Теоретичні відомості

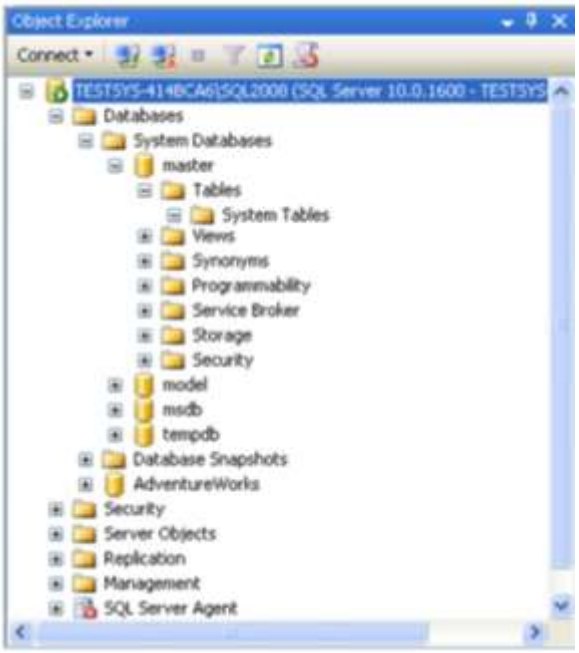
Компоненты Microsoft SQL Server 2008

Редактор запросов (Query Editor)

- Для того чтобы написать новый запрос к базе данных, необходимо выполнить команду New Query, расположенную на панели инструментов *Management Studio*. В результате откроется новая вкладка, в которой можно писать SQL-код.
- Для выполнения запроса необходимо выполнить команду **Query – Execute (F5)**. Чтобы просто проверить правильность синтаксической записи можно воспользоваться командой **Query – Parse (Ctrl+F5)**, при этом сам запрос не будет выполнен

Object Explorer

Позволяет осуществлять навигацию по базе данных: просматривать доступные объекты, выполнять запросы на просмотр содержимого таблиц, создавать скрипты для объектов и т.д.



База данных, выбранная в этом списке, используется в редакторе запросов как база данных по умолчанию. Поэтому важно перед выполнением запросов, убедиться, что выбрана нужная БД.

Это можно сделать либо через выпадающий список на панели инструментов:

либо при помощи команды SQL: USE <Ім'я БД>

Структура языка Transact SQL

SQL (Structured Query Language – Структурированный Язык Запросов) – стандартный язык запросов по работе с реляционными базами данных

1. Комментарии

Форма записи:

- /*Текст комментария*/ – обычно используется для записи многострочных комментариев
- --Текст комментария – используется для комментариев, записываемых в одну строку

2. BNF-нотация (Форма Бэкуса-Наура)

Условные обозначения при описании синтаксиса команд SQL:

- Ключевые слова записываются прописными буквами (необязательно). Они зарезервированы и составляют часть команды
- Вертикальная черта **|** указывает на необходимость выбора одного из нескольких значений (например: a **| b | c**)
- Фигурные скобки **{ }** определяют обязательный элемент (например **{a}**)
- Квадратные скобки **[ ]** определяют НЕобязательный элемент (например **[a]**)
- Круглые скобки **()** являются элементом команды
- Троеточие "…" означает, что предшествующая часть оператора может быть повторена любое количество раз
- Многоточие **",.."** указывает, что предшествующая часть команды, состоящая из нескольких элементов, разделенных запятыми, может иметь произвольное число повторений. Запятую нельзя ставить после последнего элемента
- Метки-заполнители **<>** конкретных значений элементов и переменных записываются курсивом

Оператор – это символ, обозначающий действие, выполняемое над одним или несколькими выражениями

Приоритет операторов

Если в выражении присутствует несколько операторов, то порядок их выполнения определяется приоритетом операторов (от самого высокого к самому низкому):

- () – выражения в скобках
- *, /, % – арифметические операторы типа умножения
- +, - – арифметические операторы типа сложения
- =, >, <, >=, <=, <> – операторы сравнения
- ^ (побитное исключающее ИЛИ), & (побитное И), | (побитное ИЛИ).
- NOT
- AND
- ALL, ANY, BETWEEN, IN, LIKE, OR, SOME
- = – присваивание значения переменной

Если операторы имеют одинаковый приоритет, вычисления производятся слева направо

Для изменения порядка выполнения операторов используются скобки. Выражения в скобках вычисляются первыми

Создание файла данных стандартными командами языка T-SQL

В Microsoft SQL Server БД состоит из двух частей:

- *Файл данных* – файл, имеющий расширение mdf и где находятся все таблицы и запросы;
- *Файл журнала транзакций* - файл, имеющий расширение ldf, содержит журнал, где фиксируются все действия с БД. Данный файл предназначен для восстановления БД в случае её выхода из строя.

Для создания нового файла данных используется команда CREATE DATABASE, которая имеет следующий синтаксис:

```
CREATE DATABASE <Имя БД>
ON (    Name=<Логическое имя>,
        FileName=<Имя файла>,
        [Size=<Нач.размер>,]
        [Maxsize=<Макс.размер>,]
        [FileGrowth=<Шаг>]
        )
[LOG ON (
        Name=<Логическое имя>,
        FileName=<Имя файла>
        [Size=<Нач.размер>,]
        [Maxsize=<Макс.размер >,]
        [FileGrowth=<Шаг>]
            ) ]
```

Здесь:

- Имя БД – имя создаваемой БД
- Логическое имя – определяет логическое имя файла данных БД, по которому происходит обращение к файлу данных.
- Имя файла – определяет полный путь к файлу данных.
- Нач.размер – начальный размер файла данных в Мб.
- Макс.размер – максимальный размер файла данных в Мб.
- Шаг – шаг увеличения файла данных, либо в Мб либо в %.
- Параметры в разделе LOG ON аналогичны параметрам в разделе CREATE DATABASE. Однако они определяют параметры журнала транзакций.

