Salary + 500);
```

Проверка на принадлежность диапазону значений

Еще одной формой проверки вхождения элемента во множество является

проверка на его принадлежность диапазону значений. Для этого применяется предикат BETWEEN, который определяет нахождение значения столбца между указанными минимальным и максимальным значениями. Синтаксис предиката следующий:

имя_столбца [NOT] BETWEEN минимум AND максимум

Проверять можно значения числовых, строковых и временных типов (для строк символов предполагается алфавитное упорядочение). Оператор BETWEEN является включающим - это означает, что крайние значения диапазона включаются в допустимые.

Запрос. Вывести фамилии преподавателей со ставкой в диапазоне 1000-2000.

```
SELECT NAME_TEACHER
FROM  TEACHER
WHERE Salary BETWEEN 1000 AND 2000;
```

Использование строковых значений

Использование в операторе BETWEEN в качестве границ диапазона строковых значений имеет особенности, связанные с упорядочением (это же относится и к другим операторам сравнения).

Среди строк результата нет фамилий, начинающихся на букву ‘Л’. Дело в том, что при сравнении строк символов разной длины SQL предварительно дополняет более короткую строку символами пробела, а он в упорядочениях символов предшествует всем остальным. Поэтому строка, состоящая из буквы ‘Л’ (дополненная пробелами), всегда будет меньше любой другой строки, в которой за начальной буквой 'Л’ следуют отличающиеся от пробела символы.

Чтобы это учесть, в качестве верхнего значения диапазона лучше всего указывать следующую по алфавиту букву (в данном случае — ‘М’).

Использование отрицания

Так как предикат BETWEEN возвращает истинностное значение, к нему можно применить логическое отрицание. Для этого следует воспользоваться нотацией NOT BETWEEN, в которой предикат будет истинным, только если значение столбца не входит в указанный диапазон. Представление отрицания нотацией NOT BETWEEN введено в язык для большей наглядности, так как с предикатом BETWEEN можно стандартным образом использовать логический оператор NOT (то есть ставить отрицание ко всему выражению, а не к предикату):

NOT (имя_столбца BETWEEN минимум AND максимум)

Круглые скобки в данном случае можно и опустить, так как они не меняют порядка исполнения операторов. В нотации NOT BETWEEN крайние значения в диапазон не включаются.

Использование выражений над столбцами

Как и в предикате IN, вместо имени столбца и границ диапазона можно использовать любое допустимое в языке выражение над столбцами таблицы,

включая и функции.

Запрос. Вывести данные преподавателей, зарплата которых (ставка + надбавка) находится в диапазоне от удвоенной величины надбавки до утроенной надбавки плюс 50.

```
SELECT NAME_TEACHER, Salary + Rise, 2 * Rise, 3 * Rise + 50
FROM  TEACHER
WHERE Salary + Rise BETWEEN 2 * Rise AND 3 * Rise + 50;
```

Задания для выполнения лабораторной работы

1. Изучите методический материал.

2. Для созданной базы данных, согласно номеру варианта, самостоятельно создать на языке Transact-SQL запросы с использованием конструкций IN, BETWEEN

Лабораторная работа № 13

Предикаты LIKE для поиска по шаблону. Работа с NULL (IS NULL, IS NOT NULL)

Цель лабораторной работы: Изучить используемый в реляционных СУБД оператор извлечения данных из таблиц. Получить навыки работы с оператором SELECT.

Методические указания

Проверка на соответствие шаблону

Когда необходимо отобрать строки таблицы, в которых значение некоторого столбца совпадает с заданной строкой символов, следует использовать обычное сравнение, как это показано выше. Однако во многих случаях можно не знать точное представление в базе данных интересующего значения. Название одной и той же кафедры, например, может храниться в одном из следующих вариантов: ‘базы данных’, ‘организация баз данных’, ‘информационные системы и базы данных’, ‘базы данных и знаний’.

Такая же ситуация возникает, когда не известно точное написание фамилии преподавателя, название дисциплины, факультета и т. п. Специально для таких случаев предназначен оператор сравнения LIKE, позволяющий отобрать из таблицы строки на основе частичного соответствия. Упрощенный синтаксис оператора следующий:

имя_столбца [NOT] LIKE шаблон [ESCAPE символ_пропуска]

Его можно использовать только с символьными значениями.

Использование шаблона

Оператор LIKE сравнивает значение столбца с множеством значений, определяемых шаблоном. Он представляет собой строку, в которой помимо обычных символов, составляющих основу поискового выражения, можно использовать так называемые подстановочные символы (иногда они называются групповыми символами). Имеется всего два подстановочных символа, различающихся тем, что именно на их месте может стоять:

% — любая последовательность символов, включая их отсутствие;

_ — один любой символ.

Подстановочные символы могут находиться в любом месте шаблона в любом наборе.

Например, шаблону '%Иван%' соответствуют строки 'Иван', 'Иванов', 'Иванченко', 'Петр Иванович', а шаблону 'л_с_' - 'лист', 'леса', 'лоск' (ноне 'лес', 'листок', 'плес').

Оператор LIKE, как и все другие, работающие с символьными строками, чувствителен к регистру букв, поэтому при его использовании мы рекомендуем использовать уже известные вам функции UPPER() и L0WER().

Запрос. Найти фамилии преподавателей на букву ‘М’.

```
SELECT NAME_TEACHER
FROM TEACHER
WHERE UPPER(NAME_TEACHER) LIKE 'M%';
```

Имейте в виду, что если вы запишете условие фразы WHERE как UPPER(NAME_TEACHER) = 'М%' или даже как 'М%' LIKE UPPER(NAME_TEACHER), фамилии преподавателей будут сравниваться со строкой ' М%'. Во втором случае выражение является синтаксически правильным оператором LIKE, однако в нем строка 'М%' не выступает в качестве шаблона, так как расположена перед ключевым словом LIKE.

Запрос. Указать преподавателей, в фамилиях которых первой буквой является ‘М’, а четвертой – ‘ы’.

```
SELECT NAME_TEACHER
FROM TEACHER
WHERE NAME_TEACHER LIKE 'М__ ы%’;
```

Запрос. Вывести названия кафедр, в которых присутствует словосочетание ‘анализ’ (в различных грамматических формах).

```
SELECT Name_Kafedru
FROM KAFEDRA
WHERE LOWER(Name_Kafedru) LIKE '%анализ%';
```

В левой части оператора LIKE может находиться не только имя столбца, но и любое допустимое над столбцами выражение, как это показано в следующем примере.

Запрос. Указать преподавателей, в фамилию и название должности которых входит в сумме не меньше пяти букв ‘о’.

```
SELECT NAME_TEACHER, DOLGNOST
FROM TEACHER
WHERE LOWER(NAME_TEACHER + DOLGNOST) LIKE '%о%о%о%о%о%';
```

Проверка на неопределенное значение

Как мы уже отмечали, наличие значения NULL во фразе WHERE приводит к тому, что условие принимает истинностное значение UNKNOWN и соответствующая строка не включается в результат. Детальное описание работы с неопределенным значением вы можете найти в уроке 10, а здесь мы покажем, как обрабатывать значение NULL во фразе WHERE.

Чтобы проверить столбец на неопределенное значение, следует применить унарный оператор IS NULL, имеющий такой синтаксис:

имя_столбца IS [NOT] NULL

Этот оператор принимает истинностное значение TRUE, если столбец имеет неопределенное значение, и FALSE — в противном случае. В нотации IS NOT NULL его действие обратное.

Запрос. Вывести фамилии преподавателей, у которых не задан номер телефона или идентификационный код.

