Datové a procesní modely

[Relační databáze]

Přednáška 3d

Marian Kamenický

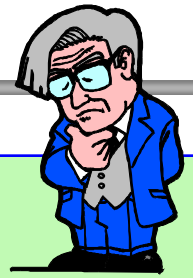
Syntea software group a.s.
marian.kamenicky@syntea.cz

MFFUK Praha

2019/20

a zas něco caseovního

Algoritmizace dotazu a ostatních povelů



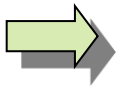
- bylo řečeno
- **jazyk SQL mocný to neprocedurální jazyk**

- umíme triviálně naplnit tabulku
- trochu jí přepsat
- zrušit data v tabulce
- vypsat data z tabulky
- vypsat data z více tabulek spojením

- nepřipadá mi to
- jako **výkonný aplikační prostředek aplikace**

Algoritmizace dotazu [úkol]

Zakaznici	
Firma	MaxUver
Alfa	700 000
Beta sro	20 000
BMV	1000 000



Firma	MaxUver	Trida
Alfa	700 000	B
Beta sro	20 000	D
BMV	1000 000	A

- výši úvěru označte jeho třídou
- dle rozpětí

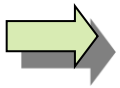
- řešte

Definice třídy úvěru			
Výše	Od	Do	Třída
> 900 000	900 001	∞	A
> 500 000	500 001	900 000	B
> 100 000	100 001	500 000	C
<= 100 000	1	100 000	D

- pane učiteli, to nejde, dyť to je algoritmus
- to se bude muset naprogramovat ňákejma IF-ama
- v ňákym programovacim jazyku
- sakra, to bude ale dřina !!!!
- do vánoc je co dělat

Algoritmizace dotazu

Zakaznici	
Firma	MaxUver
Alfa	700 000
Beta sro	20 000
BMV	1000 000



Firma	MaxUver	Trida
Alfa	700 000	B
Beta sro	20 000	D
BMV	1000 000	A

- výši úvěru označte jeho třídou
- dle rozpětí

- řešte

Definice třídy úvěru			
Výše	Od	Do	Třída
> 900 000	900 001	∞	A
> 500 000	500 001	900 000	B
> 100 000	100 001	500 000	C
<= 100 000	1	100 000	D

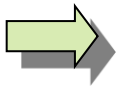
- umíte vybrat řádky s třídou A ? -- vyrobte je !!!

```
Select  Firma, MaxUver, 'A' Trida
From    Zakaznici
Where MaxUver > 900000;
```

Firma	MaxUver	Trida
BMV	1000 000	A

Algoritmizace dotazu

Zakaznici	
Firma	MaxUver
Alfa	700 000
Beta sro	20 000
BMV	1000 000



Firma	MaxUver	Trida
Alfa	700 000	B
Beta sro	20 000	D
BMV	1000 000	A

- výši úvěru označte jeho třídou
- dle rozpětí

Definice třídy úvěru			
Výše	Od	Do	Třída
> 900 000	900 001	∞	A
> 500 000	500 001	900 000	B
> 100 000	100 001	500 000	C
<= 100 000	1	100 000	D

- řešte

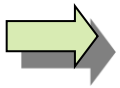
- umíte vybrat řádky s třídou **B** ? -- vyrobte je !!!

```
Select  Firma, MaxUver, 'B' Trida
From    Zakaznici
Where MaxUver > 500000
   And  MaxUver <= 900000;
```

Firma	MaxUver	Trida
Alfa	700 000	B

Algoritmizace dotazu

Zakaznici	
Firma	MaxUver
Alfa	700 000
Beta sro	20 000
BMV	1000 000



Firma	MaxUver	Trida
Alfa	700 000	B
Beta sro	20 000	D
BMV	1000 000	A

- výši úvěru označte jeho třídou
- dle rozpětí

- řešte

Definice třídy úvěru			
Výše	Od	Do	Třída
> 900 000	900 001	∞	A
> 500 000	500 001	900 000	B
> 100 000	100 001	500 000	C
<= 100 000	1	100 000	D

- umíte vybrat řádky s třídou **D** ? -- vyrobte je !!!

```
Select  Firma, MaxUver, 'D' Trida
From    Zakaznici
Where MaxUver <= 100000;
```

Firma	MaxUver	Trida
Beta sro	20 000	D

Algoritmizace dotazu

Definice třídy úvěru			
Výše	Od	Do	Třída
> 900 000	900 001	∞	A
> 500 000	500 001	900 000	B
> 100 000	100 001	500 000	C
<= 100 000	1	100 000	D

```
Select  Firma, MaxUver, 'A' Trida
From    Zakaznici
Where MaxUver > 900000;
```

Firma	MaxUver	Trida
Alfa	700 000	B
Beta sro	20 000	D
BMV	1000 000	A

```
Select  Firma, MaxUver, 'B' Trida
From    Zakaznici
Where MaxUver >   500000
   And  MaxUver <= 900000;
```

Firma	MaxUver	Trida
BMV	1000 000	A

```
Select  Firma, MaxUver, 'C' Trida
From    Zakaznici
Where MaxUver >   100000
   And  MaxUver <= 500000;
```

Firma	MaxUver	Trida
Alfa	700 000	B

```
Select  Firma, MaxUver, 'D' Trida
From    Zakaznici
Where MaxUver <= 100000;
```

Firma	MaxUver	Trida
Beta sro	20 000	D

Algoritmizace dotazu

Definice třídy úvěru			
Výše	Od	Do	Třída
> 900 000	900 001	∞	A
> 500 000	500 001	900 000	B
> 100 000	100 001	500 000	C
<= 100 000	1	100 000	D

Firma	MaxUver	Trida
Alfa	700 000	B
Beta sro	20 000	D
BMV	1000 000	A

```
Select  Firma, MaxUver, 'A' Trida
From    Zakaznici
Where MaxUver > 900000        UNION
Select  Firma, MaxUver, 'B' Trida
From    Zakaznici
Where MaxUver >   500000
   And  MaxUver <= 900000     UNION
Select  Firma, MaxUver, 'C' Trida
From    Zakaznici
Where MaxUver >   100000
   And  MaxUver <= 500000     UNION
Select  Firma, MaxUver, 'D' Trida
From    Zakaznici
Where MaxUver <= 100000;
```

Už to máme

ale je to **pěkná dřina**

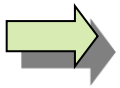
Firma	MaxUver	Trida
Alfa	700 000	B
Beta sro	20 000	D
BMV	1000 000	A

a
radši

zpět k problému

Algoritmizace dotazu

Zakaznici	
Firma	MaxUver
Alfa	700 000
Beta sro	20 000
BMV	1000 000



Firma	MaxUver	Trida
Alfa	700 000	B
Beta sro	20 000	D
BMV	1000 000	A

- výši úvěru označte jeho třídou
- dle rozpětí

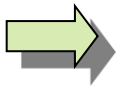
Definice třídy úvěru			
Výše	Od	Do	Třída
> 900 000	900 001	∞	A
> 500 000	500 001	900 000	B
> 100 000	100 001	500 000	C
<= 100 000	1	100 000	D

- řešte

- SQL - neprocedurální jazyk
- nelze několika povely vytvářet algoritmus
- povely jsou na sobě [procesně] nezávislé
- SQL má však v sobě bohatou algoritmickou funkcionalitu
- SQL umožňuje zadat algoritmizaci dovnitř jednoho povelu

Algoritmizace dotazu

Zakaznici	
Firma	MaxUver
Alfa	700 000
Beta sro	20 000
BMV	1000 000



Firma	MaxUver	Trida
Alfa	700 000	B
Beta sro	20 000	D
BMV	1000 000	A

- výši úvěru označte jeho třídou
-

```
IF        Uver >900 000 Then ... ... ...
ELSE IF Uver >500 000 Then ... ... ...
ELSE IF Uver >100 000 Then ... ... ...
ELSE                                 .... .... ...;
Do While ....
    ......
End While;
IF ...
END IF
```