Пример:

Создадим БД «Students», расположенную в файле «D:\Students.mdf» и имеющую начальный размер файла данных 5мб., максимальный размер файла данных 100мб. и шаг увеличения файла данных равный 1мб. Файл журнала транзакций данной БД имеет имя «StudentsLog» и расположен в файле «D:\Students.ldf». Данный файл имеет начальный размер равный 3мб., максимальный размер равный 20мб. и шаг увеличения равный 10%.

```
CREATE DATABASE Students
on (Name=Students,
FileName='D:\Students.mdf',
Size=5Mb,
Maxsize=100Mb,
FileGrowth=1Mb)
log on (Name=StudentsLog,
FileName='D:\Students.ldf',
Size=3Mb,
Maxsize=20Mb,
FileGrowth=10%)
```

Управление базами данных при помощи команд языка T-SQL

В языке запросов T-SQL с БД возможны следующие действия:

- Відображення відомостей про БД:

EXEC sp_helpdb <Имя БД>;

- Зміна імені БД, додавання нових файлів, видалення файлів, зміна параметрів файлів, що входять в БД. Команди відповідно:

ALTER DATABASE <Имя БД>

ADD FILE (<Параметры>)| REMOVE FILE <Логическое имя файла>| MODIFY FILE (<Параметры>)

- Переименование БД:

EXEC SP_RENAMEDB < Ім'я БД>,<Нове ім’я БД>;

- Видалення БД:

DROP DATABASE < Ім'я БД>

Вышеперечисленные команды используют следующие параметры:

- <Имя БД> - имя БД с которой производится действие;
- <Параметр> - изменяемый параметр;
- <Значение> - новое значение изменяемого параметра;
- <Параметры> - параметры файла БД, аналогичные параметрам, используемым в команде CREATE DATABASE;
- <Логическое имя файла> - логическое имя файла, входящего в БД;
- <Новое имя БД> - новое имя БД.

Теоретичні відомості

Типы данных

Целые числа:

- **Bit** (1 байт)
  - один байт
- **Bigint** (8 байт)
  - диапазон [-2^{63} .. $2^{63}-1$]
- **Int** (4 байта)
  - диапазон [–2 147 483 648 .. 2 147 483 647]

Числа с фиксированной запятой:

- **Decimal** или **Numeric**
  - диапазон [$-10^{38}-1$ .. $10^{38}-1$]
  - При описании столбца с этим типом данных задается точность и масштаб.
  - Точность – общее число знаков, которое можно хранить в поле.
  - Масштаб – количество десятичных знаков, которое можно хранить справа от десятичной точки.
- **Money** (8 байт) – денежный формат
  - диапазон [-2^{63} .. 2^{63}]
  - Всегда содержат 4 цифры справа от десятичной точки
- **SmallMoney** (4 байта) – денежный формат
  - диапазон [–214748,3648 .. +214748,3647]

Числа с плавающей запятой:

- **Float**
  - диапазон [–1,79E + 308 .. 1,79E + 308]
  - **Real**
  - диапазон [–3,4E + 38 .. 3,4E + 38]

Дата и время:

- **DateTime** (8 байт)
  - диапазон [01.01.1753 г. .. 31.12.9999 г.]
  - точность до трех сотых секунды
- **SmallDateTime** (4 байта)
  - диапазон [01.01.1900 г. .. 6.06.2079]
  - точность одна минута
- **Date** (3 байта)
  - хранит только дату
  - диапазон [01.01.0001 г. .. 31.12.9999 г.]
- **Time** (3-5 байт)
  - хранит только время
  - точность до 0,1 миллисекунды

Символьные строки:

- **Char** – строка фиксированной длины
  - содержит символы не относящиеся к таблице Unicode
  - Максимальная длина строки 8000 символов
- **VarChar** – строка переменной длины

- Максимальная длина строки 8000, но при использовании ключевого слова ‘‘max’’ может хранить до 2^{31} байт

- **Text** – применялся для хранения больших строк, сейчас рекомендуется использование **varchar(max)**
  - Оставлен для обратной совместимости.
- **Nchar –** строка фиксированной длины в Юникоде
  - максимальная длина строки 4000 символов
- **NvarChar –** строка переменной длины в Юникоде
  - Максимальная длина строки 4000 символов, но при использовании ключевого слова ‘‘max’’ может хранить до 2^{31} байт.
- **Ntext** – аналогичен типу **text**, но предназначен для работы с Юникод

Двоичные данные:

- **Binary –** позволяет хранить двоичные данные размером до 8000 байт
- **VarBinary –** тип данных переменной длины, позволяет хранить до 8000 байт, но при использовании ключевого слова ‘‘max’’ до 2^{31} байт
- **Image –** использовался для хранения больших объемов данных, сейчас рекомендуется использовать **varbinary(max)**

Оставлен для обратной совместимости

Прочие типы данных:

- **Table –** особый тип данных, используемый в основном для временного хранения таблиц и для передачи в качестве параметра в функции
- **HierarchyID –** используется для представления положения в иерархической структуре
- **Sql_variant** – **т**ип данных, хранящий значения различных типов данных, поддерживаемых MS SQL Server.
- **XML** – позволяет хранить XML-данные
- Прочие типы данных:
- **Cursor** (1 байт) – тип данных для переменных или выходных параметров хранимых процедур, которые содержат ссылку на курсор
- **Timestamp/ rowversion** (8 байт) – это тип данных, который представляет собой автоматически сформированные уникальные двоичные числа в БД
- Значение данного типа генерируется БД автоматически при вставке или изменении записи
- **UniqueIdentifier** (16 байт) – представляет собой GUID (Special Globally Unique Identifier)
- Гарантируется уникальность данного значения

Правила присвоения имен объектам базы данных

- Должны начинаться с буквы.
- Могут включать от 1 до 30 символов.
- Могут содержать только символы A-Z, a-z, 0-9, _ (подчеркивание), $ и #.

- Не могут совпадать с именем другого объекта, принадлежащего этому же пользователю.
- Не могут совпадать с зарезервированным словом сервера базы данных.