```
SELECT NAME_TEACHER, INDEF_KOD, TEL_TEACHER
FROM TEACHER
WHERE INDEF_KOD IS NULL OR TEL_TEACHER IS NULL;
```

Задания для выполнения лабораторной работы

1. Изучите методический материал.

2. Для созданной базы данных, согласно номеру варианта, самостоятельно создать на языке Transact-SQL запросы с использованием конструкций шаблона.

Лабораторная работа № 14

Реализация операций реляционной алгебры (проекция, выборка, соединение, деление) на SQL.

Цель лабораторной работы: Изучить операции реляционной алгебры в SQL.

Методические указания

В языке SQL можно использовать обычные операции над множествами — объединение (union), пересечение (intersection) и разность (difference), — позволяющие комбинировать результаты выполнения двух и более запросов в единую результирующую таблицу.

Все эти операции над множествами графически представлены на слайде. На таблицы, которые могут комбинироваться с помощью операций над множествами, накладываются определенные ограничения. Самое важное из них состоит в том, что таблицы должны быть совместимы по операции — т.е. они должны иметь одну и ту же структуру. Это означает, что таблицы должны иметь одинаковое количество столбцов, причем в соответствующих столбцах должны размещаться данные одного и того же типа и длины. Обязанность убедиться в том, что значения данных соответствующих столбцов принадлежат одному и тому же домену, возлагается на пользователя. Например, мало смысла в том, чтобы объединять столбец с данными о возрасте работников с информацией о количестве комнат в сдаваемых в аренду объектах, хотя оба столбца будут иметь один и тот же тип данных — SMALLINT

Три операции над множествами, предусмотренные стандартом ISO, носят название UNION, INTERSECT и EXCEPT.

При указании в формате конструкции ключевого слова CORRESPONDING BY операция над множествами выполняется для указанных столбцов. Если задано только ключевое слово CORRESPONDING, а конструкция BY отсутствует, операция над множествами выполняется для столбцов, которые являются общими для обеих таблиц. Если указано ключевое слово ALL, результирующая таблица может содержать повторяющиеся строки.

Одни диалекты языка SQL не поддерживают операций INTERSECT и EXCEPT, а в других вместо ключевого слова EXCEPT используется ключевое слово MINUS.

Остальные операции реляционной алгебры реализуются в представленных ниже языковых конструкциях.

Операция декартового произведения отношений реализуется в предложении FROM команды SELECT. Перечисление в предложении FROM нескольких реляционных таблиц через запятую приведет к их перемножению.

Операция естественного соединения реализуется в предложении FROM команды SELECT путем соединения таблиц с помощью ключевых слов INNER JOIN, LEFT JOIN, RIGHT JOIN, OUTER JOIN с указанием столбцов связи.

Операция проекции задается перечислением столбцов таблицы предложения FROM. Для того, чтобы обеспечить уникальность каждой строки результата, используется ключевое слово DISTINCT.

Операция выборки определяется предложением WHERE команды SELECT. Предикат выборки задается в форме логического выражения.

Задания для выполнения лабораторной работы

1. Изучить методический материал.

2. Подумайте какие операции реляционной алгебры применимы к вашей базе данных.

Лабораторная работа № 15

Создание и использование представлений (VIEW)

Цель лабораторной работы: Изучение назначения представлений баз данных, синтаксиса и семантики команд языка Transact-SQL для их создания, изменения и удаления, системных хранимых процедур для получения информации о представлениях.

Методические указания

Представление (View) для пользователей баз данных выглядит как таблица, но при этом оно не содержит данных, а лишь представляет данные, расположенные в одной или нескольких таблицах. Таким образом, представления – это виртуальные таблицы, определяемые запросом на языке Transact-SQL. Подобно реальным таблицам представления содержат именованные столбцы и строки с данными, которые они динамически выбирают из таблиц и предлагают эти данные пользователю для просмотра. Представления часто применяются для ограничения доступа к конфеденциальным данным в таблицах баз данных. Когда в представление не включается столбец исходной таблицы, то считают, что на таблицу наложен вертикальный фильтр. Если в SQL – запросе установлено одно или несколько условий для выборки строк, то считают, что на таблицу наложен горизонтальный фильтр.

Представление может выбирать данные из других представлений, которые, в свою очередь, могут также основываться на представлениях или таблицах. Вложенность представлений не должна превышать 32. Представления можно создавать, используя базы данных одного сервера (текущего). Максимальное количество столбцов в представлении равно 1024. Представление не может ссылаться на временные таблицы. Кроме того, нельзя создавать временное представление.

Для представления нельзя определить ограничения целостности, триггеры, правила, или умолчания, а также создать обычный или полнотекстовый индекс.

В основном представления используются для выборки данных. Однако с помощью представлений можно выполнять и изменение данных в таблицах, на основе которых построено представление, при этом требуется соблюдение ряда правил: представление должно содержать, как минимум, одну таблицу в параметре FROM команды SELECT, не разрешается использование функций агрегирования и др.

Как и для таблиц, для представлений можно определить следующие права доступа:

SELECT – просмотр данных;
INSERT – добавление данных через представления;
UPDATE – изменение данных в исходных таблицах;
DELETE –удаление данных в исходных таблицах.

Чтобы иметь возможность создавать представления, надо обладать правами владельца баз данных и иметь соответствующие разрешения для любых таблиц или представлений, упомянутых "в запросе на создание этого представления.

Для создания представления используется следующая команда Transact-SQL:

CREATE VIEW [Имя базы данных.] [имя владельца.]
Имя представления
[(Имя колонки [,... n])]
[WITH{ENCRYPITION\SHEMABINDING

VIEW_METADATA}
AS Команда SELECT
[WITH CHECK OPTION]

Если в команде не заданы имена колонок представления, то они определяются по именам выбираемых колонок в команде SELECT. Параметр ENCRYPTION скрывает код создания этого представления, а параметр SHEMABINDING обеспечивает контроль структуры исходных объектов, к которым обращается оператор SELECT. Опция WITH CHEC OPTION не позволяет изменять строки таким образом, чтобы они исчезли при отборе командой SELECT.

Задания для выполнения лабораторной работы

1. Изучите методический материал.

2. Создать представление auth, ссылающегося на таблицу authors базы данных Pubs и содержащего идентификационный номер автора au_lname и телефон phone, при этом отобразить только авторов из Калифорнии ‘CA’ или авторов, не подписавших контракт с издательством, выполнив следующую команду:

CREATE VIEW auth
WITH SHEMABINDING
AS SELECT au_id, au_lname, au_fname, phone
FROM dbo. Authors
WHERE state = ‘CA’ OR contract = 0
WITH CHECK OPTION.

Лабораторная работа № 16

Написание простых хранимых процедур и функций. Использование переменных

Цель лабораторной работы: Изучение синтаксиса и семантики функций и хранимых процедур Transact– SQL

Методические указания

Функции и хранимые процедуры используются в SQL Server для реализации на языке Transact-SQL сложных часто используемых алгоритмов обработки данных или различных административных действий создания учетных записей, получения информации об объектах базы данных, управления свойствами сервера и баз данных, управления подсистемой репликации и автоматизации и т.д. Они хранятся в виде исходного текста и являются программными модулями, существующими независимо от таблиц или каких-либо других объектов баз данных. Исключением являются расширенные хранимые процедуры, которые хранятся в двоичном формате в виде динамически подключаемых библиотек типа *.dll и создаются с помощью других языков программирования с использованием интерфейса SQL Server Open Data Services API. Такие процедуры подключаются, отключаются и выгружаются соответственно командами sp_addextendedproc, sp_dropextendedproc и DBCC dlname (FREE), где dllno me_имя dll_библиотеки.

Хранение функций и хранимых процедур в виде исходных модулей языка Transact – SQL на сервере и в соответствующих базах данных позволяет уменьшить размр запроса, посылаемого по сети от клиента на сервер, а следовательно, и нагрузку на сеть, что повышает общую производительность системы. Это также позволяет упростить сопровождение программных комплексов и внесение изменений в исходный текст модулей, причем большинство изменений не отразится на работоспособности клиентских приложений.

Локальные хранимые процедуры должны иметь имя, начинающееся с символа #, и могут быть вызваны только из того соединения, в котором они были созданы. Они автоматически удаляются при отключении пользователя, перезапуске или остановке сервера.

Глобальные хранимые процедуры должны иметь имя, начинающееся с символов ##, и доступны для любых соединений с экземпляром сервера, на котором они были созданы. Они удаляются либо при закрытии соединения, в контексте которого они были созданы, либо автоматически – при перезапуске или остановке сервера.

Создание, изменение и удаление функций и хранимых процедур производится соответственно командами:

CREATE FUNCTION,
CREATE PROCEDURE,
ALTER FUNCTION,
ALTER PROCEDURE,
DROP FUNCTION,
DROP PROCEDURE.

При создании функции указывается тип возвращаемого значения и в теле функции обязательно задается команда RETURN, за которой следует выражения для вычисления возвращаемого значения. В теле процедуры использование команды RETURN (конечно, без последующего выражения) вовсе не обязательно. Когда этой команды нет, выход из процедуры будет происходить после исполнения последней команды процедуры.

Тело, как функции, так и хранимой процедуры начинается ключевым словом AS.

Задания для выполнения лабораторной работы

1. Изучите методический материал.
2. Создать функцию для выполнения четырех арифметических операций “+”, “- ”, “*” и “/” над целыми операндами типа bigint, выполнив кодирование и проверку:

2.1. Кодирование