Definice třídy úvěru			
Výše	Od	Do	Třída
> 900 000	900 001	∞	A
> 500 000	500 001	900 000	B
> 100 000	100 001	500 000	C
<= 100 000	1	100 000	D

- SQL - neprocedurální jazyk
- n
-
- S litu
- SQL umožňuje zadat algoritmizaci dovnitř jednoho povelu

SQL povel .. ;

Jak jsou definovány povely SQL


```
Select      Sloupec1 [ , Sloupec2 , Sloupec 3, … ]
From        Tab1
                    [Join  Tab2     ON  Výraz-Podmínka    … ]
Where       Podmínka-Výraz
Group By    Sloupec1  [ , Sloupec2, Sloupec3  … ]
Having      Podmínka-Výraz
Order By    Sloupec1  [, Sloupec2 , Sloupec 3, … ]
```

Jak jsou definovány povely SQL


```
Select      Sloupec1 [ , Sloupec2 , Sloupec 3, … ]
From        Tab1
                    [Join  Tab2     ON  Výraz-Podmínka  … ]
Where       Podmínka-Výraz
Group By    Sloupec1  [ , Sloupec2, Sloupec3  … ]
Having      Podmínka-Výraz
Order By    Sloupec1  [, Sloupec2 , Sloupec 3, … ]
```

A toť vše

Jak jsou definovány povely SQL

Select **Výraz1** [, **Výraz2** , **Výraz3**, …]

From Tab1

[Join Tab2 ON **Výraz-Podmínka** …]

Where **Podmínka-Výraz**

Group By **Výraz1** [, **Výraz2**, **Výraz3** …]

Having **Podmínka-Výraz**

Order By **Výraz1** [, **Výraz2** , **Výraz3**, …]

Podmíněný operátor Case

- operátor CASE [neboli podmíněný výraz] umožňuje
- provádět jednoduchá rozhodnutí
  - na úrovni VÝRAZU
- struktura operátoru je podobná konstrukci

Podmíněný operátor Case


```
CASE    WHEN  (a>b)     THEN   nějaká hodnota
        WHEN  (a>c)     THEN   nějaká hodnota
        WHEN  (a+b>c)   THEN   nějaká hodnota
        ELSE                   nějaká hodnota
END
```

varianta 1

To není algoritmus průběhu

To je algoritmus výrazu

Algoritmus

vyrob buď to

nebo vyrob něco jiného

anebo úplně něo jiného

Je to **algoritmus**

vyhodnocení výrazu

Podmíněný operátor Case


```
CASE    WHEN  (a>b)      THEN   nějaká hodnota
        WHEN  (a>c)      THEN   nějaká hodnota
        WHEN  (a+b>c)    THEN   nějaká hodnota
        ELSE                    nějaká hodnota
END
```

varianta 1

```
CASE a  WHEN  (2)        THEN   ...
        WHEN  (3)        THEN   ...
        WHEN  (88)       THEN   ...
        ELSE                    ...
END
```

varianta 2

```
CASE WHEN  (a=2)  THEN   ...
     WHEN  (a=3)  THEN   ...
     WHEN  (a=88) THEN   ...
     ELSE                ...
END
```

varianta 1

Podmíněný operátor Case

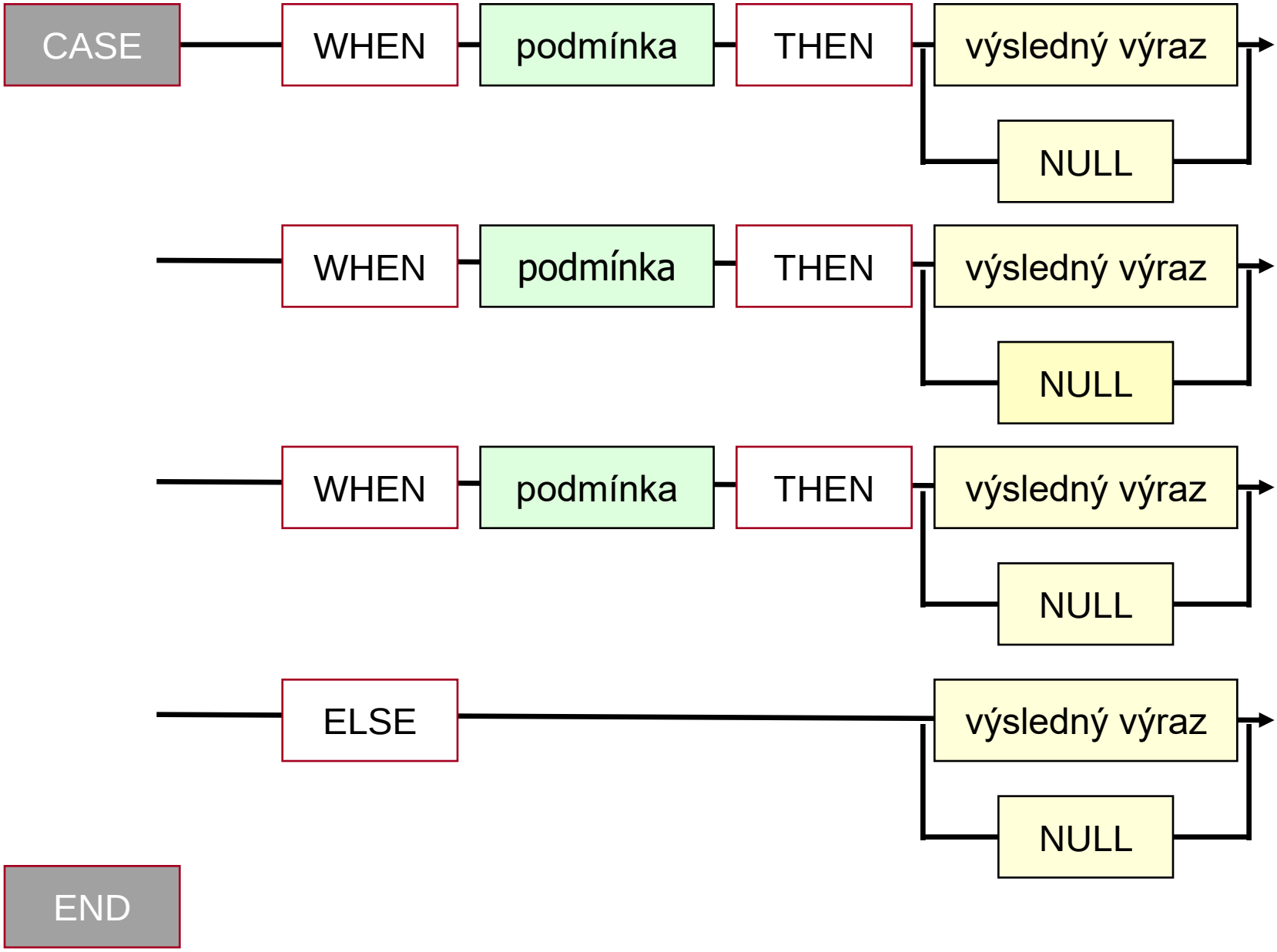

```
CASE WHEN  (Length(Jmeno) + Length(Prijmeni) = 10)  THEN ...
       WHEN  (Length(Jmeno) + Length(Prijmeni) = 11)  THEN ...
       WHEN  (Length(Jmeno) + Length(Prijmeni) = 12)  THEN ...
       ELSE
END
```