Создание таблицы стандартными командами языка T-SQL

Для создания таблиц в SQL Server в первую очередь необходимо сделать активной ту БД, в которой создается таблица. Для этого можно в новом запросе можно набрать команду:

USE <Имя БД>,

либо на панели инструментов необходимо выбрать в выпадающем списке рабочую БД. После выбора БД можно создавать таблицы.

Таблицы создаются командой:

```
CREATE TABLE <Имя таблицы>
(       <Имя поля1> <Тип1> [IDENTITY NULL|NOTNULL],
        <Имя поля2> <Тип2>, …
        CONSTRAINT <имя связи> PRIMARY KEY (<имя ключевого поля>),
        CONSTRAINT <имя связи> Foreign KEY (<имя ключевого поля>)
                References <имя родительской таблицы>
                (<ключевое поле родительской таблицы>)
        )
```

Здесь <Имя таблицы> – имя создаваемой таблицы;
<Имя поля> – имена полей таблицы;
<Тип> – типы полей;
<IDENTITY NULL|NOT NULL> – поле счётчик.

Замечание: Если имя поля содержит пробел, то оно заключается в квадратные скобки.

Если необходимо создать вычислимое поле, то в команде Create Table у вычислимого поля вместо типа данных нужно указать выражение.

Получение информации о таблице осуществляется применением команды:

EXEC SP_HELP <Имя таблицы>.

Удаление таблицы осуществляется командой:

DROP TABLE <Имя таблицы>.

```
ALTER TABLE Stud
ADD SumOcenka as (Ocenka1+Ocenka2)

ALTER TABLE Stud
DROP Column SumOcenka

ALTER TABLE Stud
ALTER Column Ocenka1 int NULL
```

Заповнення таблиць даними

В SQL Server 2008 заполнение таблиц производится при помощи следующей команды:

INSERT INTO <Имя таблицы> [(<Список полей>)]

VALUES (<Значения полей>)

где <Имя таблицы> – таблица, куда вводим данные,
(<Список полей>) – список полей, куда вводим данные, если не указываем, то подразумевается заполнение всех полей, в списке полей поля указываются через запятую,
<Значения полей>) – значение полей через запятую.

Примітка

При реалізації команди INSERT необхідно відстежувати , щоб:

- Послідовність даних в пропозицію VALUES, відповідала порядку стовпців в таблиці.
- Заповнювалися всі стовпці з ознакою NOT NULL.
- Незважаючи на те, що в команді INSERT список стовпців є необов'язковим, його рекомендується вказувати явно. Якщо ви все-таки хочете відмовитися від перерахування стовпців, то доведеться відстежувати, щоб порядок стовпців в таблиці відповідав порядку стовпців в команді INSERT.
- Спочатку заповнюються таблиці у яких немає зовнішніх ключів.
- Поле, яке є ідентифікатором в таблиці, заповнюється автоматично, оскільки має властивість IDENTITY.

Пример: Добавление записи имеющей следующие значения полей ФИО = Иванов, Адрес = Москва, Код специальности = 5 в таблицу «Студенты».

```
INSERT INTO Студенты (ФИО, Адрес, [Код специальности])
VALUES ('Иванов А.А.', 'Москва', 5)
```

Розділ 2. Формування запитів до таблиці БД
2.1 Синтаксис оператора SELECT

Оператор SELECT

Щоб створити запит необхідно зробити активною БД для якої створюється запит, потім в робочій області редактора запитів створити запит за допомогою команди SELECT

Синтаксис

SELECT [ALL | DISTINCT]
[TOP|PERCENT n]
[{*|[имя_столбца [[AS] псевдоним]] [,...n] }
[INTO <Имя_новой_таблицы>]
FROM имя_таблицы [[AS] псевдоним] [,...n]
[WHERE <условие_поиска>]
[GROUP BY имя_столбца [,...n]]
[HAVING <критерии выбора групп>]
[ORDER BY имя_столбца [,...n] [ASC|DESC]]
де

- Параметры ALL | DISTINCT показывают, какие записи обрабатываются:
- ALL обрабатывает все записи
- DISTINCT только уникальные, удаляются повторения записей
- TOP n определяет какое количество записей обрабатывают
- PERCENT n указывает процент от общего числа записей
- {* | [имя_столбца [[AS] псевдоним]] [,...n] } - указываются отображаемые поля из таблиц все или через запятую нужные
- INTO - Если присутствует этот раздел, то на основе результатов запроса создается новая таблица. Параметр INTO это имя новой таблицы
- FROM - Здесь указываются таблицы и запросы, через запятую, которые участвуют в новом запросе
- WHERE - Данный раздел используют для создания:
  - *простых запросов*, в этом случае в качестве условия указываем связываемые поля
  - *фильтров*, здесь указывают условия отбора
- GROUP BY – определяет поле для группировки записей в запросе
- HAVING - определяет условие поиска для группы или статистического выражения. Предложение можно использовать только в инструкции SELECT. Предложение HAVING обычно используется в предложении GROUP BY. Когда GROUP BY не используется, предложение HAVING работает так же, как и предложение WHERE
- ORDER BY – определяет поле для сортировки записей в запросе. Если указан параметр ASC, то будет производиться сортировка по возрастанию, если DESC – по убыванию. По умолчанию используется сортировка по возрастанию

Замечания:

- ☐ Здесь можно присваивать псевдонимы полям, следующим образом

<Имя поля> AS <Псевдоним> или <Имя поля> <Псевдоним>

- ☐ Если необходимо вывести все поля из таблицы –используется значк «*»

Выполнение запроса SELECT

1. Предложение **FROM** собирает данные из разных источников
2. Предложение **WHERE** ограничивает число строк на основании некоторого условия
--3. Предложение **GROUP BY** собирает подмножества данных
4. Вычисляются итоговые функции **(SELECT)**
---5. Предложение **HAVING** фильтрует подмножества данных
6. Вычисляются все выражения **(SELECT)**
7. Предложение **ORDER BY** сортирует результат

Имена объектов в SQL Server при формировании простых запросов

Название Таблицы

- zkVnz.dbo.Employe - имяБД. Dbo.ИмяТаблицы
- Employe - Имя Таблицы
- Учитывается регистр в названии Таблицы

Название Поля Таблицы

- Employe. EmployeeId - ИмяТаблицы.ИмяСтолбца
- EmployeeId – ИмяСтолбца
- Учитывается регистр в названии Поля Таблицы

2.1. Формування простого запиту

Приклад 1

Задача.