```
CREATE FUNCTION Calculator
(@ Opd1 bigint,
@ Opd2 bigint,
@ Oprt char(1) = “*”)
RETURNS bigint
AS
BEGINT
DECLARE @ Result bigint
SET @ Result =
CASE @ Oprt
WHEN “+” THEN @ Opd1 + @ Opd2
WHEN “-” THEN @ Opd1 - @ Opd2
WHEN “*” THEN @ Opd1 * @ Opd2
WHEN “/” THEN @ Opd1 / @ Opd2
ELSE 0
END
Return @ Result
END
```

2.2. Тестирование

```
SELECT dbo.Calculator(4,5, ‘+’),
dbo. Calculator(3,7, ‘*’) – dbo.Calculator(64,4,’/’)*2.
9 -11
(1 row (s) affected)
```

Лабораторная работа № 17

Управление транзакциями. BEGIN, COMMIT, ROLLBACK. Обработка ошибок.

Цель лабораторной работы: Изучение способов обеспечения надежной работы SQL Server с помощью механизма транзакций и контрольных точек, приобретение навыков управления локальными и распределенными транзакциями различных видов, а также ознакомление с физической и логической архитектурой журнала транзакций и способами восстановления баз данных.

Методические указания

Одним из способов повышения надежности работы системы MS SQL Server 2000 является применение встроенного в систему механизма транзакций и контрольных точек и умелое его управление.

Транзакция – это одна или несколько последовательных команд языка Transact – SQL, образующих логически завершенный пакет и выполняемых как единое целое. Если по какой-либо причине хотя бы одна из команд пакета не выполняется, то происходит откат системы к состоянию, в котором она была до начала транзакции, и транзакция считается не выполненной. По умолчанию каждая команда выполняется как самостоятельная транзакция. При необходимости в пакете можно явно указать начало и конец транзакции.

SQL Server поддерживает три вида определений транзакций: явное, автоматическое и подразумеваемое.

Для управления явными транзакциями применяют команды:

BEGIN TRANSACTION [Имя транзакции] – начало транзакции;

COMMIT TRANSACTION [Имя транзакции] – конец транзакции;

ROLLBACK TRANSACTION [Имя транзакции] – откат транзакции;

В последних двух командах слово TRANSACTION можно либо опускать, либо заменять словом WORK. Во всех трех командах допускается использование сокращения

TRAN вместо слова TRANSACTION и переменной строкового типа, которой присваивается имя транзакции, вместо непосредственного указания этого имени. Дополнительный аргумент WITH MARK ‘Описание’ позволяет специальным образом маркировать транзакцию в журнале транзакций, что используется при восстановлении базы данных.

Если команды явного определения транзакций не используются, то сервер работает в одном из двух режимов:

а) в режиме автоматического начала транзакций, в котором каждая рассматривается как отдельная транзакция, при этом если команда выполнена успешно, то сделанные ей изменения фиксируются, и выполняется следующая команда, в противном случае производится откат транзакции и выполнение команды повторяется;

б) в режиме неявного начала транзакции, когда начала транзакции не указывается, а ее завершение задается явно командой COMMIT или инициируется командами:

ALTER TABLE,
CREATE,
DELETE,
DROP,
FETCH,
GRANT,
INSERT,
OPEN,
REVOKE,
SELECT,
TRANCATE
TABLE и UPDATE;

в этом режиме можно использовать команды COMMIT и ROLLBACK; после завершения текущей транзакции, начинается выполнение следующей, если не был задан откат транзакции.

Режим автоматического начала транзакций устанавливается по умолчанию или командой.

Задания для выполнения лабораторной работы

Задание 1. Проверить режимы автоматического начала транзакций и неявного начала транзакций, используя переключатель IMPLICIT_TRANSACTION и команду COMMIT.
Задание 2. Создать несколькими способами распределенные транзакции и убедиться в корректности их выполнения.
Задание 3. Создать вложенные транзакции, выполнив следующие команды:

```
CREATE TABLE #aaa (cola int) -- 0-й уровень
BEGIN TRAN -- 1-й уровень
INSERT INTO #aaaVALUES (111)
BEGIN TRAN -- 2-й уровень
INSERT INTO #aaaVALUES (222)
BEGIN TRAN -- 3-й уровень
INSERT INTO #aaaVALUES (333)
SELECT * FROM #aaa
SELECT ‘Вложенность транзкций’, @@TRANCOUNT
ROLLBACK TRAN
SELECT * FROM #aaa -- откат на 0-й уровень
SELECT ‘Вложенность транзакций’, @@TRANCOUNT
```

Проанализировать полученные результаты.

Лабораторная работа № 18

Создание и анализ отчетов на основе SQL-запросов. Экспорт результатов.

Цель лабораторной работы: изучение способов создания отчётов и экспорта данных в SQL.

Методические указания

На основе SQL-запросов можно создавать отчёты, анализировать результаты и экспортировать данные. Для этого используются разные инструменты, например конструкторы отчётов, инструменты для анализа запросов и программы для экспорта результатов.

Создание отчетов

Некоторые инструменты для создания отчётов на основе SQL-запросов:

- Конструкторы отчётов. В них прописывается SQL-запрос, добавляются и настраиваются колонки отчёта, настраиваются параметры для запуска отчёта. Например, в конструкторе отчётов есть блоки «Скрипт», «Таблица» и «Параметры».
- Построители отчётов. В них можно указать данные в области данных отчёта и настроить макет отчёта. Например, в SQL Server Reporting Services (SSRS) есть режим конструктора отчётов, где нужно указать данные и настроить макет.
- Инструменты для анализа данных. Некоторые из них позволяют использовать SQL для извлечения данных из базы данных, а также сохранять результаты в виде отчёта. Например, Tableau, Crystal Reports, Power BI, Zoho Analytics, Jasper Reports.

Анализ

Для анализа результатов SQL-запросов используются, например:

- Инструмент EXPLAIN ANALYZE. Показывает пошаговый план выполнения запроса, включая используемые индексы, количество обработанных строк и затраченное время.
- План выполнения запросов. Показывает, что происходит внутри запроса. Например, в SQL Server Management Studio есть кнопка с иконкой плана выполнения.
- Инструменты для анализа производительности. Например, горизонтальные диаграммы, которые показывают ключевые метрики, влияющие на производительность SQL-запросов: время выполнения, использование CPU, потребление памяти, I/O-операции и количество обработанных строк.

Экспорт

Результаты SQL-запросов можно экспортировать в разные форматы, например:

- В формате Excel. Некоторые программы, такие как Microsoft SQL Server Management Studio, MySQL Workbench или phpMyAdmin, позволяют экспортировать результаты SQL-запросов в формате Excel. Для этого необходимо выполнить SQL-запрос, затем выбрать опцию экспорта в Excel в меню программы.

- В формате CSV. Можно выполнить SQL-запрос с использованием ключевого слова INTO OUTFILE, чтобы сохранить результаты запроса в CSV-файл. Например:

*SELECT * FROM таблица INTO OUTFILE 'путь_к_файлу.csv' FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n'*

После выполнения этого запроса CSV-файл будет сохранён в указанном каталоге.

- В текстовом файле. В некоторых программах, например в SQL Server Management Studio (SSMS), есть возможность сохранить результаты запроса в текстовый файл. Для этого нужно выбрать опцию «Результат в файл» и указать имя и путь.

Задания для выполнения лабораторной работы

1. Изучить методический материал.

2. Создайте SQL-запрос к своей базе данных.

Лабораторная работа № 19

Оптимизация запросов: создание индексов, использование EXPLAIN для анализа плана выполнения.

Цель лабораторной работы: изучение оптимизации запров в SQL.

Индексы — это важный инструмент для оптимизации SQL-запросов, который позволяет значительно ускорить поиск данных и снизить нагрузку на базу. Они работают как указатель, который направляет СУБД к нужным строкам вместо того, чтобы сканировать всю таблицу. Оптимизация SQL-запросов с помощью индексов позволяет повысить производительность и сократить время выполнения сложных выборок.

Что такое индексы и как они работают?

Индекс — это дополнительная структура данных, созданная на основе столбцов таблицы. При добавлении индекса СУБД организует данные таким образом, чтобы ускорить операции чтения. Основной принцип работы индексов заключается в упрощении поиска с помощью упорядоченных структур, по типу B-деревьев.

Так, без индекса при запросе типа:

```
SELECT *
FROM employees
WHERE department = 'HR';
```

СУБД выполняет полное сканирование таблицы (Full Table Scan). Если таблица содержит миллионы строк, это может существенно снизить производительность.

С индексом, например:

```
CREATE INDEX idx_department ON employees(department);
```

Запрос вместо полной выборки использует индекс для мгновенного поиска всех строк, соответствующих условию department = ‘HR’.

Есть несколько вариантов индексов. Рассмотрим подробнее:

1. Первичные индексы

Первичный индекс создаётся автоматически для каждой таблицы, в которой объявлен PRIMARY KEY. Этот индекс гарантирует уникальность значений и позволяет быстро находить записи по основному идентификатору.

Пример:

```
CREATE TABLE orders (
   order_id SERIAL PRIMARY KEY,
   customer_id INT,
   order_date DATE
);
```

Для order_id автоматически создаётся первичный индекс, что ускоряет запросы по этой колонке.

2. Уникальные индексы

Уникальные индексы предотвращают дублирование данных. Например, это актуально для столбцов, в которых должны храниться уникальные значения (адрес электронной почты в нашем случае):

```
CREATE UNIQUE INDEX idx_email ON users(email);
```

Этот индекс позволяет ускорить запросы вроде:

```
SELECT *
FROM users
WHERE email = 'example@example.com';
```

и гарантирует, что в таблице не будет двух пользователей с одинаковым email.

3. Составные индексы

Составной индекс создаётся для нескольких столбцов одновременно. Он полезен для ускорения запросов, которые используют фильтры по 2-м и более полям.

Пример:

```
CREATE INDEX idx_orders_customer_date ON orders(customer_id, order_date);
```

Этот индекс ускорит запросы, которые фильтруют данные сразу по клиенту и дате. Вот так:

```
SELECT *
FROM orders
WHERE customer_id = 101 AND order_date >= '2024-01-01';
```

Как избежать чрезмерного количества индексов

Индексы занимают место в памяти и замедляют операции, поскольку их данные необходимо обновлять при каждой модификации таблицы.

1. Анализируйте, какие запросы наиболее часто используются, и создавайте индексы только для ключевых столбцов.

2. Периодически проверяйте использование индексов. Команда EXPLAIN поможет это определить.

3. Удаляйте неиспользуемые индексы.

Пример удаления:

```
DROP INDEX idx_unused_index;
```

Примеры использования индексов для ускорения поиска данных

Пример 1. Ускорение фильтрации с WHERE

Индекс улучшает производительность запроса:

```
CREATE INDEX idx_price ON products(price);
SELECT *
FROM products
WHERE price > 1000;
```

Без индекса фильтрация займёт больше времени, так как придётся проверять каждую строку таблицы.

Пример 2. Оптимизация сортировки сложных запросов с ORDER BY

Индекс помогает при сортировке данных и оптимизации запросов. Она выполняется быстрее, так как все уже упорядочено:

```
CREATE INDEX idx_salary ON employees(salary);
SELECT name, salary
FROM employees
ORDER BY salary DESC;
```

Пример 3. Поиск по нескольким колонкам

Составной индекс помогает при запросах с фильтрацией по нескольким полям:

```
CREATE INDEX idx_customer_date ON orders(customer_id, order_date);
```

```
SELECT *
FROM orders
WHERE customer_id = 42 AND order_date BETWEEN '2024-01-01' AND '2024-12-31';
```

Важно! в этом случае индекс будет эффективен только тогда, когда используются обе колонки в фильтре или только первая.

Пример 4. Частичный индекс

Если необходимо ускорить выборку по часто встречающемуся значению, например, только для активных записей, создаётся частичный индекс:

```
CREATE INDEX idx_active_products ON products(price) WHERE is_active = true;
SELECT *
FROM products
WHERE is_active = true AND price > 1000;
```

Задания для выполнения лабораторной работы

1. Изучите медотический материал.
2. Оптимизируйте запрос с помощью индекса для созданной базы данных, согласно номеру варианта.

Лабораторная работа № 20

Итоговый комплексный проект: разработка полноценной базы данных и набора запросов по индивидуальному заданию.

Цель лабораторной работы: реализовать бизнес-логику модуля: добавление/удаление, пересчет итогов, реактивное обновление интерфейса.

Задания для выполнения лабораторной работы

1. На основе пройденных лабораторных занятий разработать реляционную базу данных в СУБД Microsoft SQL Server.
2. Сформулировать и создать запросы к базе данных на языке SQL.
3. Разработать программное приложение для управления базой данных на объектно-ориентированном языке программирования.

Темы запросов:

1. Выборка данных (запросы с использованием операторов сравнения, логических операторов AND, OR и NOT, использование выражений над столбцами, с проверкой на принадлежность множеству, с проверкой на принадлежность диапазону значений, с проверкой на соответствие шаблону, с проверкой на неопределенное значение).

2. Запросы на соединение данных (с использованием декартового произведения, соединения таблиц по равенству, соединения таблиц по равенству и условием отбора, внешнего полного соединения таблиц, левого внешнего соединения, правого внешнего соединения, симметричного соединения и удаление избыточности данных).

3. Группировка и сортировка данных (запросы с использованием COUNT, SUM, AVG, временных функций, группировки по одному столбцу, группировки по нескольким столбцам, условия отбора групп HAVING, с использованием фразы HAVING без фразы GROUP BY, сортировки).

4. Подзапросы (подзапросы с использованием операторов сравнения, с использованием оператора DISTINCT, с использованием агрегатных функций, оператора IN, выражений, группировки GROUP BY, группировки с условием HAVING, соотнесенных подзапросов).

5. Запросы на изменение данных (запросы на добавление новых данных, добавление новых данных по результатам запроса в качестве вставляемого значения, на обновление существующих данных в таблице, на обновление существующих данных по результатам подзапроса во фразе WHERE, запрос на удаление существующих данных).

6. Представления (представление с использованием данных из базовых таблиц, запросы на соединение двух представлений, представление с использованием функций COUNT, SUM, с использованием группировки GROUP BY, с использованием условия отбора групп HAVING, на обновление данных с помощью представлений, с использованием результатов подзапроса во фразе WHERE, на обновление существующих данных по результатам подзапроса во фразе WHERE, запрос на удаление представления).

7. Транзакции (запрос для создания транзакции, с помощью которой

выполнить вставку одной строки в таблицу; с помощью которой выполнить вставку одной строки во все таблицы БД; с помощью которой добавить 2 строки в таблицу и удалить 2 записи из таблицы; с помощью которой выполнить обновление записей в трех таблицах, в случае ошибки после обновления (при срабатывании проверки на ограничение), транзакция откатывалась; с помощью которой выполнить удаление записей в двух таблицах и если удаление прошло не успешно, то откатить транзакцию; с помощью которой выполнить вставку в две таблицы произвольных значений; с помощью которой обновить данные в трех таблицах БД, использовать транзакции и блок TRY CATCH; с помощью которой выполнить вставку одной строки во все таблицы, использовать блок TRY CATCH).

ОБЩАЯ ХАРАКТЕРИСТИКА САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

Самостоятельная работа - целенаправленная, планируемая в рамках учебного плана деятельность студентов, которая осуществляется по заданию, при методическом руководстве и контроле преподавателя, но без его непосредственного участия. Самостоятельная работа студентов является одной из важнейших составляющих образовательного процесса.

В учебном процессе учебного заведения выделяют два вида самостоятельной работы: аудиторная и внеаудиторная.

Аудиторная самостоятельная работа выполняется на учебных занятиях под непосредственным руководством преподавателя и по его заданию.

Внеаудиторная — планируемая учебная, учебно-исследовательская, научно-исследовательская работа студентов, выполняемая во внеаудиторное время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия.

Целью самостоятельной работы студентов является:

- систематизация и закрепление полученных теоретических знаний и практических умений;
- углубление и расширение теоретических знаний;
- формирование умений использовать нормативную, правовую, справочную документацию и специальную литературу;
- развитие познавательных способностей и активности студентов, творческой инициативы, самостоятельности, ответственности, организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, совершенствованию и самоорганизации;
- формирование общих и профессиональных компетенций.

Самостоятельная работа студентов должна быть хорошо спланирована и организована. При планировании такой работы необходимо учитывать условия, обеспечивающие её успешное выполнение:

- чёткое определение преподавателем объёма и содержания самостоятельной работы;
- определение видов консультативной помощи;
- постановка цели самостоятельной работы и критерии её оценки;
- виды и формы контроля её выполнения.

Выполняя самостоятельную работу под контролем преподавателя, студент должен:

- освоить минимум знаний;
- планировать свою самостоятельную работу в соответствии разработанным графиком;
- выполнять самостоятельную работу и отчитываться по ее результатам в соответствии с графиком представления результатов, видами и сроками отчетности по самостоятельной работе студентов.

В процессе самостоятельной работы студент приобретает навыки самоорганизации, самоконтроля, самоуправления, саморефлексии и становится

активным самостоятельным субъектом учебной деятельности.

Таким образом, самостоятельная работа студентов оказывает важное влияние на формирование личности будущего специалиста.

Самостоятельная работа студентов является обязательной для каждого студента, объем ее определяется учебным планом в соответствии с требованиями Государственных образовательных стандартов.

При изучении тем дисциплины студенты выполняют следующие виды самостоятельной работы:

- проработка конспектов занятий, учебных изданий и специальной технической литературы;
- составление конспекта, тематических схем, таблиц;
- подготовка к лабораторным работам и практическим занятиям с использованием методических рекомендаций преподавателя;
- оформление отчетов по лабораторным работам и практическим занятиям, подготовка к их защите;
- моделирование и решение производственных процессов и ситуационных задач;
- подготовка презентаций;
- работа с электронными ресурсами в сети Интернет;
- подготовка к семинару;
- подготовка к зачетам, экзаменам.

Технология организации самостоятельной работы студентов включает использование информационных и материально-технических ресурсов образовательного учреждения. Материально-техническое и информационно - техническое обеспечение самостоятельной работы студентов включает в себя:

- библиотеку с читальным залом, укомплектованную в соответствии с существующими нормами;
- учебно-методическую базу учебных кабинетов, лабораторий и методического центра;
- компьютерные классы с возможностью работы в Интернет;
- базы практики в соответствии с заключенными договорами;
- аудитории для консультационной деятельности;
- учебную и учебно-методическую литературу, разработанную с учетом увеличения доли самостоятельной работы студентов, и иные методические материалы.

Перед выполнением внеаудиторной самостоятельной работы преподаватель проводит инструктаж по выполнению задания, в котором указывает цель задания, его содержание, сроки выполнения, ориентировочный объем работы, основные требования к результатам работы, критерии оценки. Во время выполнения студентами внеаудиторной самостоятельной работы и при необходимости преподаватель может проводить консультации. Самостоятельная работа может осуществляться индивидуально или группами студентов в зависимости от цели, объема, конкретной тематики самостоятельной работы, уровня сложности, уровня умений обучающихся.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Общие методические рекомендации студенту при изучении тем дисциплины.