varianta 1

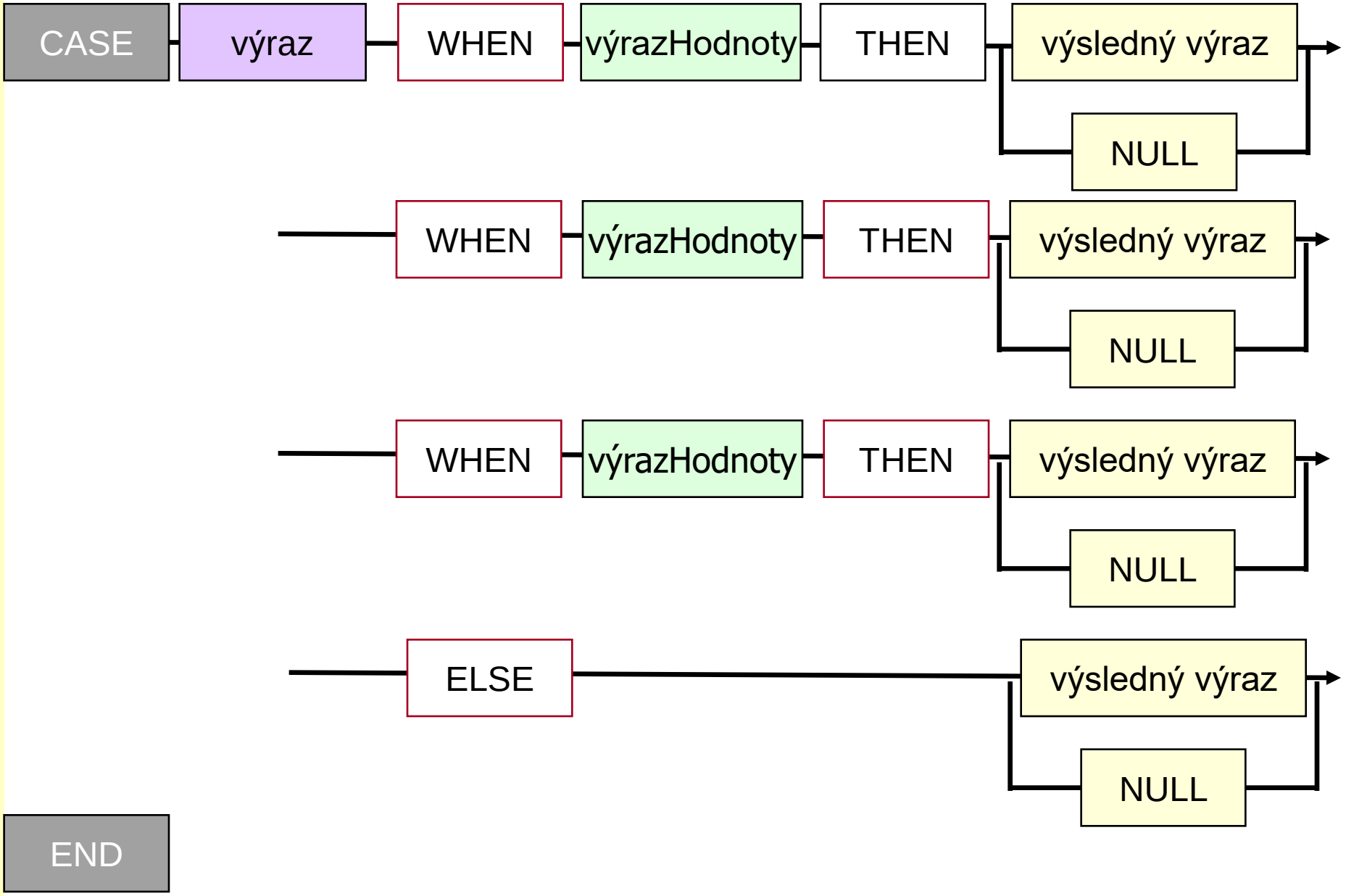
```
CASE  Length(Jmeno) + Length(Prijmeni)  WHEN  (10)  THEN ...
                                        WHEN  (11)  THEN ...
                                        WHEN  (12)  THEN ...
                                        ELSE
END
```

