Условная конструкция CASE

Утром Антон шел на работу в надежде, что сегодня он завершит начатую в понедельник задачу. Но не тут-то было: в офисе его перехватил тимлид и утащил в переговорку, где собралась команда. Все спорили о том, как правильно рассчитать показатели в таблице, прототип которой хотели направить заказчику. Все дело было в формировании нового поля — тип клиента.

Один из руководителей предполагал следующую логику расчета для этого поля:

- если доход клиента менее 40 тыс. руб., тип клиента *Стандарт;*
- если доход клиента от 40 тыс. до 100 тыс. руб., тип клиента *Премиум;*
- если доход клиента более 100 тыс. руб., это уже *VIP.*

Второй настаивал, что нужна дополнительная группа *VIP Нестандарт* для клиентов с доходом выше полумиллиона рублей.

Антон предложил сделать два новых поля — каждое со своей логикой — и дать тем самым заказчику право выбора. Но как Антон сможет это реализовать, если пока ему известна только сумма дохода клиента? Очень просто: с помощью специальной условной конструкции **CASE**. Она позволяет создавать новые поля и заполнять их значениями на основе выполнения логических проверок над текущими полями таблицы.

У конструкции CASE есть два варианта синтаксиса:

Первая форма	Вторая форма

CASE
  WHEN условие_1 THEN значение_1
  [WHEN условие_2 THEN значение_2]
  [WHEN условие_3 THEN значение_3]
  ……
  [WHEN условие_N THEN значение_N]
[ELSE
значение_если_ничего_до_этого_не_подошло]
END
	CASE проверяемое_поле
	  WHEN с_чем_сравниваем_1 THEN значение_1
	  [WHEN с_чем_сравниваем_2 THEN значение_2]
	  [WHEN с_чем_сравниваем_3 THEN значение_3]
	
	…….
	  [WHEN с_чем_сравниваем_N THEN значение_N]
	[ELSE
	значение_если_ничего_до_этого_не_подошло]
	END

Заметьте, обе формы похожи. Обратите внимание на то, из чего состоит первая форма и как она работает, затем рассмотрите ее отличие от второй.

Ключевые слова конструкции **CASE**:

1. Конструкция всегда начинается со слова **CASE**, это обязательно.
2. Далее необходимо прописать логическую фразу: **WHEN *условие* THEN *значение*** (такая фраза еще называется веткой). Веток может быть несколько, но обязательно хотя бы одна должна быть. Каждая такая ветка является аналогом логического выражения, которое используется в повседневной жизни: ЕСЛИ выполняется некоторое условие, ТО необходимо сделать что-то (в данном случае заполнить поле определенным значением).

   К примеру, вы хотите создать новое поле, отражающее деление на две возрастные группы — совершеннолетние и несовершеннолетние. Тогда вам необходимо указать две ветки:

   - WHEN age >= 18 THEN 'Совершеннолетний' (Если возраст человека не менее 18 лет, то новое поле примет значение *Совершеннолетний*)
   - WHEN age < 18 THEN 'Несовершеннолетний' (Если возраст человека менее 18 лет, то новое поле примет значение *Несовершеннолетний*)

   Примечание: после WHEN можно указать как одно, так и несколько логических условий. Если условий больше одного, они должны быть соединены операторами OR или AND. К примеру,

   WHEN age < 18 AND age >0 THEN 'Несовершеннолетний'.

3. После всех веток можно указать **ELSE-значение**, это необязательное выражение и может быть использовано в конструкции **CASE** только один раз. Ключевое слово ELSE срабатывает, если для строки не выполнилась ни одна ветка, и присваивает значение, указанное после ELSE. В приведенном выше примере могут быть люди, у которых явно неправильный возраст (к примеру, меньше 0), поэтому невозможно отнести их ни к одной из возрастных групп. Таких людей вы можете объединить в группу *Указан неправильный возраст*, прописав следующее выражение: ELSE 'Указан неправильный возраст'.

   Важно! ELSE всегда присутствует в конструкции **CASE**, даже если это слово вы не указали. В таком случае оно будет скрыто, а значение нового поля будет заполнено NULL. Другими словами, если вы не прописали ELSE, это равносильно фразе ELSE NULL.

4. В конце стоит обязательное замыкающее слово END, которое показывает, что вы заканчиваете работать с условной конструкцией.

