# Структура следования. Использование структуры следования, общий вид и структура следования. Блок схема операторов следования.

## *CASE (Transact-SQL)*

Оценка списка условий и возвращение одного из нескольких возможных выражений результатов.

Выражение CASE имеет два формата:

- простое выражение CASE для определения результата, которое сравнивает выражение с набором простых выражений;
- поисковое выражение CASE для определения результата, которое вычисляет набор логических выражений.

Оба формата поддерживают дополнительный аргумент ELSE.

Выражение CASE может использоваться в любой инструкции или предложении, которые допускают допустимые выражения. Например, выражение CASE можно использовать в таких инструкциях, как SELECT, UPDATE, DELETE и SET, а также в таких предложениях, как select_list, IN, WHERE, ORDER BY и HAVING.

*Синтаксис*

```
Simple CASE expression:
CASE input_expression
     WHEN when_expression THEN result_expression [ ...n ]
     [ ELSE else_result_expression ]
END
Searched CASE expression:
CASE
     WHEN Boolean_expression THEN result_expression [ ...n ]
     [ ELSE else_result_expression ]
END
 Аргументы
input_expression
```

Выражение, полученное при использовании простого формата функции CASE. Аргумент input_expression представляет собой любое допустимое выражение.

`WHEN when_expression`

Простое выражение, с которым сравнивается аргумент input_expression при использовании простого формата функции CASE. Аргумент when_expression представляет собой любое допустимое выражение. Типы данных аргумента input_expression и каждого из выражений when_expression должны быть одинаковыми или неявно приводимыми друг к другу.

`THEN result_expression`

Выражение, возвращаемое, если сравнение выражений input_expression и when_expression дает в результате TRUE или выражение Boolean_expression вычисляется в TRUE. Аргумент result_expression представляет собой любое допустимое выражение.

`ELSE else_result_expression`

Это выражение, возвращаемое, если ни одна из операций сравнения не дает в результате TRUE. Если этот аргумент опущен и ни одна из операций сравнения не дает в результате TRUE, функция CASE возвращает NULL. Аргумент else_result_expression представляет собой любое допустимое выражение. Типы данных аргумента else_result_expression и любого из аргументов result_expression должны быть одинаковыми или неявно приводимыми друг к другу.

`WHEN Boolean_expression`

Это логическое выражение, полученное при использовании поискового формата функции CASE. Аргумент Boolean_expression представляет собой любое допустимое логическое выражение.

*Замечания*

- SQL Server допускает применение в выражениях CASE не более 10 уровней вложенности.

- Выражение CASE нельзя использовать для управления потоком выполнения инструкций Transact-SQL, блоков инструкций, определяемых пользователем функций и хранимых процедур.

### *Типы результата*

Возвращает выражение с наивысшим приоритетом из набора выражений `result_expressions` и необязательного выражения `else_result_expression`.

- Результирующие значения
- Простое выражение CASE

Простое выражение CASE сравнивает первое выражение с выражением в каждом предложении WHEN. Если эти выражения эквивалентны, то возвращается выражение в предложении THEN.

Вычисляет выражение `input_expression`, затем в указанном порядке сравнивает значения выражений `input_expression` и `when_expression` для каждого предложения WHEN.

Возвращает выражение `result_expression`, соответствующее первому предложению WHEN, для которого операция сравнения `input_expression = when_expression` вычисляется в TRUE.

Если ни одна из операций `input_expression = when_expression` не вычисляется в TRUE, компонент SQL Server Database Engine возвращает выражение `else_result_expression`, если указано предложение ELSE, или значение NULL, если предложение ELSE не указано.

### *Поисковое выражение CASE*

Вычисляет в указанном порядке выражения `Boolean_expression` для каждого предложения WHEN.

Возвращает выражение `result_expression`, соответствующее первому предложению WHEN, для которого выражение `Boolean_expression` вычисляется в TRUE.

Если ни одно выражение `Boolean_expression` не вычисляется в TRUE, компонент Database Engine возвращает выражение `else_result_expression`, если указано предложение ELSE, или значение NULL, если предложение ELSE не указано.