Большая часть самостоятельной работы выполняется студентом вне учебных занятий при подготовке домашних заданий. Общие требования к выполнению этого вида самостоятельной работы заключаются в следующем:

- активно работать на уроке, усваиваю основную часть нового материала;
- если что-то непонятно, не стесняться задавать вопросы преподавателю;
- большое задание необходимо разбивать на части и работать над каждой из них в отдельности;
- выполняя домашнее задание, надо не просто думать, что надо сделать, а еще и решать, с помощью каких средств и приемов этого можно добиться;
- в процессе приготовления домашнего задания необходимо делать перерывы;
- готовиться к докладам, рефератам, защите курсовых работ и проектов, практических и лабораторных занятий надо заранее, равномерно распределяя нагрузку, а не оставлять такую ответственную работу на последний день;
- изучая заданный материал, сначала надо его понять, а уже потом запомнить;
- научиться находить интересующую нужную информацию с помощью компьютера;
- не стесняться обращаться за помощью к взрослым и однокурсникам;
- надо составлять план устного ответа и проверять себя;
- на письменном столе должно лежать только то, что необходимо для выполнения одного задания. После его завершения со стола убираются уже использованные материалы, и кладутся те учебные принадлежности, которые необходимы для выполнения следующего задания;
- нужно решить, в какой последовательности лучше выполнять задания и сколько времени понадобится на каждое из них;
- трудный материал урока лучше повторить в тот же день, чтобы сразу закрепить его и запомнить;
- читая учебник, надо задавать самому себе вопросы по тексту.

Подготовка тематических сообщений, докладов, рефератов

Реферат доклад, сообщение (от латинского refero - передаю, сообщаю) - краткое письменное изложение материала по определенной теме с целью привития студентам навыков самостоятельного поиска и анализа информации, формирования умения подбора и изучения литературных источников, используя при этом дополнительную научную, методическую и периодическую литературу.

Тема реферата выбирается по желанию студента из списка, предлагаемого преподавателем. Тема может быть сформулирована студентом самостоятельно.

Выбранная тема согласовывается с преподавателем.

После выбора темы требуется:

- составить план реферата;

- подобрать необходимую информацию;
- изучить подобранную информацию;
- составить текст реферата.

План реферата должен включать в себя введение, основной текст и заключение. Во введении аргументируется актуальность выбранной темы, указываются цели и задачи исследования. В нем также отражается методика исследования и структура работы. Основная часть работы предполагает освещение материала в соответствии с планом. В заключении излагаются основные выводы и рекомендации по теме исследования.

Реферат оформляется согласно требованиям, установленным в учебном заведении. Он должен содержать: титульный лист, оглавление и список использованной литературы. На титульном листе указываются: название учебного заведения, название профессионального модуля, междисциплинарного курса, тема работы, курс, группа, фамилии, имена, отчества студента и руководителя работы, название города, в котором находится учебное заведение, год написания данной работы. Реферат может содержать приложения в форме схем, образцов документов и другие изображения в соответствии с темой исследования. Все страницы работы, включая оглавление и список литературы, нумеруются по порядку с титульного листа (на нем цифра не ставится) до последней страницы без пропусков и повторений. Введение, заключение, новые главы, список использованных источников и литературы должны начинаться с нового листа. Подбор литературы производится студентом из предложенного преподавателем списка литературы. Текст реферата необходимо набирать на компьютере на одной стороне листа. Размер левого поля 30 мм, правого - 15 мм, верхнего - 20 мм, нижнего - 20 мм. Шрифт - Times New Roman, размер - 14, межстрочный интервал - 1,5. Фразы, начинающиеся с новой строки, печатаются с абзацным отступом от начала строки (1,25 см). Реферат, выполненный небрежно, неразборчиво, без соблюдения требований по оформлению, возвращается студенту без проверки с указанием причин возврата на титульном листе.

Критерии оценки:

- знание и понимание проблемы;
- умение систематизировать и анализировать материал, четко и обоснованно формулировать выводы;
- «трудозатратность» (объем изученной литературы, добросовестное отношение к анализу проблемы);
- самостоятельность, способность к определению собственной позиции по проблеме и к практической адаптации материала, недопустимость плагиата;
- выполнение необходимых формальностей (точность в цитировании и указании источника текстового фрагмента, аккуратность оформления).

Проработка занятый, учебных изданий и специальной технической литературы

Работа с конспектом лекций по темам междисциплинарных курсов заключается в том, что студент после рассмотрения темы на учебных занятиях в период между очередными лекциями изучает материал конспекта. При этом непонятные положения конспекта необходимо выяснять у преподавателя на

консультациях или при чтении основной и дополнительной литературы.

При работе с книгой необходимо научиться правильно ее читать, вести записи. Для подбора литературы в библиотеке используются алфавитный и систематический каталоги. Правильный подбор учебников рекомендуется преподавателем. Необходимая литература может быть также указана в методических разработках. Изучая материал по учебнику, следует переходить к следующему вопросу только после правильного уяснения предыдущего, описывая на бумаге все выкладки и определения (в том числе те, которые в учебнике опущены или на лекции даны для самостоятельного вывода). Полезно составлять опорные конспекты. При изучении материала по учебнику, полезно в тетради (на специально отведенных полях) дополнять конспект лекций, написанный на учебных занятиях. Там же следует отмечать вопросы, выделенные студентом для консультации с преподавателем. Выводы, полученные в результате изучения, рекомендуется в конспекте выделять, чтобы они при пропитывании записей лучше запоминались. Различают два вида чтения; первичное и вторичное. Первичное - эти внимательное, неторопливое чтение, при котором можно остановиться на трудных местах. После него не должно остаться ни одного непонятного слова. Содержание не всегда может быть понятно после первичного чтения. Задача вторичного чтения - полное усвоение смысла целого (по счету это чтение может быть и не вторым, а третьим или четвертым).

Чтение научного текста является частью познавательной деятельности. Ее цель - извлечение из текста необходимой информации. От того на сколько осознанна читающим собственная внутренняя установка при обращении к печатному слову (найти нужные сведения, усвоить информацию полностью или частично, критически проанализировать материал и т.п.) во многом зависит эффективность осуществляемого действия. Выделяют четыре основные установки в чтении научного текста:

- информационно-поисковая, задача которой - найти, выделить искомую информацию;
- усваивающая, при которой усилия читателя направлены на то, чтобы как можно полнее осознать и запомнить как сами сведения, излагаемые автором, так и всю логику его рассуждений;
- аналитико-критическая - читатель стремится критически осмыслить материал, проанализировав его, определив свое отношение к нему;
- творческая, создающая у читателя готовность в том или ином виде использовать суждения автора, ход его мыслей, результат наблюдения, разработанную методику, дополнить их, подвергнуть новой проверке.

Самостоятельная работа при чтении учебной литературы начинается с изучения конспекта материала, полученного при слушании лекций преподавателя. Полученную информацию необходимо осмыслить. При необходимости, в конспект лекций могут быть внесены схемы, эскизы рисунков, другая дополнительная информация.

Составление конспекта, тематических схем, таблиц

При изучении нового материала, как правило, составляется конспект. Конспект - изложение текста, которому присущи краткость, связность и

последовательность. При этом максимально точно записываются формулы, определения, схемы, трудные для запоминания места.

При оформлении конспекта необходимо стремиться к емкости каждого предложения. Мысли автора книги следует излагать кратко, заботясь о стиле и выразительности написанного. Число дополнительных элементов конспекта должно быть логически обоснованным, записи должны распределяться в определенной последовательности, отвечающей логической структуре текста. Для уточнения и дополнения необходимо оставлять поля. Овладение навыками конспектирования требует от студента целеустремленности, повседневной самостоятельной работы.