varianta 2

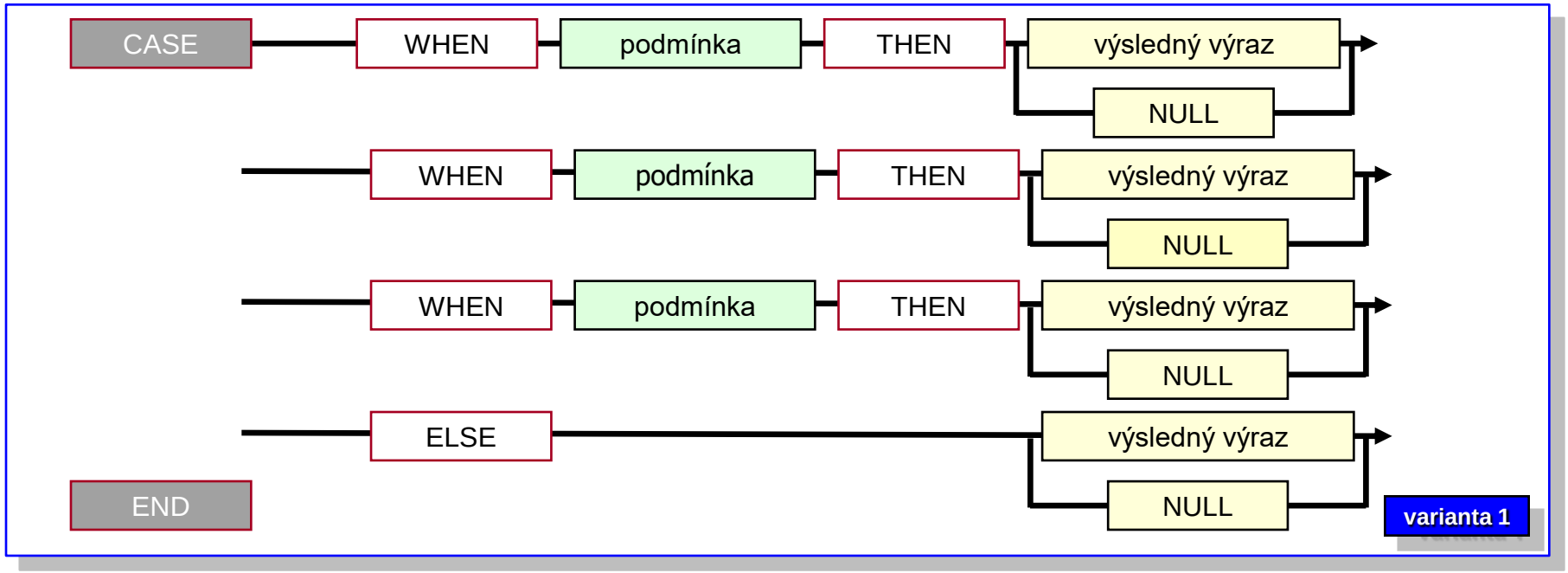
Podmíněný operátor Case - varianta 1

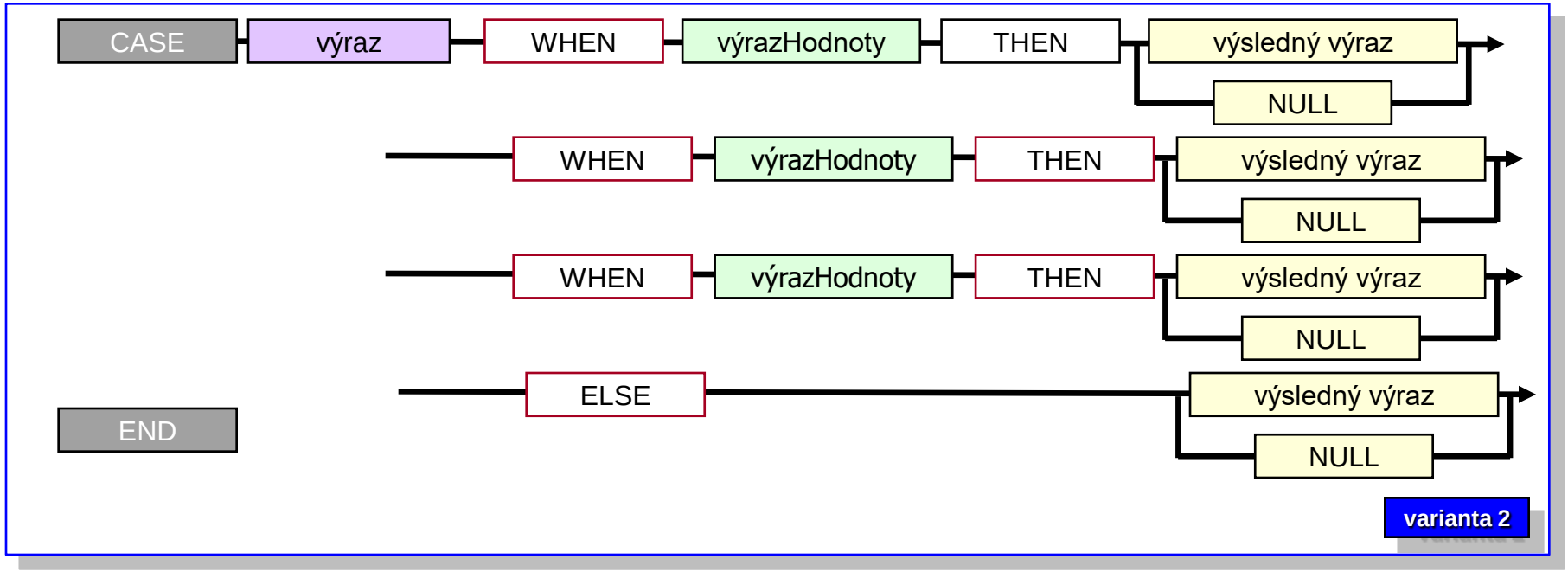


Podmíněný operátor Case - varianta 2

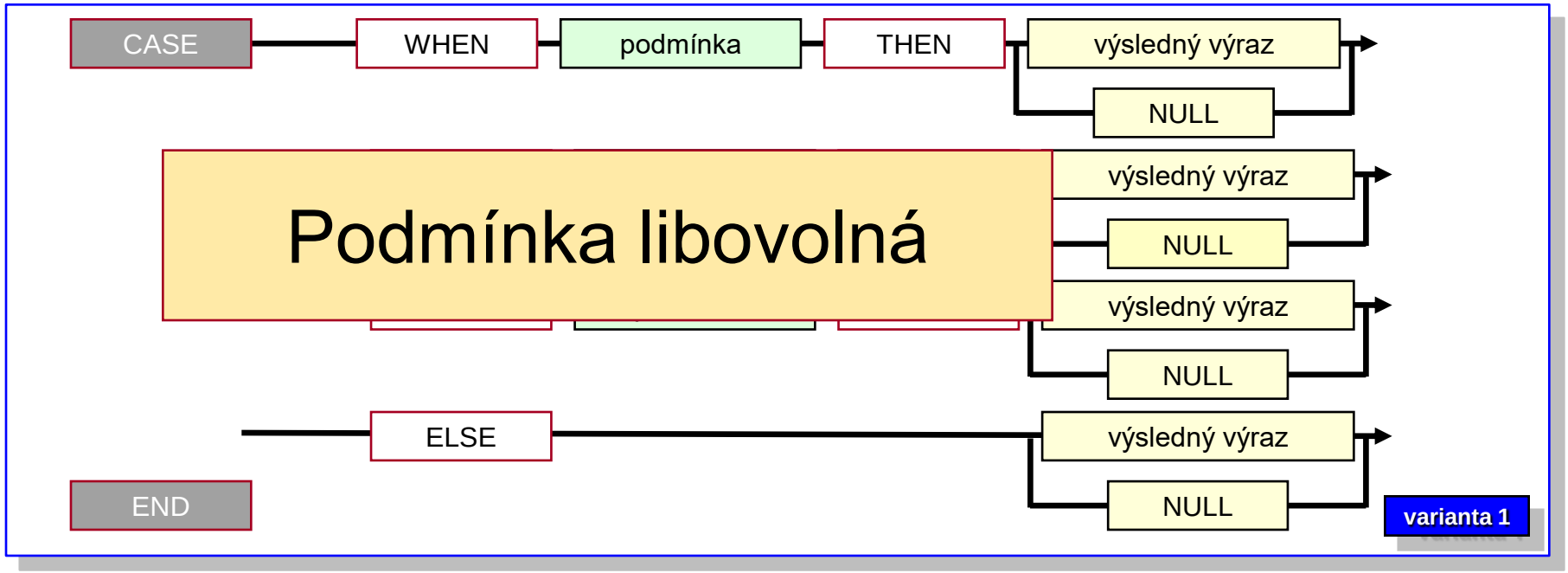


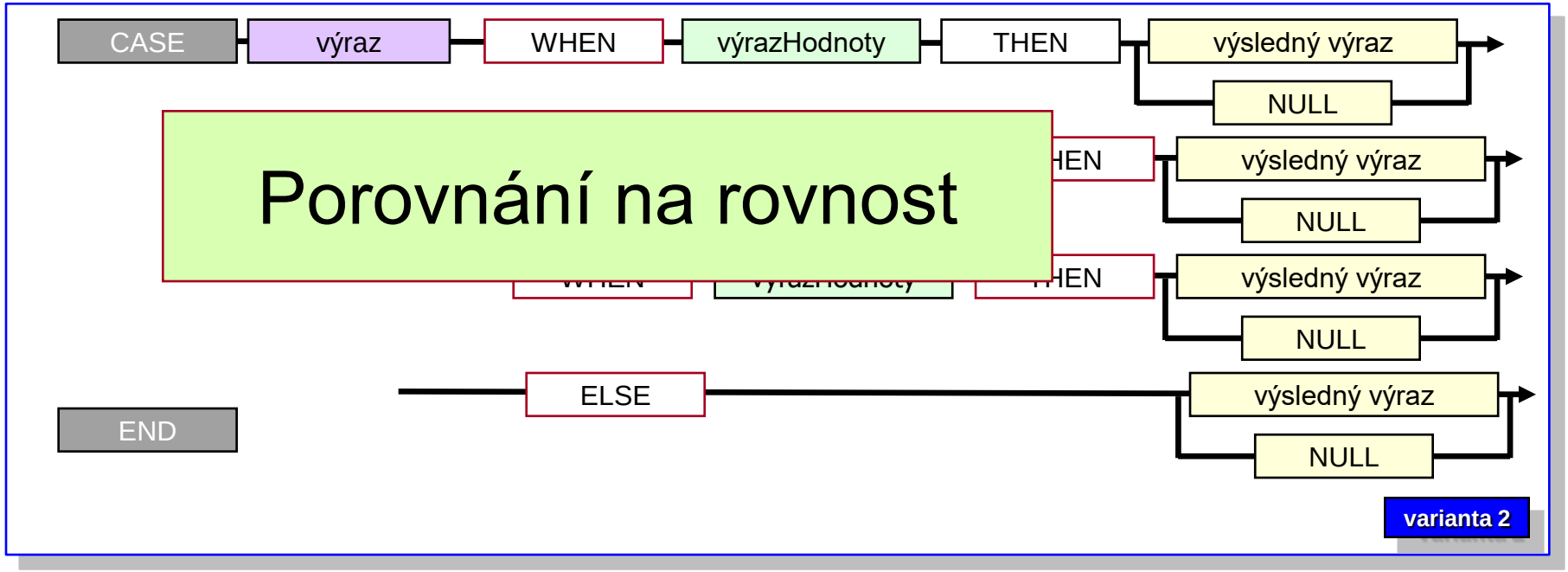
Podmíněný operátor Case





Podmíněný operátor Case





Podmíněný operátor Case

- CASE je operátor
- vrací skalární hodnotu
- komplikovaný algoritmus
  - vracející hodnotu

Podmíněný operátor Case

Zakaznici	
Firma	MaxUver
Alfa	700 000
Beta sro	20 000
BMV	1000 000

řešte

Firma	MaxUver	Trida
Alfa	700 000	B
Beta sro	20 000	D
BMV	1000 000	A

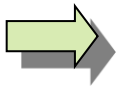
SELECT Firma, MaxUver ,

a 3. sloupec si "vyrobíme" sami výrazem
buď 'A' nebo 'B' nebo cokoliv jiného
podmíněných operátorem CASE
dle hodnoty ve sloupci MaxUver