Теперь давайте обсудим алгоритм работы условной конструкции **CASE** (первой формы) на примере создания нового поля AGE_GROUP_NUM (Возрастная группа) в таблице person, в которой сейчас есть следующие поля:

- NAME — фамилия и имя;
- AGE — возраст.

NAME	AGE
Петров Сергей	10
Иванова Ольга	NULL
Балашов Василий	40
Антонов Борис	-47

Для начала сформируем запрос с конструкцией CASE, опираясь на логику, которую разработали выше на примере возрастных групп.

(Язык — SQL)
```
 SELECT
    name,
    age,
    CASE
         WHEN age >= 18 AND age <150 THEN 'Совершеннолетний'
         WHEN age >=0 AND age < 18 THEN 'Несовершеннолетний'
         ELSE 'Указан неправильный возраст'
   END as age_group_nm
FROM person
```

Примечание: каждую логическую фразу в конструкции **CASE** лучше писать на отдельной строке. Это необязательное действие, но в разы улучшает читаемость кода.

Итак, при работе запроса рассматривается некоторая строка таблицы и для нее проверяется первое условие конструкции (возраст больше или равен 18, но менее 150 лет). Рассмотрим первую строку таблицы: Петрову Сергею 10 лет, поэтому это условие не выполняется и запрос переходит к следующей проверке (возраст больше или равен 0, но меньше 18 лет). Поскольку это условие выполняется, для первой строки поле примет значение *Несовершеннолетний*, и, так как найдена ветка с подходящим условием, проверка заканчивается.

Далее для каждой строки из таблицы person алгоритм последовательно проходит по всем условиям конструкции **CASE**, пока не найдет верное утверждение. Если строка не удовлетворяет ни одному из условий, поле примет значение *Указан неправильный возраст*. Так произойдет со второй и четвертой строками таблицы person.

В итоге запрос выведет следующий результат:

NAME	AGE	AGE_GROUP_NM
Петров Сергей	10	Несовершеннолетний
Иванова Ольга	NULL	Указан неправильный возраст
Балашов Василий	40	Совершеннолетний
Антонов Борис	-47	Указан неправильный возраст

Однако возникла проблема: для Ивановой Ольги поле AGE_GROUP_NM заполнено некорректно, на самом деле ее возраст не указан вообще. Почему? Остался без внимания тот факт, что NULL при сравнении с любым числом или строкой выводит не True. Другими словами, строке с NULL всегда присваивается значение, стоящее после ELSE. Такую ошибку часто допускают специалисты, работающие с данными, — помните про NULL!

Чтобы решить эту проблему, можно случай с NULL вывести в отдельную ветку. Тогда запрос примет вид

(Язык — SQL)

```
SELECT
    name,
    age,
    CASE
        WHEN age >= 18 AND age <150 THEN 'Совершеннолетний'
        WHEN age >=0 AND age < 18 THEN 'Несовершеннолетний'
        WHEN age is NULL THEN 'Возраст не указан'
        ELSE 'Указан неправильный возраст'
   END as age_group_nm
FROM person
```

Наконец, мы получим правильный результат:

NAME	AGE	AGE_GROUP_NM
Петров Сергей	10	Несовершеннолетний
Иванова Ольга	NULL	~~Указан неправильный возраст~~ Возраст не указан
Балашов Василий	40	Совершеннолетний
Антонов Борис	-47	Указан неправильный возраст

Вторая форма конструкции CASE и прототип для заказчика

Вернемся ко второй форме конструкции **CASE**. Для нее сохраняются все правила использования ключевых слов первой формы, но появляются следующие ограничения:

- Логическую проверку можно выполнять только по одному полю. Оно должно быть указано между CASE и WHEN, а после WHEN необходимо сразу указать значение, с которым будет сравниваться атрибут.
- Проверяется только равенство значений, то есть любые проверки с использованием других операторов сравнения ('<', '>' и так далее) не могут быть произведены.

Как правило, вторая форма используется, когда проверку нужно провести по одному полю, которое принимает заранее известные значения, не равные NULL. Причем количество уникальных значений, которые может принимать это поле, обычно не более 10.

Поскольку на практике приходится устанавливать условия на несколько полей или использовать различные логические операторы сравнения, чаще используют первую форму — более универсальную.

Итак, мы разобрались в конструкции **CASE**, самое время вернуться к Антону: удалось ли ему справиться с поставленной задачей?