Вивести всю інформацію про студентів

Рішення 1.

```
SELECT *
FROM zkVnz.dbo.Student
```

Рішення 2.

```
SELECT *
FROM Student
```

Приклад 2

Задача.

Вывести только поля ФИО, должность Сотрудника (повторы)

Рішення.

```
SELECT Surname
    ,Name
    ,Patronymic
    ,Duties
 FROM zkVnz.dbo.Employee
```

Приклад 3

Задача.

Вывести без повторов ФИО, должность Сотрудника (повторов нет)

Рішення.

```
SELECT DISTINCT
      Surname
     ,Name
     ,Patronymic
     ,Duties
  FROM zkVnz.dbo.Employee
```

2.2 Простые запросы с использованием фразы WHERE

WHERE - (где) строки из указанной таблицы должны удовлетворять указанному перечню условий отбора

WHERE не допускает использования агрегирующих функций

Существует пять основных типов условий поиска (или предикатов):

- Сравнение: сравниваются результаты вычисления одного выражения с результатами вычисления другого
- Диапазон: проверяется, попадает ли результат вычисления выражения в заданный диапазон значений
- Принадлежность множеству: проверяется, принадлежит ли результат вычислений выражения заданному множеству значений
- Соответствие шаблону: проверяется, отвечает ли некоторое строковое значение заданному шаблону
- Значение NULL: проверяется, содержит ли данный столбец определитель NULL (неизвестное значение)

1. Операторы сравнения

>	больше	<	меньше
>=	больше или равно	<=	меньше или равно
!>	не больше	**!<**	не меньше
<> или !=	не равно	**=**	равно
AND	и	**NOT**	нет
OR	или		

Приклад 1

Задача.

Вивести інформацію про співробітників:

- з окладом більше 3000
- з окладом більше 3000 та надбавкою менше 200
- всіх окрім Ванжула

Рішення.

```
SELECT Surname,Name,Patronymic,Duties,Oklad
FROM zkVnz.dbo.Employee
  WHERE Oklad > 3000
--WHERE Oklad > 3000 AND Nadbavka < 220
--WHERE Surname<>'Ванжула‘
```

2. Диапазон : оператор BETWEEN ... AND ...

Используется для поиска значения внутри некоторого интервала, определяемого своими минимальным и максимальным значениями

(интервал от ... до ...)

При этом указанные значения включаются в условие поиска

Синтаксис

SELECT
[{* | [имя_столбца [[AS] псевдоним]] [,...n] }
FROM имя_таблицы [[AS] псевдоним] [,...n]
WHERE <имя столбца1> **BETWEEN** <Значение1> **AND** <Значение2>

Примітка

NOT BETWEEN (не принадлежит диапазону между)

Приклад 2

Задача.

Вивести інформацію про співробітників які:

- отримають оклади більш ніж 3000 та менш ніж 5000, т.б. знаходяться в даному діапазону;
- народилися в проміжок часу 1960-01-01 та 1980-12-31;
- отримають оклади менш ніж 3000 та більш ніж 5000, т.б. не знаходяться в в діапазоні;

Рішення.

```
SELECT * FROM zkVnz.dbo.Employee
WHERE Oklad BETWEEN 3000 AND 5000
-- WHERE Birthday BETWEEN '1960-01-01' AND '1980-12-31'
--WHERE Oklad NOT BETWEEN 3000 AND 5000
```

Или (что эквивалентно) WHERE Oklad<3000 **OR** Oklad>5000

3. Принадлежность множеству : оператор IN

Используется для сравнения некоторого значения со списком заданных значений, при этом проверяется, соответствует ли результат вычисления выражения одному из значений в предоставленном списке.

При помощи оператора **IN** может быть достигнут тот же результат, что и в случае применения оператора **OR**, однако оператор IN выполняется быстрее.

NOT IN используется для отбора любых значений, кроме тех, которые указаны в представленном списке.

Синтаксис

SELECT
[{* | [имя_столбца [[AS] псевдоним]] [,...n] }
FROM имя_таблицы [[AS] псевдоним] [,...n]
WHERE <имя столбца1> **IN (**<Значение1>, <Значение2>, [,...n])

Приклад 3

Задача.

Вивести інформація про групи з ідентифікаторами 1 або 2 або 3.

Рішення.

SELECT *
FROM [zkVnz].[dbo].[Student]
WHERE [StudyGroupId] **IN** (1,2,3)
Или (что эквивалентно)
WHERE [StudyGroupId]=1 **OR** [StudyGroupId]=2 **OR** [StudyGroupId]=3

4. Соответствие шаблону : оператор LIKE

Обычная форма "имя_столбца LIKE текстовая_константа" для столбца текстового типа позволяет отыскать все значения указанного столбца, соответствующие образцу, заданному "текстовой_константой".

Символы :

% – вместо этого символа может быть подставлено любое количество произвольных символов
_ - заменяет один символ строки
[] – вместо символа строки будет подставлен один из возможных символов, указанный в этих ограничителях
[^] – вместо соответствующего символа строки будут подставлены все символы, кроме указанных в ограничителях

Приклад 4

в номере телефона вторая цифра – 4:	Like "_4%"
в номере телефона вторая цифра – 2 или 4:	Like "_[24]%"
в номере телефона вторая цифра 2, 3 или 4:	Like "_[2-4]%"
в фамилии встречается слог "ро":	Like "%ро%"
фамилия начинается с буквы "С":	Like N'С%‘
первая буква в фамилии неизвестна:	Like '_иденко'
в дате рождения известен год:	Like '1990%‘
в дате рождения известен месяц:	Like '%-07-%‘

5. Значение NULL

Оператор IS NULL используется для сравнения текущего значения со значением NULL – специальным значением, указывающим на отсутствие любого значения

Синтаксис

SELECT
[{* | [имя_столбца [[AS] псевдоним]] [,...n] }
FROM имя_таблицы [[AS] псевдоним] [,...n]
WHERE <имя столбца1> **is not null** <Значение>
WHERE <имя столбца1> **is null** <Значение>

Приклад 5

Задача.