FROM Zakaznici

Podmíněný operátor Case

Zakaznici	
Firma	MaxUver
Alfa	700 000
Beta sro	20 000
BMV	1000 000



řešte

Firma	MaxUver	Trida
Alfa	700 000	B
Beta sro	20 000	D
BMV	1000 000	A

```
SELECT  Firma, MaxUver ,
        CASE   WHEN (MaxUver>900000)     THEN  'A'
               WHEN (MaxUver>500000)     THEN 'B'
               WHEN (MaxUver>100000)     THEN 'C'
               ELSE                            'D'
        END
  FROM  Zakaznici
```

Podmíněný operátor Case

Zakaznici	
Firma	MaxUver
Alfa	700
Beta sro	20
BMV	1000

- pane učiteli
- to nemáte dobře

Uver	Trida
700 000	B
20 000	D
1000 000	A

```
SELECT  Firma, MaxUver ,
        CASE   WHEN (MaxUver>900000)     THEN  'A'
               WHEN (MaxUver>500000)     THEN 'B'
               WHEN (MaxUver>100000)     THEN 'C'
               ELSE                            'D'
         END
  FROM  Zakaz
```

- a cotak můžu dobře
- nemít copak ???

Podmíněný operátor Case

Zakaznici	
Firma	MaxUver
Alfa	700
Beta sro	20
BMV	1000

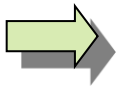
- pane učiteli
- to nemáte dobře
- protože

	axUver	Trida
	700 000	B
	20 000	D
	1000 000	A

```
SELECT  Firma, MaxUver ,
        CASE   WHEN (MaxUver>900000)     THEN  'A'
               WHEN (MaxUver>500000)     THEN 'B'
               WHEN (MaxUver>100000)     THEN 'C'
               ELSE                            'D'
        END
  FROM  Zakaznici
```

Podmíněný operátor Case

Zakaznici	
Firma	MaxUver
Alfa	700 000
Beta sro	20 000
BMV	1000 000



řešte

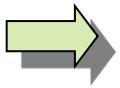
Firma	MaxUver	Trida
Alfa	700 000	B
Beta sro	20 000	D
BMV	1000 000	A

```
SELECT  Firma, MaxUver ,
        CASE   WHEN (MaxUver>900000)     THEN  'A'
               WHEN (MaxUver>500000)     THEN 'B'
               WHEN (MaxUver>100000)     THEN 'C'
               ELSE                           'D'
        END    as Trida
  FROM  Zakaznici
```

Firma	MaxUver	Trida
Alfa	700 000	B
Beta sro	20 000	D
BMV	1000 000	A

Podmíněný operátor Case

ToAuto	
aId	iBarva
1	1
2	2
3	3
4	1
5	10



řešte

aId	iBarva	Nazev
1	1	zelena
2	2	modra
3	3	bila
4	1	zelena
5	10	====

```
SELECT      aid,iBarva,
            CASE (iBarva)
              WHEN (1)  THEN 'zelena'
              WHEN  (2)   THEN 'modra'
              WHEN  (3)   THEN 'bila'
               ELSE                '==='
            END CASE
   FROM    ToAuto
```

Podmíněný operátor Case

ToAuto	
aId	iBarva
1	1
2	2
3	3
4	1
5	10

- pane učiteli
- máte tam 2 chyby

no proto

aId	iBarva	Nazev
1	1	zelena
2	2	modra
3	3	bila
4	1	zelena
5	10	====

```
SELECT      aid,iBa
            CASE (iBarva)
              WHEN (1)  THEN 'zelena'
              WHEN  (2)   THEN 'modra'
              WHEN  (3)   THEN 'bila'
               ELSE              '==='
            END CASE     AS Nazev
   FROM     ToAuto
```

error

Podmíněný operátor Case

ToAuto	
aId	iBarva
1	1
2	2
3	3
4	1
5	10

- není operátor CASE
- ani žádný podobný
- vyřešte úkol
- bez použití CASE
- SELECT vyrobí totéž

aId	iBarva	Nazev
1	1	zelena
2	2	modra
3	3	bila
4	1	zelena
5	10	====

- řešte

```
SELECT  aid,iBarva, 'zelena' From ToAuto Where iBarva=1

SELECT  aid,iBarva, 'modra' From ToAuto Where iBarva=2

SELECT  aid,iBarva, 'bila'     From ToAuto Where iBarva=3

SELECT  aid,iBarva, '==='    From ToAuto Where iBarva not in (1,2,3)
```

aId	iBarva	Nazev
1	1	zelena

aId	iBarva	Nazev
2	2	modra

aId	iBarva	Nazev
3	3	bila

aId	iBarva	Nazev
5	10	====

Podmíněný operátor Case

ToAuto	
aId	iBarva
1	1
2	2
3	3
4	1
5	10

- není operátor CASE
- ani žádný podobný
- vyřešte úkol
- bez použití CASE
- SELECT vyrobí totéž

aId	iBarva	Nazev
1	1	zelena
2	2	modra
3	3	bila
4	1	zelena
5	10	====

- řešte

```
SELECT  aid,iBarva, 'zelena' From ToAuto Where iBarva=1
 UNION
SELECT  aid,iBarva, 'modra' From ToAuto Where iBarva=2
 UNION
SELECT  aid,iBarva, 'bila'     From ToAuto Where iBarva=3
 UNION
SELECT  aid,iBarva, '==='   From ToAuto Where iBarva not in (1,2,3)
ORDER BY aid;
```

aId	iBarva	Nazev
1	1	zelena

aId	iBarva	Nazev
2	2	modra

aId	iBarva	Nazev
3	3	bila

aId	iBarva	Nazev
5	10	====

Podmíněný operátor Case

ToAuto	
aId	iBarva
1	1
2	2
3	3
4	1
5	10

- není operátor CASE
- ani žádný podobný
- vyřešte úkol
- bez použití CASE
- SELECT vyrobí totéž

aId	iBarva	Nazev
1	1	zelena
2	2	modra
3	3	bila
4	1	zelena
5	10	====

- řešte

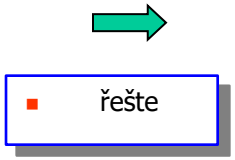
```
SELECT  aid,iBarva, 'zelena' From ToAuto Where iBarva=1
 UNION
SELECT  aid,iBarva, 'modra' From ToAuto Where iBarva=2
 UNION
SELECT  aid,iBarva, 'bila'     From ToAuto Where iBarva=3
 UNION
SELECT  aid,iBarva, '==='   From ToAuto Where iBarva not in (1,2,3)
ORDER BY aid;
```

aId	**iBarva**	**Nazev**
1	**1**	**zelena**
2	**2**	**modra**
3	**3**	**bila**
4	**1**	**zelena**
5	**10**	**====**

a zas něco jiného

Coalesce - varianta Case

Prodejci		
Jmeno	PlanProdeje	Prodej
Hanka	2000	3500
Pavel	NULL	1500
Marie	NULL	NULL



Jmeno	
Hanka	2000
Pavel	1500
Marie	0

```
SELECT   Jmeno, PlanProdeje        as Cosi
   From   Prodejci
   Where  PlanProdeje Is Not Null
```

Jmeno	
Hanka	2000

```
SELECT   Jmeno, Prodej
 From     Prodejci
 Where    PlanProdeje Is     Null
   And    Prodej       Is Not Null
```

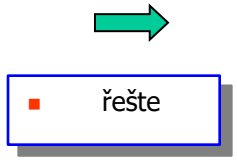
Jmeno	
Pavel	1500

```
SELECT   Jmeno, 0
 From     Prodejci
 Where    PlanProdeje Is Null
   And    Prodej       Is Null
```

Jmeno	
Marie	0

Coalesce - varianta Case

Prodejci		
Jmeno	PlanProdeje	Prodej
Hanka	2000	3500
Pavel	NULL	1500
Marie	NULL	NULL