### *Примеры*

- Использование инструкции SELECT с простым выражением CASE

При использовании в инструкции SELECT простое выражение CASE позволяет выполнить только проверку на равенство. Другие проверки не выполняются. В следующем примере выражение CASE используется для изменения способа отображения категорий линейки продуктов с целью сделать их более понятными.

```
USE AdventureWorks;
GO
SELECT   ProductNumber, Category =
      CASE ProductLine
         WHEN 'R' THEN 'Road'
         WHEN 'M' THEN 'Mountain'
         WHEN 'T' THEN 'Touring'
         WHEN 'S' THEN 'Other sale items'
         ELSE 'Not for sale'
      END,
   Name
FROM Production.Product
ORDER BY ProductNumber;
GO
```

- Использование инструкции SELECT с поисковым выражением CASE

При использовании в инструкции SELECT поисковое выражение CASE позволяет заменять значения в результирующем наборе в зависимости от

результатов сравнения. В следующем примере отображается список цен в виде текстового комментария, основанного на диапазоне цен для продукта.

```
USE AdventureWorks;
GO
SELECT   ProductNumber, Name, 'Price Range' =
      CASE
         WHEN ListPrice =  0 THEN 'Mfg item - not for resale'
         WHEN ListPrice < 50 THEN 'Under $50'
         WHEN ListPrice >= 50 and ListPrice < 250 THEN 'Under $250'
         WHEN ListPrice >= 250 and ListPrice < 1000 THEN 'Under $1000'
         ELSE 'Over $1000'
      END
FROM Production.Product
ORDER BY ProductNumber ;
GO
```

- Использование функции CASE для замены функции If, используемой в `Microsoft Access`

Возможности функции CASE схожи с возможностями функции If СУБД `Microsoft Access`. Следующий пример показывает простой запрос, использующий функцию If для задания выходного значения столбца `TelephoneInstructions` таблицы `Access` с именем `db1.ContactInfo`.

```
SELECT FirstName, LastName, TelephoneNumber,
     If(IsNull(TelephoneInstructions),"Any time",
     TelephoneInstructions) AS [When to Contact]
FROM db1.ContactInfo;
```

В следующем примере выражение CASE используется для задания выходного значения столбца TelephoneSpecialInstructions в представлении AdventureWorks, Person.vAdditionalContactInfo.

```
USE AdventureWorks;
GO
SELECT FirstName, LastName, TelephoneNumber, 'When to Contact' =
     CASE
          WHEN TelephoneSpecialInstructions IS NULL THEN 'Any time'
          ELSE TelephoneSpecialInstructions
     END
FROM Person.vAdditionalContactInfo;
```

- Использование выражения CASE в предложении ORDER BY

В следующем примере выражение CASE используется в предложении `ORDER BY`, чтобы определить порядок сортировки строк на основе значения в столбце `SalariedFlag` таблицы `HumanResources.Employee`. Сотрудники, для которых столбец `SalariedFlag` имеет значение 1, возвращаются в порядке `EmployeeID` (по убыванию). Сотрудники, для которых столбец SalariedFlag имеет значение 0, возвращаются в порядке `EmployeeID` (по возрастанию).

```
SELECT EmployeeID, SalariedFlag
FROM HumanResources.Employee
ORDER BY CASE SalariedFlag WHEN 1 THEN EmployeeID END DESC
        ,CASE WHEN SalariedFlag = 0 THEN EmployeeID END;
GO
```

- Использование выражения CASE в инструкции UPDATE

В следующем примере выражение CASE используется в инструкции UPDATE, чтобы определить значение, устанавливаемое в столбце `VacationHours` для сотрудников, у которых столбец `SalariedFlag` имеет значение 0. Если при вычитании 10 часов из `VacationHours` получается отрицательное значение, `VacationHours` увеличивается на 40 часов. В противном случае значение `VacationHours` увеличивается на 20 часов. С помощью предложения `OUTPUT` отображаются исходная и обновленная продолжительность отпуска.

```
USE AdventureWorks;
GO
UPDATE HumanResources.Employee
SET VacationHours =
    ( CASE
         WHEN ((VacationHours - 10.00) < 0) THEN VacationHours + 40
         ELSE (VacationHours + 20.00)
       END
    )
OUTPUT Deleted.EmployeeID, Deleted.VacationHours AS BeforeValue,
       Inserted.VacationHours AS AfterValue
WHERE SalariedFlag = 0;
```