Вывести людей, у которых:

- не задано отчества
- задано отчества

Рішення.

```
SELECT [Surname],[Name],[Patronymic],[Duties],[Oklad]
FROM [zkVnz].[dbo].[Employee]
WHERE [Patronymic] is null
--WHERE [Patronymic] is not null
```

2.3 Выборка с упорядочением ORDER BY

ORDER BY позволяет упорядочить выбранные записи в порядке возрастания или убывания значений любого столбца или комбинации столбцов, независимо от того, присутствуют эти столбцы в таблице результата или нет

По умолчанию реализуется сортировка по возрастанию - ключевое слово ASC

Для выполнения сортировки в обратной последовательности необходимо после имени поля, по которому она выполняется, указать ключевое слово DESC

Фраза ORDER BY всегда должна быть последним элементом в операторе SELECT

Приклад 5

Вивести інформацію про співробитників. Отсортувати результат:

- по прізвищу
- по прізвищу та імені
- по псевдоніму стовбу, що обраховується, від більшого значення до меншого

Рішення.

```
SELECT [EmployeeId]
    ,[Surname]
    ,[Name]
    ,[Patronymic]
    ,[Duties]
    ,[Oklad]
    ,[DcSubdivisionId]
    ,[City]
    ,[Nadbavka]
    ,(Oklad+Nadbavka) as Vuplata --- стовбець що обраховується, з присвоєним псевдонімом Vuplata
  FROM [zkVnz].[dbo].[Employee]
   ORDER BY [Surname]
---ORDER BY [Surname], [Name]
---ORDER BY [Surname] DESC,[Vuplata]
```

2.4 Агрегирующие функции

Агрегирующие функции - это функции, которые совершают действия над совокупностью одинаковых полей в группе записей ,т.е. применяются к наборам строк, а не к отдельным строкам

Информация, содержащаяся в многочисленных строках, обрабатывается каким-то определенным способом, а затем оформляется в виде результирующих данных, состоящих из одной строки

Агрегирующие функции часто используются вместе с конструкцией **GROUP BY**

При вызове большинства агрегирующих функций может использоваться ключевое слово **ALL** или **DISTINCT.** Параметр **ALL** (по умолчанию) и указывает на то, что действие функции должно распространяться на все значения в выражении, даже если одно и то же значение появляется несколько раз; параметр **DISTINCT** означает, что значение должно быть включено в функцию только один раз, даже если обнаруживается несколько дубликатов этого значения.

Параметр **<expression>** не может представлять собой подзапрос

Основные итоговые функции

- **Функция AVG**

AVG ([ALL | DISTINCT] <expression>)

- Возвращает арифметическое среднее значение (для всех непустых значений в столбце), представленное в параметре <expression>
- Поддерживаемый тип данных для <expression> - числовой
- NULL-значения игнорируются

- **Функция COUNT**

COUNT ([ALL I DISTINCT] <expression> | *)

- Возвращает данные о кол-ве элементов, которые представлены в параметре <expression> (Выполняет подсчет всех строк в результирующем наборе данных вплоть до 2147483647)
- Поддерживаемый тип данных для <expression> - int | любой тип данных
- При использовании значения параметра * происходит возврат данных о количестве строк в таблице;
- Дублирующиеся значения или NULL-значения не исключаются

Приклад 6

Задача.

Отримати кількість записів в таблиці.

Рішення.

```
SELECT count (*)
FROM [zkVnz].[dbo].[Employee]
```

Задача.

Отримати кількість прізвищ в таблиці.

Рішення.

```
SELECT COUNT (Surname)
FROM [zkVnz].[dbo].[Employee]
```

Задача.
Отримати кількість прізвищ в таблиці, що неповторюються
Рішення.
```
SELECT COUNT (distinct Surname)
FROM [zkVnz].[dbo].[Employee]
```

Задача.
Отримати кількість прізвищ в таблиці, що неповторюються та починаються на Петр.
Рішення.
```
SELECT COUNT (Surname)
FROM [zkVnz].[dbo].[Employee]
where Surname Like 'Петр%'
```

- **Функция MAX | MIN**

MAX | MIN ([ALL | DISTINCT] <expression>)

- Возвращает максимальное | минимальное из значений, представленных в параметре <expression> (Возвращает из столбца макс/мин число, самую раннюю/поздюю дату-время)
- Поддерживаемый тип данных для <expression> - числовой, строковый, даты-времени
- NULL-значения игнорируются

Приклад 7
Задача.
Вивести максимальний оклад серед співробітників з посадою доцент.
прізвиша яких починаються на Петр.
Рішення.
```
SELECT max (Oklad)
FROM [zkVnz].[dbo].[Employee]
where Duties='доцент'
```

- **Функция SUM**

SUM ([ALL I DISTINCT] <expression>)

- Функция SUM возвращает сумму всех значений, представленных в параметре <expression>
- Поддерживаемый тип данных для <expression> - числовой

NULL-значения игнорируются

Приклад 8
Задача.
Вывести общую сумму выплат в организации и общее количество должностей, которое определено в организации
Рішення.
```
SELECT SUM([Oklad]),COUNT([Duties])
FROM [zkVnz].[dbo].[Employee]
```

2.5 Группировка в запросах

Группировка записей

- Для группировки записей по полям или выражениям применяется раздел GROUP BY оператора SELECT, что позволяет применять для каждой группы функции агрегирования
- Синтаксис данной части :

[GROUP BY ВыражениеГруппировки, […n]]

- При группировке записей допускается также использование раздела WHERE, в этом случае группируются записи, удовлетворяющие этому условию
- Раздел WHERE позволяет определить, какие записи должны подвергнуться группировке, а раздел HAVING – какие группы должны быть выведены в итоговый набор данных
- Ключевое слово HAVING можно использовать только в разделе GROUP BY

Приклад 9

Задача.