Jmeno	
Hanka	2000
Pavel	1500
Marie	0

```
SELECT   Jmeno, PlanProdeje        as Cosi
   From   Prodejci
   Where  PlanProdeje Is Not Null
 UNION
SELECT   Jmeno, Prodej
 From     Prodejci
 Where    PlanProdeje Is     Null
   And    Prodej       Is Not Null
 UNION
SELECT   Jmeno, 0
 From     Prodejci
 Where    PlanProdeje Is Null
   And    Prodej       Is Null
```

Jmeno	
Hanka	2000
Pavel	1500
Marie	0

Coalesce - varianta Case

Prodejci		
Jmeno	PlanProdeje	Prodej
Hanka	2000	3500
Pavel	NULL	1500
Marie	NULL	NULL

- řešte

Jmeno	
Hanka	2000
Pavel	1500
Marie	0

```
SELECT   Jmeno,
   CASE
         WHEN PlanProdeje  IS NOT NULL   THEN   PlanProdeje
         WHEN Prodej       IS NOT NULL   THEN   Prodej
         ELSE                                   0
   END   AS Cosi
  FROM    PRODEJCI;
```

- máme seznam zdrojových hodnot
- [PlanProdeje, Prodej, 0]
- ze seznamu chceme zobrazit
- první známou [NOT NULL] hodnotu

Coalesce - varianta Case

Prodejci		
Jmeno	PlanProdeje	Prodej
Hanka	2000	3500
Pavel	NULL	1500
Marie	NULL	NULL

Jmeno	
Hanka	2000
Pavel	1500
Marie	0

```
SELECT   Jmeno,
   CASE
         WHEN PlanProdeje  IS NOT NULL   THEN   PlanProdeje
         WHEN Prodej       IS NOT NULL   THEN   Prodej
         ELSE                                   0
   END   AS Cosi
  FROM    PRODEJCI;
```

- toto CASE lze zapsat
  - jednoduseji
- pomocí speciální varianty operátoru CASE
  - **COALESCE**

Coalesce - varianta Case

Prodejci		
Jmeno	PlanProdeje	Prodej
Hanka	2000	3500
Pavel	NULL	1500
Marie	NULL	NULL

Jmeno	
Hanka	2000
Pavel	1500
Marie	0

```
SELECT   Jmeno,
   CASE
         WHEN PlanProdeje  IS NOT NULL   THEN   PlanProdeje
         WHEN Prodej       IS NOT NULL   THEN   Prodej
         ELSE                                   0
   END   AS Cosi
  FROM    PRODEJCI;
```

```
SELECT   Jmeno,  COALESCE (PlanProdeje, Prodej ,0)
   FROM   PRODEJCI;
```

- obkreslete

```
COALESCE   (výraz1, výraz2, výraz3, ...)
```

Coalesce - varianta Case

COALESCE (výraz1, výraz2, výraz3, ...)

- v pořadí výraz1, výraz2, ...
- se vybere **první NOT NULL** výraz
- konstrukce-operátor
- pro odstranění nežádoucích NULL

```
SELECT COALESCE(NULL,1) FROM DUAL
```
→ 1

```
SELECT COALESCE(NULL,55,NULL,33)  FROM DUAL
```
→ 55

```
SELECT COALESCE(NULL, NULL, NULL)  FROM DUAL
```
→ Null

Coalesce - varianta Case

COALESCE (výraz1, výraz2, výraz3, ...)

- v pořadí výraz1, výraz2, ...
- se vybere **první NOT NULL** výraz
- kostrukce-operátor
- pro odstranění nežádoucích NULL

- coalesce
  - ↗ splynout, spojit
- coalescence
  - ↗ koalice

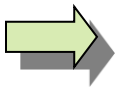
```
SELECT COALESCE(NULL,1) FROM DUAL
```
→ 1

```
SELECT COALESCE(NULL,55,NULL,33)  FROM DUAL
```
→ 55

```
SELECT COALESCE(NULL, NULL, NULL)  FROM DUAL
```
→ Null

Podmíněný operátor Case

Sarada				
S1	S2	S3	S4	S5
1	null	null	null	null
null	2	null	null	null
null	null	3	null	null
null	null	null	4	null
null	null	null	null	5



1
2
3
4
5

- v každém řádku vypište první nenulový sloupec

- řešte

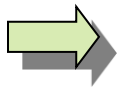
```
SELECT COALESCE   (S1, S2, S3, S4, S5)  From Sarada;
```

- úlohu řešte pomocí CASE

- řešte

Podmíněný operátor Case

Sarada				
S1	S2	S3	S4	S5
1	null	null	null	null
null	2	null	null	null
null	null	3	null	null
null	null	null	4	null
null	null	null	null	5



1
2
3
4
5

```
SELECT COALESCE   (S1, S2, S3, S4, S5)  From Sarada;
```

```
SELECT CASE
   WHEN   (S1  IS NOT NULL)     THEN   S1
   WHEN   (S2  IS NOT NULL)     THEN   S2
   WHEN   (S3  IS NOT NULL)     THEN   S3
   WHEN   (S4  IS NOT NULL)     THEN   S4
   WHEN   (S5  IS NOT NULL)     THEN   S5
   ELSE      NULL
 END
FROM Sarada;
```

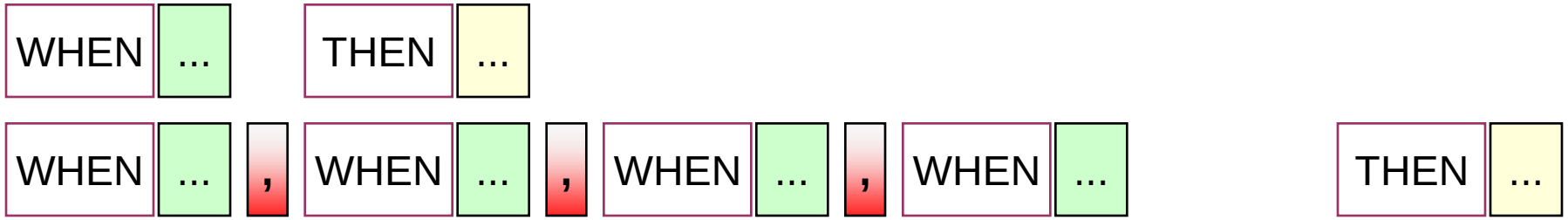
Coalesce - varianta Case


```
COALESCE    (výraz1, výraz2, výraz3,.., výrazN)
```