- Использование выражения CASE в инструкции SET

В следующем примере выражение CASE используется в инструкции SET, в возвращающей табличное значение функции dbo.GetContactInfo. В базе данных AdventureWorks все данные, связанные с физическими лицами, хранятся в таблице Person.Contact. Например, физическим лицом может являться сотрудник, представитель поставщика, представитель магазина или покупатель. Функция возвращает имя и фамилию заданного ContactID и тип контакта для этого лица. Выражение CASE в инструкции SET определяет значение, отображаемое для столбца ContactType в зависимости от наличия ContactID в таблицах Employee, StoreContact, VendorContact или Individual (покупатели).

```
USE AdventureWorks;
GO
CREATE FUNCTION dbo.GetContactInformation(@ContactID int)
RETURNS @retContactInformation TABLE
(
      ContactID int NOT NULL,
      FirstName nvarchar(50) NULL,
      LastName nvarchar(50) NULL,
      ContactType nvarchar(50) NULL,
    PRIMARY KEY CLUSTERED (ContactID ASC)
)
AS
-- Returns the first name, last name and contact type for the specified contact.
BEGIN
    DECLARE
        @FirstName nvarchar(50),
        @LastName nvarchar(50),
        @ContactType nvarchar(50);

    -- Get common contact information
    SELECT
        @ContactID = ContactID,
        @FirstName = FirstName,
        @LastName = LastName
    FROM Person.Contact
    WHERE ContactID = @ContactID;

    SET @ContactType =
        CASE
            -- Check for employee
```

```
            WHEN EXISTS(SELECT * FROM HumanResources.Employee AS e
                WHERE e.ContactID = @ContactID)
                THEN 'Employee'

            -- Check for vendor
            WHEN EXISTS(SELECT * FROM Purchasing.VendorContact AS vc
                    INNER JOIN Person.ContactType AS ct
                    ON vc.ContactTypeID = ct.ContactTypeID
                WHERE vc.ContactID = @ContactID)
                THEN 'Vendor Contact'

            -- Check for store
            WHEN EXISTS(SELECT * FROM Sales.StoreContact AS sc
                    INNER JOIN Person.ContactType AS ct
                    ON sc.ContactTypeID = ct.ContactTypeID
                WHERE sc.ContactID = @ContactID)
                THEN 'Store Contact'

            -- Check for individual consumer
            WHEN EXISTS(SELECT * FROM Sales.Individual AS i
                WHERE i.ContactID = @ContactID)
                THEN 'Consumer'
        END;

    -- Return the information to the caller
    IF @ContactID IS NOT NULL
    BEGIN
        INSERT @retContactInformation
        SELECT @ContactID, @FirstName, @LastName, @ContactType;
    END;

    RETURN;
END;
GO
SELECT ContactID, FirstName, LastName, ContactType
FROM dbo.GetContactInformation(2200);
GO
SELECT ContactID, FirstName, LastName, ContactType
FROM dbo.GetContactInformation(5);
```

- Использование выражения CASE в предложении HAVING

В следующем примере выражение CASE используется в предложении HAVING, чтобы ограничить строки, возвращаемые инструкцией SELECT.

Инструкция возвращает максимальную почасовую ставку для каждой должности в таблице `HumanResources.Employee`. Предложение HAVING ограничивает должности, оставляя только те, которые заняты мужчинами с максимальной почасовой ставкой более 40 долларов или женщинами с максимальной почасовой ставкой более 42 долларов.

```
USE AdventureWorks;
GO
SELECT Title, MAX(ph1.Rate)AS MaximumRate
FROM HumanResources.Employee AS e
JOIN HumanResources.EmployeePayHistory AS ph1 ON e.EmployeeID = ph1.EmployeeID
GROUP BY Title
HAVING (MAX(CASE WHEN Gender = 'M'
        THEN ph1.Rate
        ELSE NULL END) > 40.00
     OR MAX(CASE WHEN Gender  = 'F'
        THEN ph1.Rate
        ELSE NULL END) > 42.00)
ORDER BY MaximumRate DESC;
```