определить значения для подразделений (+псевдоним для выводимого поля)

Рішення.

Обработка:

1-группируются записи с одинаковым идентификатором
2-подсчитываются значения

```
SELECT DcSubdivisionId,
Count(distinct EmployeeId) as Count,
Sum(Oklad) as Sum, Avg(Oklad) as [Avg],
Min(Oklad) as [Min],Max(Oklad) as [Max]
FROM Employee
GROUP BY DcSubdivisionId
```

Приклад 10

Определить значения для подразделений по датам

Рішення.

Обработка:

1-группируются записи с одинаковым идентификатором (слева, направо)
2-группируются записи с одинаковой датой по полученной группировке на 1 шаге
3-подсчитываются значения

```
SELECT DcSubdivisionId,DataVuplatu,
Count(*) as Count,
Sum(Oklad) as Sum, Avg(Oklad) as [Avg],
Min(Oklad) as [Min],Max(Oklad) as [Max]
FROM Employee
GROUP BY DcSubdivisionId, DataVuplatu
```

Приклад 11

Определить общие финансовые выплаты по каждому подразделению, провести сортировку полученного результата

Задать псевдоним для выводимого поля

Рішення.

Обработка:

1-группируются записи с одинаковым идентификатором

2-подсчитывается сумму в каждой группе

3-сортируются данные

```
SELECT DcSubdivisionId,
                SUM([Oklad]) As Itogo
FROM [zkVNZnew].[dbo].[Employee]
GROUP BY DcSubdivisionId
order by Itogo   /* по возрастанию*/
--order by Itogo DESC  /*по убыванию*/
```

Приклад 12

Вывести среднюю выплату окладов для подразделений, при условии обработки только окладов **> 3000**

Задать псевдонимы для выводимых полей

Рішення.

Обработка:

1 - извлечение строк, у которых Oklad > 3000

2 - полученная выборка группируется по подразделениям

3 - подсчет средних выплат по сгруппированным подразделениям

```
SELECT DcSubdivisionId,
        AVG(Oklad) As ItogoOklad,
        COUNT (EmployeeId) As Staff
FROM zkVnz.dbo.Employee
WHERE Oklad > 3000
GROUP BY DcSubdivisionId
```

Приклад 13

Вывести сколько фамилий начинается на «Пе»

Рішення.

Обработка:

1 – фамилии группируются по совпадению

2 - выбираются только те, которые начинаются на «Пе»

3 - подсчитывается полученная выборка

```
SELECT Surname,  COUNT (Surname)
FROM [zkVnz].[dbo].[Employee]
GROUP BY Surname
HAVING Surname LIKE N'Пе%'
```

ТЕЗИСЫ - Соединения и теоретико-множественные операции над отношениями

Выборка данных из нескольких таблиц

Большая часть запросов обращается не к одной, а нескольким таблицам, перечень которых и условия их соединения указываются в предложении FROM оператора SELECT

Присвоение псевдонима таблице способствует уменьшению кода и улучшает восприятия текста запроса

Псевдоним должен быть уникален в рамках одного запроса

Операции над отношениями:

1. Выборка
2. Проекция
3. Декартово произведение
4. Соединение

1. Выборка

Операция выборки - построение горизонтального подмножества, т.е. подмножества кортежей, обладающих заданными свойствами

Выборка таблицы **создается** из тех ее **строк**, которые удовлетворяют заданным условиям

Применяется для одной таблицы

Пример:

Вывести информацию о сотрудниках с окладом > 3000

```
SELECT [Surname] ,[Name] ,[Patronymic] ,[Oklad]
FROM [zkVnz].[dbo].[Employee]
WHERE [Oklad] > 3000
```

2. Проекция

Операция проекции - построение вертикального подмножества отношения, т.е. подмножества кортежей, получаемого выбором одних и исключением других атрибутов

Проекция таблицы **создается** из указанных ее **столбцов** (в заданном порядке) с последующим **исключением** избыточных **дубликатов** строк

Применяется для одной таблицы

Пример:

Вывести без повторов ФИО, должность Сотрудника

```
SELECT DISTINCT
    [Surname],[Name],[Patronymic]
FROM [zkVnz].[dbo].[Employee]
```

3. Декартово произведение

Операция декартового произведения – это построение множества, когда каждая строка из первой таблицы соединяется с каждой строкой второй таблицы

В результате количество строк результирующего набора равно произведению количества строк операндов декартова произведения

Пример
Таблица-1 содержит 5 строк, а таблица-2 – 16 строк
В результате получается 5 * 16 = 80 строк

- **Синтаксис**

SELECT a.column , b.column
FROM table1 **a CROSS JOIN** table2 **b**

4. Операция соединения

Соединение - это процесс, когда две или более таблицы объединяются в одну

Правила:

- нельзя объединять таблицы без объединения соответствующих столбцов
- для объединения N таблиц должно быть как минимум N-1 условие
- для **улучшения скорости** обработки запроса желательно возле имя столбца в разделе SELECT указывать также имя (**аллиас**) используемой таблицы
- если одно и то же имя столбца используется в нескольких таблицах то независимо от того в каком разделе выполняется запрос – необходимо в качестве префикса указать имя используемой таблицы или аллиас

Для задания типа соединения таблиц в логический набор записей, из которого будет выбираться необходимая информация, используется операция JOIN в предложении FROM

Аллиасы имен таблиц

Аллиас (синоним) – бывает необходим при написании запросов для уменьшения размера скрипта и увеличения, как производительности так удобночитаемости SQL кода

Синтаксис:

```
SELECT a.column , b.column
FROM table1 a, table2 b
WHERE a.column = b.column
```

Свойства:

- После имени таблицы следует аллиас
- Обозначив таблицу через определенную букву – в дальнейшем используется эта букву вместо полного названия таблицы в качестве префикса
- Длина названия аллиаса не должна превышать 30 символов
- Аллиас используемый в разделе FROM может быть использован также в разделе WHERE, SELECT , HAVING и GROUP BY

Аллиасы доступными лишь для текущего запроса

4.1 Соединение по равенству

Неявный синтаксис соединения (старый стиль синтаксиса соединения)

ОР - каждая операция соединения определена неявно через выражение WHERE, используя так называемые *столбцы соединения.* Обычно в качестве ключевых столбцов для объединения служит ограничения целостности

Синтаксис

```
SELECT table1.column,  table2.column
FROM   table1,table2
WHERE table1.column = table2.column
```

Условие типа table1.column = table2.column – является условием объединения таблиц при условии, что между этими столбцами существует смысловая связь

Можно использовать аллиасы

Пример:

Вывести ФИО и название студенческой группы

```
SELECT a.[Surname],a.[Name],a.[Patronymic]
      ,b.Name
FROM Student a, StudyGroup b
where a.StudyGroupId=b.StudyGroupId
```

4.2 Операция соединения

Команд JOIN в предложении FROM

Явный синтаксис соединения (SQL ANSI:1992 синтаксис соединения)

В языке SQL для задания типа соединения таблиц в логический набор записей, из которого будет выбираться необходимая информация, используется операция **JOIN** в предложении **FROM**

Синтаксис:

FROM имя_таблицы_1 **{INNER | LEFT | RIGHT | FULL OUTER }**
    JOIN имя_таблицы_2
        ON условие_соединения_1-2
    JOIN имя_таблицы_3
        ON условие_соединения_1-3
…

Виды соединений:

- *Внутреннее соединение* – в результат включаются только совпадающие записи ведущей таблицы (таблица сразу после оператора FROM)
- *Внешнее соединение* – в результирующий набор данных включаются также записи ведущей таблицы соединения, которые объединяются с пустым множеством записей другой таблицы

4.2.1 Внутренние соединение (или тета-соединение) (по эквивалентности если =)

Команда INNER JOIN

Внутренние соединение

A INNER JOIN B

INNER JOIN – внутреннее соединение, включающее только совпадающие по условию соединения записи из соединяемых таблиц

4.2.2 Левое внешнее соединение.

Команда LEFT OUTER JOIN

Левое внешнее соединение

A LEFT OUTER JOIN B

LEFT (OUTER) – левое (внешнее). Это соединение включает в себя *все строки из таблицы A* (совпадающие и несовпадающие) плюс *совпадающие значения из таблицы B*.

Для строк из таблицы A, которым не найдено соответствие, значения NULL заносятся в столбцы, извлекаемые из таблицы B.

```
SELECT R.a1,  R.a2, S.b1, S.b2
FROM R LEFT JOIN S ON R.a2=S.b1
```

4.2.3 Правое внешнее соединение.

Команда RIGHT OUTER JOIN

Правое внешнее соединение

A RIGHT OUTER JOIN B

RIGHT (OUTER) – правое (внешнее). Это соединение является обратным предыдущему, т.е. все строки из таблицы B (правой таблицы) представлены в соединении и они дополнены совпадающими строками из таблицы A, в столбцы для строк, не имеющих совпадения, заносятся значения NULL.

4.2.4 Полное внешнее соединение

Команда FULL OUTER JOIN

Полное внешнее соединение

A FULL OUTER JOIN B

FULL (OUTER) – полное (внешнее). Это комбинация левого и правого соединения. Присутствуют все строки из обоих таблиц. Если строки совпадают, то они заполнены реальными значениями. В несовпадающих строках значения столбцов заполняются значениями NULL.

Требования к объединении результатов двух или более запросов в одну таблицу
Предложения UNION | INTERSECT | EXCEPT

Чтобы таблицы результатов нескольких запросов можно было объединять с помощью предложений, они должны соответствовать ***следующим требованиям***:

- содержать одинаковое число столбцов
- тип данных каждого столбца любой таблицы должен совпадать с типом данных соответствующего столбца любой другой таблицы
- ни одна из таблиц промежуточного запроса не может быть отсортирована с помощью предложения ORDER BY
- Предложение ORDER BY можно указывать только в конце инструкции. Это предложение нельзя использовать внутри отдельных запросов, составляющих инструкцию.
- разрешается использовать в списке возвращаемых элементов только имена столбцов или указывать все столбцы (SELECT *) и запрещается использовать выражения
- Предложения GROUP BY и HAVING можно использовать только внутри отдельных запросов; их нельзя использовать для того, чтобы повлиять на конечный результирующий набор.

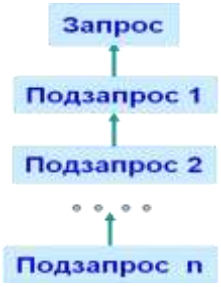
Объединение таблиц с помощью данных предложений **отличается от соединений таблиц** тем, что в нем ни один из запросов не управляет другим запросом. Все запросы выполняются **независимо друг от друга**, а уже их *вывод объединяется.*

Замечания

- Операторы UNION, EXCEPT и INTERSECT нельзя использовать вместе с инструкцией INSERT.
- Первый запрос может содержать предложение INTO, создающее таблицу, в которой будет храниться результирующий набор. Предложение INTO можно использовать только в первом запросе. Если предложение INTO будет указано в любом другом месте, SQL Server возвратит сообщение об ошибке.

Подзапросы

Подзапрос – это команда SELECT, вложенная в предложение другой команды SQL. Подзапросы могут использоваться в командах SELECT, UPDATE, INSERT, DELETE, CREATE TABLE. Например, каждая команда SELECT может включать в себя несколько других команд SELECT[1]. При этом подзапрос (внутренний запрос) генерирует значение, которое проверяется в предикате внешнего запроса.



Подзапросы всегда выполняются от внутренних к внешнему, если только не являются коррелированными. Подзапрос может возвращать одну и более строк или один и более столбцов.