Классификация конспектов:

- плановый конспект, для чего сначала нужно написать план текста, а затем на пункты плана делаются комментарии: свободно изложенный текст либо цитаты;
- обзорный конспект - краткое изложение данной темы с использованием нескольких источников;
- текстуальный конспект состоит из цитат одного текста;
- свободный конспект предполагает цитаты текста и собственные формулировки прочитанного текста;
- сложный - конспект, в котором отражается определенная тема или вопрос;
- хронологический конспект отражает последовательность событий;
- опорный конспект, в котором излагается информация в виде опорных знаков, слов, сигналов.

Методические рекомендации по составлению конспекта:

- определить цель написания конспекта;
- внимательно прочитать текст, уточнить в справочной литературе непонятные слова;
- выделить основные смысловые части текста;
- определить главное, составить план;
- кратко сформулировать основные положения текста, отметить аргументацию автора;
- составить текст конспекта, изложив информацию кратко и своими словами, четко следуя пунктам плана, записи следует вести четко, ясно;
- грамотно записывать цитаты, учитывая лаконичность, значимость мысли;
- в тексте конспекта желательно приводить не только тезисные положения, но и их доказательства.

При составлении тематических схем, таблиц необходимо внимательно прочитать текст соответствующий параграф учебника. Продумать «конструкцию» таблицы или схемы, расположение порядковых номеров, терминов, примеров и пояснений (и прочего). Начертить схему или таблицу и заполнить ее графы необходимым содержимым.

Подготовка к лабораторным работам и практическим занятиям, оформление отчетов по лабораторным работам и практическим занятиям, подготовка к их

защите

Программы профессиональных модулей предусматривают выполнение практических и лабораторных занятий.

Лабораторное занятие - форма учебного занятия, ведущей дидактической целью которого является экспериментальное подтверждение и проверка существующих теоретических положений (законов, зависимостей), формирование учебных и профессиональных практических умений и навыков.

Практическое занятие - это одна из форм учебной работы, которая ориентирована на закрепление изученного теоретического материала, его более глубокое усвоение и формирование умения применять теоретические знания в практических целях. Особое внимание на практических занятиях уделяется выработке учебных или профессиональных навыков. Такие навыки формируются в процессе выполнения конкретных заданий - упражнений, задач - под руководством и контролем преподавателя.

Подготовка к практическим и лабораторным занятиям заключается в работе с конспектом лекций по данной теме, в изучении соответствующего раздела учебника или учебного пособия, в просмотре дополнительной литературы. Этапы подготовки к практическому или лабораторному занятию заключаются в следующем: освежить в памяти теоретические сведения, полученные на лекциях и в процессе самостоятельной работы, подобрать необходимую учебную и справочную литературу. Отобрать те материалы, которые позволят в полной мере реализовать цели и задачи предстоящей работы. Еще раз проверить соответствие отобранного материала. Студент должен прийти на лабораторное или практическое занятие подготовленным по данной теме.

При выполнении заданий практического или лабораторного занятия студент должен быть ознакомлен преподавателем с целью и ходом выполнения задания и, по необходимости, с правилами техники безопасности. Если у студентов во время выполнения заданий возникают вопросы, то преподаватель консультирует студентов. Порядок выполнения того или иного задания излагается в инструкционных картах или рабочих тетрадях.

После проведения занятия студент представляет письменный отчет, который оформляется в соответствие с принятыми в образовательном учреждении правилами. Отчеты оформляются на листах писчей бумаги формата А4 или в специальных рабочих тетрадях, разработанных преподавателем. Содержание отчета указано в инструкционных картах или рабочих тетрадях.

При подготовке к защите практических и лабораторных занятий студент должен ответить на контрольные вопросы, указанные также в инструкционных картах или рабочих тетрадях, проштудировав при этом конспект лекций, учебную литературу.

Моделирование и решение производственных процессов и ситуационных задач

При изучении дисциплины очень часто студенту приходится сталкиваться с профессиональными задачами и ситуациями, которые необходимо решить самостоятельно, как во время аудиторной работы, так и во время внеаудиторной. При решении таких задач необходимо:

- провести анализ ситуации для определения проблемы в целом; представить ситуацию и себя в качестве действующего в ней лица; проанализировать ошибочные или правильные действия всех участников ситуации;
- определить проблемные узлы - возможные причины и прогнозируемые последствия развития данной ситуации;
- рассмотреть условное прогнозирование развития ситуации: определить окончательную гипотезу, представить обоснованный и доказательный прогноз вероятностного развития ситуации; предложить варианты действий, обоснованные теоретически и, по возможности, подкрепленные практическим личным опытом, опираясь на принципы профессиональной этики; определить способы и методы воздействия на предлагаемую ситуацию;
- сформулировать итоговые выводы, используя профессиональные термины, доказательства правильности своего решения.

Подготовка презентаций

Подготовка презентации позволит студенту логически выстроить изучаемый материал, систематизировать его, сформировать коммуникативные компетенции. Материал презентации представляется в виде текста, схем, диаграмм, таблиц, которые призваны дополнить текстовую информацию или передать ее в более наглядном виде. Желательно избегать в презентации изображений, не несущих смысловой нагрузки, если они не являются частью стилевого оформления. Цвет графических изображений не должен резко контрастировать с общим стилевым оформлением слайдов, иллюстрации рекомендуется сопровождать пояснительным текстом.

Анимационные эффекты используются для привлечения внимания слушателей или для демонстрации динамики развития какого - либо процесса. В этих случаях использование анимации оправдано, но не стоит чрезмерно насыщать презентацию такими эффектами, иначе это вызовет негативную реакцию аудитории.

Звуковое сопровождение должно отражать суть или подчеркивать особенность темы слайда, презентации. Фоновая музыка не должна отвлекать внимание слушателей и заглушать слова докладчика.

Оптимальное количество слайдов, как правило, десять - пятнадцать. Для оформления слайдов презентации рекомендуется использовать несложные шаблоны, соблюдать единый стиль. Не рекомендуется на одном слайде использовать более трех цветов. Смену слайдов для управления презентацией докладчиком желательно устанавливать по щелчку без времени. Шрифт, выбираемый для презентации, должен обеспечивать читаемость информации на экране и соответствовать выбранному шаблону оформления. Не желательно использовать разные шрифты в одной презентации.

Алгоритм выстраивания презентации должен соответствовать логической структуре работы и отражать последовательность ее этапов. Независимо от алгоритма выстраивания презентации на первом слайде рекомендуется выносить следующие данные: полное наименование образовательной организации; тема презентации; фамилия, имя, отчество студента; специальность обучения; фамилия,

имя, отчество руководителя. Последний слайд должен содержать фразу «Спасибо за внимание».

Работа с электронными ресурсами в сети Интернет

Для повышения эффективности самостоятельной работы студент должен учиться работать в поисковой системе сети Интернет, в электронно-библиотечной системе и использовать найденную информацию при подготовке к занятиям.

Интернет сегодня - правомерный источник научных статей, статистической и аналитической информации, и использование его наряду с книгами давно уже стало нормой. Однако, несмотря на то, что ресурсы Интернета позволяют достаточно быстро и эффективно осуществлять поиск необходимой информации, следует помнить о том, что эта информация может быть неточной или вовсе не соответствовать действительности. В связи с этим при поиске материала по заданной тематике следует обращать внимание на научные труды признанных авторов, которые посоветовали вам преподаватели.

Поиск информации можно вести по автору, заглавию, виду издания, году издания или издательству. Также в сети Интернет доступна услуга по скачиванию методических указаний и учебных пособий, подбору необходимой учебной и научно - технической литературы.