У него есть таблица с именами клиентов и информацией о них:

- LAST_NAME — фамилия клиента;
- FIRST_NAME — имя клиента;
- TRANS_AMT_AVG — средняя сумма транзакций в месяц;
- INCOME_AMT — месячный доход клиента.

Таблица customers

LAST_NAME	FIRST_NAME	TRANS_AMT_AVG	INCOME_AMT
Иванова	Екатерина	1512	20 170
Сидорян	Валентин	12 951	61 230
Орехова	Ольга	15 700	56 820
Гуров	Сергей	8610	184 030
Савинов	Василий	2472	38 240

После доработки таблицы Антон должен получить такую же таблицу с двумя дополнительными полями — CUSTOMER_TYPE_1 и CUSTOMER_TYPE_2 (типы клиента в соответствии с логикой каждого из руководителей):

LAST_NAME	FIRST_NAME	TRANS_AMT_AVG	INCOME_AMT	CUSTOMER_TYPE_1	CUSTOMER_TYPE_2
Иванова	Екатерина	1512	20 170	Стандарт	Стандарт
Сидорян	Валентин	12 951	612 300	VIP	VIP Нестандарт
Орехова	Ольга	15 700	56 820	Премиум	Премиум
Гуров	Сергей	8610	184 030	VIP	VIP
Савинов	Василий	2472	38 240	Стандарт	Стандарт

Для формирования этих полей Антон написал следующий запрос:

(Язык — SQL)
```
 SELECT
    last_name,
    first_name,
    trans_amt_avg,
    income_amt,
    CASE
        WHEN income_amt < 40000 THEN 'Стандартный'
        WHEN income_amt>= 40000 AND income_amt< 100000 THEN 'Премиум'
        WHEN income_amt>= 100000 THEN 'VIP'
    END as customer_type_1,
    CASE
        WHEN income_amt< 40000 THEN 'Стандартный'
        WHEN income_amt>= 40000 AND income_amt< 100000 THEN 'Премиум'
        WHEN income_amt>= 100000 AND income_amt< 500000 THEN 'VIP'
        WHEN income_amt>= 500000 THEN 'VIP Нестандарт'
    END as customer_type_2
FROM customers
```

Ну что же, задача решена!

Примечание: в примерах мы применили конструкцию **CASE** в SELECT, однако она используется и в WHERE, и в операторе группировки (о нем речь в следующих разделах). К примеру, если нужно вывести на экран информацию о людях, у которых неправильно указан возраст, вот что следует написать:

(Язык — SQL)
```
SELECT
    name,
    age
FROM person
WHERE
CASE
        WHEN age >= 18 AND age <150 THEN 'Совершеннолетний'
        WHEN age >=0 AND age < 18 THEN 'Несовершеннолетний'
        WHEN age is NULL THEN 'Возраст не указан'
        ELSE 'Указан неправильный возраст'
   END = 'Указан неправильный возраст'
```

Итоги:

- Конструкция **CASE** позволяет создавать новые поля и заполнять их значениями на основе выполнения логических проверок над текущими полями таблицы.
- Она имеет несколько обязательных ключевых слов: CASE, WHEN THEN, END и одно необязательное — ELSE. Логических веток WHEN THEN должно быть не менее одной, а все остальные ключевые слова могут быть использованы только один раз.
- Несмотря на то, что ELSE является необязательным словом, оно по умолчанию всегда есть в конструкции. Если вы не указываете его, это равносильно конструкции ELSE NULL.

- У конструкции **CASE** две формы синтаксиса. Чаще используют ту, где есть проверка по нескольким полям с различными логическими операторами.
- При работе конструкции каждая строка таблицы последовательно проверяется по всем логическим веткам конструкции **CASE**, пока не найдется удовлетворяющая строке ветка. Если она найдена, новое поле принимает значение, которое указано после слова THEN. Поэтому если одна и та же строка подходит под логику нескольких веток, новому полю присваивается значение первой среди них. Если для строки не подошла ни одна ветка, новое поле принимает значение, указанное после ELSE.
- При использовании конструкции **CASE** специалисты часто забывают про наличие NULL в данных, что может привести к дальнейшим ошибкам при решении задач.