1. Подзапрос помещается в круглые скобки и должен стоять в правой части оператора сравнения внешнего запроса.
2. Подзапрос может обращаться к таблицам отличным от тех, к которым обращается основной запрос.
3. Подзапрос может задаваться в сложных критериях поиска внешних запросов с использованием логических связок AND и OR.
4. Предложение ORDER BY ставится последним в основном запросе и не может содержаться в подзапросе.
5. В команде SELECT подзапрос может стоять в предложениях FROM, WHERE, HAVING.
6. Подзапрос может содержать группы и групповые функции.
7. Имена столбцов в предложении SELECT внутреннего запроса должны стоять в той же последовательности, что и имена столбцов в левой части оператора сравнения внешнего запроса. Типы столбцов должны попарно соответствовать.
8. В критерии поиска могут использоваться логические операторы, операторы ANY (SOME), ALL.

Подзапрос на уровне предложения WHERE

Подзапрос вычисляет среднюю оценку и подставляет высчитанное значение в предложение WHERE внешнего запроса.

[1]Допускается до 255 уровней вложенности подзапросов.

Коррелированные подзапросы на уровне предложения WHERE

Коррелированные подзапросы – это вложенные подзапросы. Они выполняются для каждой строки главного запроса.

Последовательность выполнения коррелированного подзапроса:

- внешний запрос выбирает строку;
- выполняется внутренний запрос, используя значение строки внешнего запроса;
- результат выполнения внутреннего запроса возвращается во внешний запрос, где проверяется его соответствие выбранной строке;
- выбирается следующая строка внешнего запроса.

При задании вложенных запросов допускаются применение операторов ANY, EXISTS, ALL и логических операторов.

Подзапрос на уровне предложения HAVING

Наибольшее затруднение, как ни странно, вызывают функции по работе с датами, поэтому ниже мы приводим наиболее употребляемые функции (см. Приложения 2-3).

Объединение двух или более запросов с помощью UNION

UNION – специальный оператор, с помощью которого можно из двух или более запросов построить единое результирующее множество, т.е. выходные данные одного запроса присоединяются к выходным данным другого запроса.

Основные правила при реализации операций над множествами

- запросы, соединяемые оператором UNION должны иметь одинаковое количество столбцов в предложении SELECT;
- возвращаемый комбинированный результат будет иметь заголовки столбцов первого предложения SELECT;
- тип данных каждого столбца должен быть совместим с типом данных соответствующего столбца другого запроса;
- по умолчанию режимом вывода для UNION является DISTINCT.

После выполнения операции UNION ALL каждое имя будет выведено столько раз, сколько раз оно встречается в запросах.

Предложение ORDER BY в операциях над множествами может стоять только в последнем предложении запроса, при этом вместо имен столбцов используются их номера из предложения SELECT.

ТЕЗИС - **З**апросы модификации данных в таблице

- ***Добавление данных - INSERT***
- ***Обновления данных - UPDATE***
- ***Удаления данных - DELETE***

Команда добавления - INSERT

Вставка данных из одной Таблицы в Другую (импорт данных)

Таблицы в пределах одной БД

Скопировать записи из одной таблицы в другую с **одинаковой** структурой

```
INSERT INTO  table1
SELECT *
FROM table2
[Where <условие_для_данных_table2>]
```

Где

Table1 – предварительно созданная таблица, в которую записываются данные

Table2 - исходная таблица

Скопировать записи из одной таблицы в другую если структура **различна**

```
INSET INTO  table1  (a,b,c,d)
SELECT  a, b, c, d
FROM  table2
[Where <условие>]
```

Где

a,b,c,d – перечень полей таблиц

Таблицы в разных БД

Скопировать записи из одной таблицы в другую если структура **одинакова**

```
INSET INTO DB1.dbo.table 1
SELECT  *
FROM DB2.dbo.table 2
[Where <условие_для_данных_DB2.dbo.table2 >]
```

Заполнение данными НОВОЙ Таблицы из существующей Таблицы (экспорт данных)

Таблицы в одной БД

Скопировать записи из одной таблицы в другую с **одинаковой** структурой

```
SELECT *
INTO  table 1
FROM table2
[Where < условие_для_данных_ table2 >]
```

Table1 – таблица, в которую копируются (экспортируются) данные, таблица предварительно **не** создается

Table2 - исходная таблица

Таблицы в разных БД

Скопировать записи из одной таблицы в другую с **одинаковой** структурой

```
SELECT *
 INTO  DB1.dbo.table 1
FROM DB2.dbo.table2
[Where < условие_для_данных_ DB2.dbo.table2>]
```

DB1.dbo.Table1 – таблица, в которую копируются (экспортируются) данные, таблица предварительно **не** создается

DB1.dbo.Table2 - исходная таблица, принадлежащая другой БД

Обновление данных в Таблице
Команда обновления данных - UPDATE

UPDATE <Имя_таблицы>
SET
<Имя_столбца1> = <Выражение1>,
[<Имя столбца2> = <Выражение2>,]
…
[WHERE <Условие_отбора>]
Где

- <Имя_таблицы> – имя таблицы, в которой будет обновление данных
- <Имя поля1>, <Имя поля2> - имена изменяемых полей
- <Выражение1>, <Выражение 2> - новое значение, совместимое по типу данных (либо конкретные значения, либо NULL, либо операторы SELECT)
- Здесь:
  - SELECT применяется как функция
  - <Условие> – условие, которым должны соответствовать записи, поля которых изменяем. Если не задать условие, будет обновлены ВСЕ строки таблицы

Удаление данных из Таблице

Команда удаления ОПРЕДЕЛЕННЫХ данных

DELETE
from <Имя таблицы>
[WHERE <Условие>]

Где

<Условие> - условие, которым удовлетворяют удаляемые записи, если условие не указаны, то **удаляются все столбцы таблицы**. Если условие указано то удаляются записи поля которых соответствуют условию.

Удаление ВСЕХ данных из столбцов в Таблице

DELETE <Имя таблицы>