Подготовка к семинару

Семинар — это особая форма учебно-теоретических занятий, которая, как правило, служит дополнением к лекционному курсу. Семинар обычно посвящен детальному изучению отдельной темы.

Этапы подготовки к семинару:

- проанализировать тему семинара, подумать о цели и основных проблемах, вынесенных на обсуждение;
- внимательно прочитать материал, данный преподавателем по этой теме на лекции;
- изучить рекомендованную литературу, делая при этом конспекты прочитанного или выписки, которые понадобятся при обсуждении на семинаре;
- постараться сформулировать свое мнение по каждому вопросу и аргументированно его обосновать;
- записать возникшие во время самостоятельной работы с учебниками и научной литературой вопросы, чтобы затем на семинаре получить на них ответы.

При подготовке к семинарским занятиям следует руководствоваться указаниями и рекомендациями преподавателя, использовать основную и дополнительную литературу из представленного им списка.

При подготовке доклада на семинарское занятие желательно заранее обсудить с преподавателем перечень используемой литературы, за день до семинарского занятия предупредить его о необходимых для представления материала технических средствах. Напечатанный текст доклада представить преподавателю на рецензию.

Подготовка к зачетам, экзаменам

Изучение выше перечисленных тем дисциплины завершается зачетами или

экзаменами.

Подготовка к зачету или экзамену способствует закреплению, углублению и обобщению знаний, получаемых в процессе обучения, а также применению их к решению практических задач. Готовясь к зачету или экзамену, студент ликвидирует имеющиеся пробелы в знаниях, углубляет, систематизирует и упорядочивает свои знания. На зачете или экзамене студент демонстрирует то, что он приобрел в процессе обучения конкретным темам междисциплинарных курсов или модулям в целом.

Экзаменационная сессия - это серия экзаменов, установленных учебным планом. Между экзаменами, согласно графику их проведения, дается интервал времени в несколько дней. Не следует думать, что их достаточно для успешной подготовки к экзаменам. В эти дни нужно систематизировать уже имеющиеся знания. На консультации перед экзаменом студентов познакомят с основными требованиями, ответят на возникшие у них вопросы. Поэтому посещение консультаций обязательно.

Требования к организации подготовки студента к экзаменам те же, что и при занятиях в течение семестра, но соблюдаться они должны более строго. Во-первых, очень важно соблюдение режима дня: сон не менее 8 часов в сутки, занятия должны заканчиваться не позднее, чем за 2-3 часа до сна.

Оптимальное время занятий - утренние и дневные часы. В перерывах между занятиями рекомендуются прогулки на свежем воздухе, неутомительные занятия спортом. Во-вторых, наличие хороших собственных конспектов лекций. Даже в том случае, если была пропущена какая- либо лекция, необходимо во время ее восстановить, обдумать, снять возникшие вопросы для того, чтобы запоминание материала было осознанным. В-третьих, при подготовке к зачету или экзамену у студента должен быть хороший учебник или конспект литературы, прочитанной по указанию преподавателя в течение семестра. Здесь можно эффективно использовать листы опорных конспектов. Вначале следует просмотреть весь материал по сдаваемой теме, отметить для себя трудные вопросы, обязательно в них разобраться. В заключение еще раз целесообразно повторить основные положения. Систематическая подготовка к занятиям в течение семестра позволит использовать время экзаменационной сессии для систематизации знаний.

Правила подготовки к экзамену:

- сориентироваться во всем материале и обязательно расположить его согласно экзаменационным вопросам или вопросам, обсуждаемым на семинарах, учебных занятиях. Эта работа может занять много времени, но все остальное - уже технические детали, главное - это ориентировка в материале;
- постараться максимально запомнить материал, переосмыслить его, рассмотреть альтернативные идеи;
- подготовить «шпаргалки», главный смысл которых систематизация и оптимизация знаний, однако пользоваться таким подспорьем не рекомендуется. Это очень сложная и важная для студента работа, более сложная и важная, чем простое поглощение массы учебной информации. Если студент самостоятельно подготовил такие «шпаргалки», то, скорее всего, он и экзамены сдавать будет более уверенно, так как у него уже сформирована общая ориентировка в сложном материале. Как это ни парадоксально, но использование «шпаргалок» часто

позволяет отвечающему студенту лучше демонстрировать свои познания, точнее - ориентировку в знаниях, что намного важнее знания «запомненного» и «тут же забытого» после сдачи экзамена.

При ответе на экзамене студент сначала должен продемонстрировать преподавателю усвоенный по программе обучения материал, и лишь после этого высказать иную, желательно аргументированную точку зрения.

МЕТОДИКА ВЫПОЛНЕНИЯ ВНЕАУДИТОРНОЙ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Получить у преподавателя задание и необходимую литературу.
2. Найти предложенную литературу на образовательном портале или в библиотеке.
3. Изучить имеющуюся литературу в электронном или печатном виде, прочитать материалы лекций, практических и (или) семинарских занятий по теме.
4. Изучить методические рекомендации.
5. Оформить работу в тетради или на компьютере в соответствии с требованиями преподавателя.
6. Сдать самостоятельную работу преподавателю, предварительно ответив на вопросы для самоконтроля.

МЕТОДЫ КОНТРОЛЯ И ОЦЕНКА ВНЕАУДИТОРНОЙ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Контроль результатов самостоятельной работы проводится преподавателем одновременно с текущим и промежуточным контролем знаний обучающихся. Для контроля самостоятельной работы обучающегося используются разнообразные формы и методы: фронтальный, индивидуальный, выборочный, самоконтроль, защита презентации, участие в семинарском занятии, ответы на контрольные вопросы и т. д. При контроле результатов самостоятельной работы используются следующие критерии:

- уровень освоения обучающимся учебного материала;
- умение обучающегося использовать теоретические знания при выполнении заданий;
- обоснованность и чёткость изложения ответа;
- оформления материала в соответствии с требованиями.

Критерии оценки выполненной обучающимися работы:

оценка «5» - работа выполнена без ошибок; чисто, без исправлений; тема раскрыта полностью;

оценка «4» - работа выполнена с незначительными ошибками; тема раскрыта не полностью;

оценка «3» - работа выполнена со значительными ошибками; тема практически не раскрыта;

оценка «2» - работа не выполнена.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Нестеров, С. А. Базы данных: учебник и практикум для вузов / С. А. Нестеров. — Москва : Издательство Юрайт, 2021. — 230 с. — (Высшее образование). — ISBN 978-5-534-00874-6.

2. Гордеев, С. И. Организация баз данных в 2 ч. Часть 1 : учебник для вузов / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2021. — 310 с. — (Высшее образование). — ISBN 978-5-534-04469-0.

3. Гордеев, С. И. Организация баз данных в 2 ч. Часть 2 : учебник для вузов / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2021. — 513 с. — (Высшее образование). — ISBN 978-5-534-04470-6.

4. Кариев, Ч. А. Технология Microsoft ADO .NET : учебное пособие / Ч. А. Кариев. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 665 с. — ISBN 978-5-4497-0945-5.

5. Стасышин, В. М. Базы данных: технологии доступа : учебное пособие для вузов / В. М. Стасышин, Т. Л. Стасышина. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2020. — 164 с. — (Высшее образование). — ISBN 978-5-534-08687-4.

7. Маркин, А. В. Программирование на SQL в 2 ч. Часть 1 : учебник и практикум для вузов / А. В. Маркин. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2021. — 403 с. — (Высшее образование). — ISBN 978-5-534-12256-5.

8. Маркин, А. В. Программирование на SQL в 2 ч. Часть 2 : учебник и практикум для вузов / А. В. Маркин. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2021. — 340 с. — (Высшее образование). — ISBN 978-5-534-12258-9.

9. Базы данных : учеб.-метод. пособие по выполнению лаб. работ. / О. В. Игнатьева ; ФГБОУ ВО РГУПС. - Ростов н/Д : [б. и.], 2017. - 253 с.

Учебное издание

ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Ростовский государственный университет путей сообщения»
(ФГБОУ ВО РГУПС)

Адрес университета:
344038, г. Ростов н/Д, пл. Ростовского Стрелкового Полка
Народного Ополчения, д. 2, www.rgups.ru