Вложенные конструкции CASE

Когда прототип таблицы был готов, его показали заказчику. Он выбрал логику формирования поля CUSTOMER_TYPE_2, но уточнил некоторые условия расчета сегментов:

Сумма дохода	Сумма средней транзакции	Тип клиента
<20 тыс.	любая	Стандарт
20–100 тыс.	< 2 тыс.	Стандарт
20–100 тыс.	2–5 тыс.	Премиум
20–100 тыс.	5–10 тыс	VIP
20–100 тыс.	>10 тыс.	VIP Нестандарт
100–500 тыс.	любая	VIP
> 500 тыс.	любая	VIP Нестандарт

По новым требованиям клиент с доходом от 20–40 тыс. может быть отнесен к одному из четырех сегментов в зависимости от суммы среднемесячной транзакции. Конечно, Антон может написать запрос, в котором для каждого условия будет своя логическая ветка, но тогда в нем будут четыре ветки с одинаковым условием на сумму дохода. Поэтому Антон решает разработать новую логику с вложенной конструкцией **CASE**. Для этого он в ветку проверки принадлежности дохода диапазону 20–40 тыс. вставляет еще одну конструкцию **CASE** с проверкой размера средней суммы транзакции. Вот результат:

```
(Язык — SQL)
SELECT
    last_name,
    first_name,
    trans_amt_avg,
    income_amt,
CASE
        WHEN income_amt < 20000 THEN 'Стандарт'
        WHEN income_amt>= 20000 AND  income_amt<100000 THEN
         CASE
                WHEN trans_amt_avg<2000 THEN 'Стандарт'
                WHEN  trans_amt_avg>=2000  AND  trans_amt_avg<5000
                     THEN 'Премиум'
                WHEN trans_amt_avg>= 5000 AND trans_amt_avg<10000
                     THEN 'VIP'
                WHEN trans_amt_avg>= 10000 THEN 'VIP Нестандарт'
         END
WHEN income_amt>= 100000 AND income_amt < 500000 THEN 'VIP'
WHEN income_amt>= 500000 THEN 'VIP Нестандарт'
 ELSE 'Сегмент не установлен'
    END as customer_type
FROM customers
```

Тогда скрипт вернет такой результат:

LAST_NAME	FIRST_NAME	TRANS_AMT_AVG	INCOME_AMT	CUSTOMER_TYPE
Иванова	Екатерина	1512	20 170	Стандарт
Сидорян	Валентин	12 951	612 300	VIP Нестандарт
Орехова	Ольга	15 700	56 820	~~Премиум~~ VIP Нестандарт
Гуров	Сергей	8610	184 030	VIP
Савинов	Василий	2472	38 240	~~Стандарт~~ Премиум

Ну, вроде заказчик удовлетворен, можно и отдохнуть.

Функция COALESCE

Вспомним пример с определением возрастной группы. Когда Антон обнаружил, что возраст указан не во всех полях, коллеги подсказали ему, что значение NULL на самом деле соответствует 45 годам. Поэтому для правильного расчета возрастной группы необходимо NULL заменить на 45. Для этого также надо использовать конструкцию CASE, но только теперь, чтобы сформировать поле AGE.

```
(Язык — SQL)
SELECT
    name,
    CASE
```

```
        WHEN age is NULL THEN 45 –если AGE принимает значение NULL, то заменяем на 45
        ELSE age–иначе  оставляем прежнее значение
    END as age
FROM person
```

На практике часто нужно заменить NULL каким-либо другим значением, это может быть как константа, так и значение другого поля в этой же строке. Чтобы каждый раз не писать конструкцию CASE, существует специальная функция

COALESCE(текущее_поле, значение_1, значение_2, …).

У нее есть несколько аргументов:

- текущее_поле — наименование атрибута, в котором происходит поиск значения NULL, является обязательным аргументом;
- значение_1 — значение, которое примет этот атрибут, если текущее_поле соответствует NULL, является обязательным аргументом;
- значение_2 — значение, которое примет атрибут, если текущее_поле и значение_1 являются NULL. Все аргументы, начиная со значение_2, не являются обязательными.

Функция **COALESCE** проверяет заполняемость атрибута:

- если атрибут непустой, его значение не меняется;
- если поле принимает NULL-значение, функция заменяет его на первый не NULL-аргумент;
- если все аргументы функции являются NULL, атрибут также будет заполнен NULL-значением.

А теперь давайте перепишем запрос с помощью функции **COALESCE**:

(Язык — SQL)

```
SELECT
    name,
    COALESCE(age, 45) as age
FROM person
```

Получается значительно короче!

Итак, в этом блоке вы узнали о новой функции **COALESCE**, которая значительно упрощает процесс поиска и замены NULL-значений.