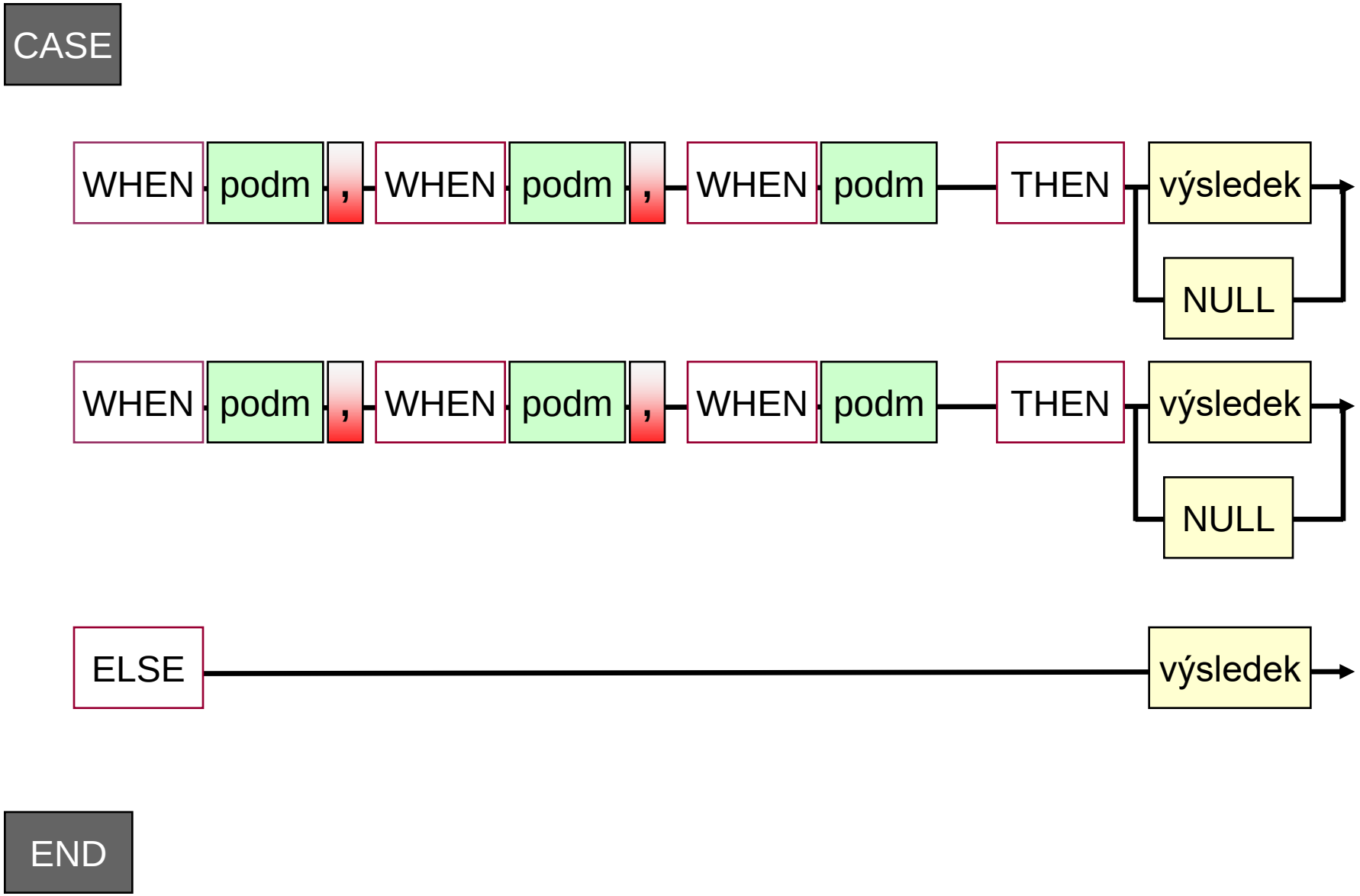
```
CASE
   WHEN   (výraz1  IS NOT NULL)     THEN   výraz1
    WHEN   (výraz2  IS NOT NULL)     THEN   výraz2
    WHEN   (výraz3  IS NOT NULL)     THEN   výraz3
    .........
    WHEN   (výrazN  IS NOT NULL)     THEN   výrazN
    ELSE      NULL
 END
```

Podmíněný operátor Case standard SQL2008

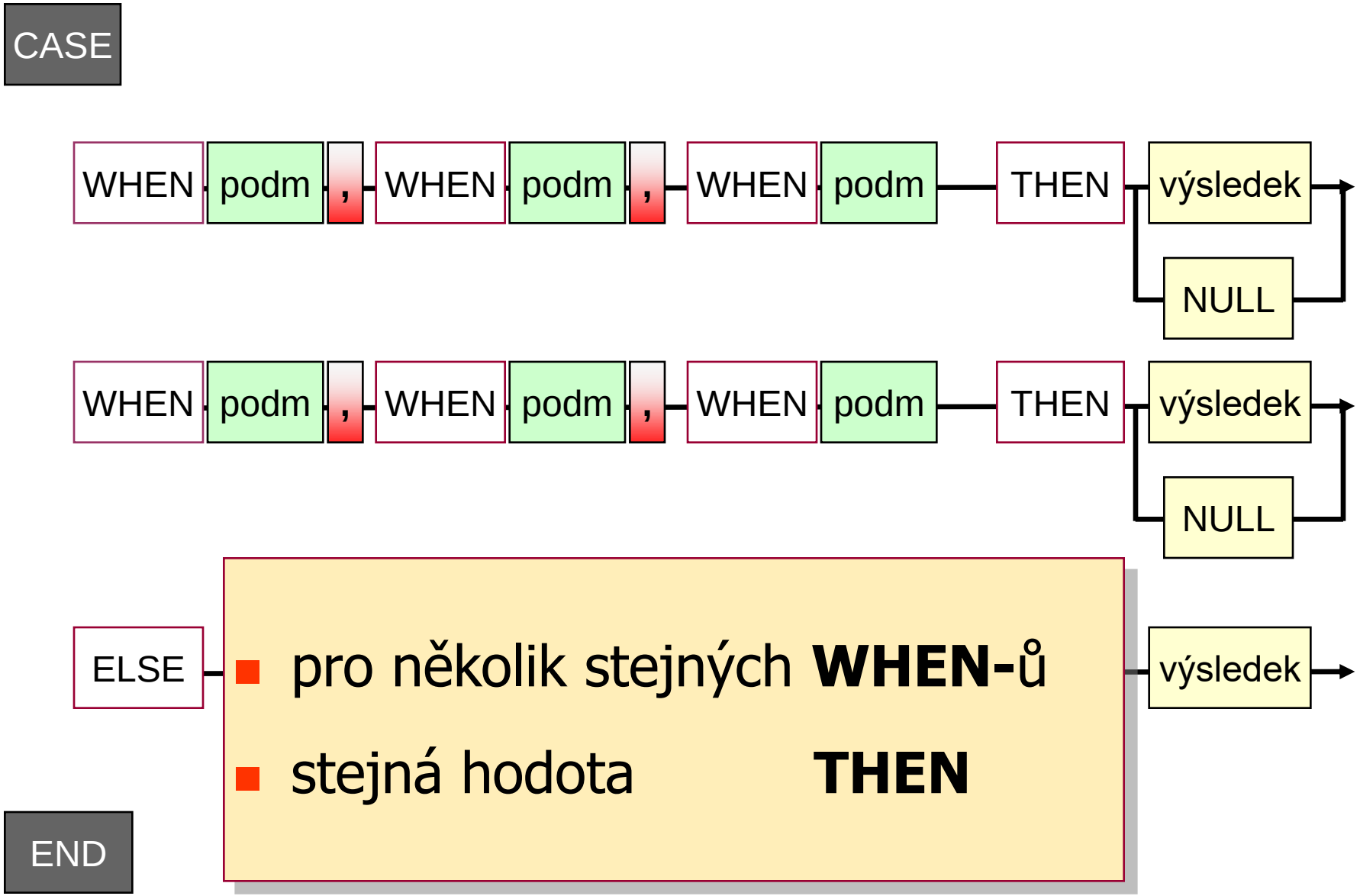
- ve standardu **SQL 2008**
- byla rozšířena funkcionalita CASE
- pro jedno THEN
  - se dá uvést
  - SEZNAM WHEN podmínek
  - či seznam WHEN hodnost
  - ne pouze jedna WHEN podmínka
  - či jedna WHEN hodnota



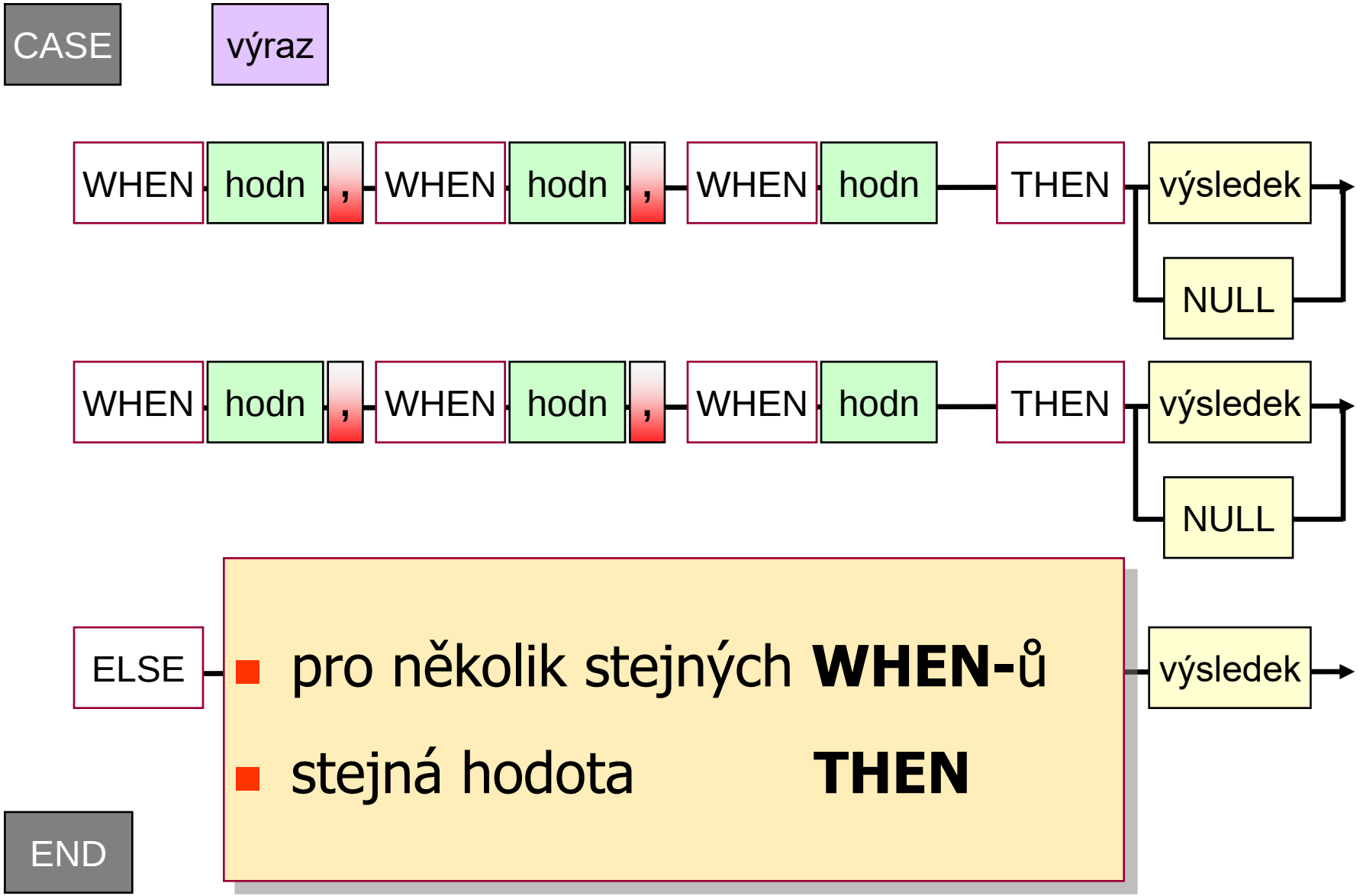
Podmíněný operátor Case standard SQL2008



Podmíněný operátor Case standard SQL2008



Podmíněný operátor Case standard SQL2008



Podmíněný operátor Case standard SQL2008


```
CASE IdBarva
   WHEN   (1) , WHEN(2) , WHEN(3)  THEN   výraz1
   WHEN   (4)                      THEN   výraz2
   ELSE                                   výraz3
 END
```

```
CASE
   WHEN   (a>b) , WHEN(c>d) , WHEN(e>0)  THEN   výraz1
   ..............
   ELSE                                         výrazN
 END
```

```
CASE
   WHEN   (a>b) OR (c>d) OR (e>0)  THEN   výraz1
   ..............
   ELSE                                   výrazN
 END
```

Užití operátou CASE - hlavní body

- potřeba konverze z data-typu na jiný datový typ
  - žádná funkce pro to neexistuje
- sumární řádky [group by] je třeba vytvářet z detailu
  - netradičně, zvláštním způsobem, podmíněně
- potřeba podmíněného procesu [větvení, IF]
  - k určení výsledku
- UNIONem se "slepují dohromady"
  - různé části jedné tabulky
- COALESCE - varianta CASE
  - pro odstranění nežádoucích hodnot NULL výsledku

Užití operátou CASE - hlavní body

- potřeba konverze z data-typu na jiný datový typ
  - žádná funkce pro to neexistuje
- sumární řádky [group by] je třeba vytvářet z detailu
  - netradičně, zvláštním způsobem, podmíněně
- potřeba podmíněného procesu [větvení, IF]
  - k určení výsledku
- UNIONem se "slepují dohromady"
  - různé části jedné tabulky
- COALESCE - varianta CASE
  - pro odstranění nežádoucích hodnot NULL výsledku

Užití operátou CASE - hlavní body

- potřeba konverze z data-typu na jiný datový typ
  - žádná funkce pro to neexistuje
- sumární řádky [group by] je třeba vytvářet z detailu
  - netradičně, zvláštním způsobem, podmíněně
  - k určení výsledku

vysvětlíme později

- UNIONem se "slepují dohromady"
  - různé části jedné tabulky
- COALESCE - varianta CASE
  - pro odstranění nežádoucích hodnot NULL výsledku

Operátor CASE

- přináší obrovskou sílu pro alikace DB
- zjednodušuje přístup [prezentace -SELECT]
- i aktualizaci [UPDATE] DB dat
  - většinou i efektivnější SELECT či UPDATE
- činí SQL zápis čtivější
- mnohdy pouhým CASE výrazem
  - lze nahradit potřebu psaní uživatelských DB rutin
- zkracuje dobu vývoje aplikace
- CASE konstrukce
  - by měla být komponentou každémo SQL vývojáře

- nepřehánět, **nezatěžovat** DB prostředí
- nevytvářet zbytečně komplikované konstrukce

trochu to zdramatizujeme

Součty

- zjistěte součty částek z tabulky

Lid			
Jm	Poh	Castka1	Castka2
Anna	z	100	1
Bobo	m	2	20
Dana	z	300	3
Emil	m	4	40

- řešte

```
Select  Sum(Castka1)   Suma1,
        Sum(Castka2)   Suma2
From    Lid;
```

Suma1	Suma2
406	64

Součty

- zjistěte **součet** částek z tabulky
- **avšak**
- pro ženy načítejte Částku1
- pro muže Částku2

Lid			
Jm	Poh	Castka1	Castka2
Anna	z	100	1
Bobo	m	2	20
Dana	z	300	3
Emil	m	4	40

- řešte

Suma
460

Suma1	Suma2
406	64

```
Select  Sum(Castka1)   Suma1
From    Lid
Where Poh='z';

Select  Sum(Castka2)   Suma2
From    Lid
Where Poh='m';
```

Suma1
400

Suma2
60

Součty

- zjistěte **součet** částek z tabulky
- avšak
  - pro ženy načítejte Částku1
  - pro muže Částku2

Lid			
Jm	Poh	Castka1	Castka2
Anna	z	100	1
Bobo	m	2	20
Dana	z	300	3
Emil	m	4	40

- řešte

Suma
460

Suma1	Suma2
406	64

```
Select  Sum(Castka1)   Suma1
From    Lid
Where Poh='z'

                Union

Select  Sum(Castka2)   Suma2
From    Lid
Where Poh='m';
```

????
460

Suma1
400
60

Součty

- zjistěte **součet** částek z tabulky
- pro ženy načítejte Částku1
- pro muže Částku2

Lid			
Jm	Poh	Castka1	Castka2
Anna	z	100	1
Bobo	m	2	20
Dana	z	300	3
Emil	m	4	40

```
Select  Sum(Suma1)
From (
Select  Sum(Castka1)   Suma1
 From   Lid
Where Poh='z'
                          Union
Select  Sum(Castka2)   Suma2
From   Lid
Where Poh='m'
) T;
```

- a je to !!!

Suma1
400

Suma2
60

Sum(Suma1)
460

Operátor Case - Coalesce v agregaci SUM

- zjistěte **součet** částek z tabulky
- avšak
- pro ženy načítejte Částku1
- pro muže Částku2

Lid			
Jm	Poh	Castka1	Castka2
Anna	z	100	1
Bobo	m	2	20
Dana	z	300	3
Emil	m	4	40

- řešte

Suma
460

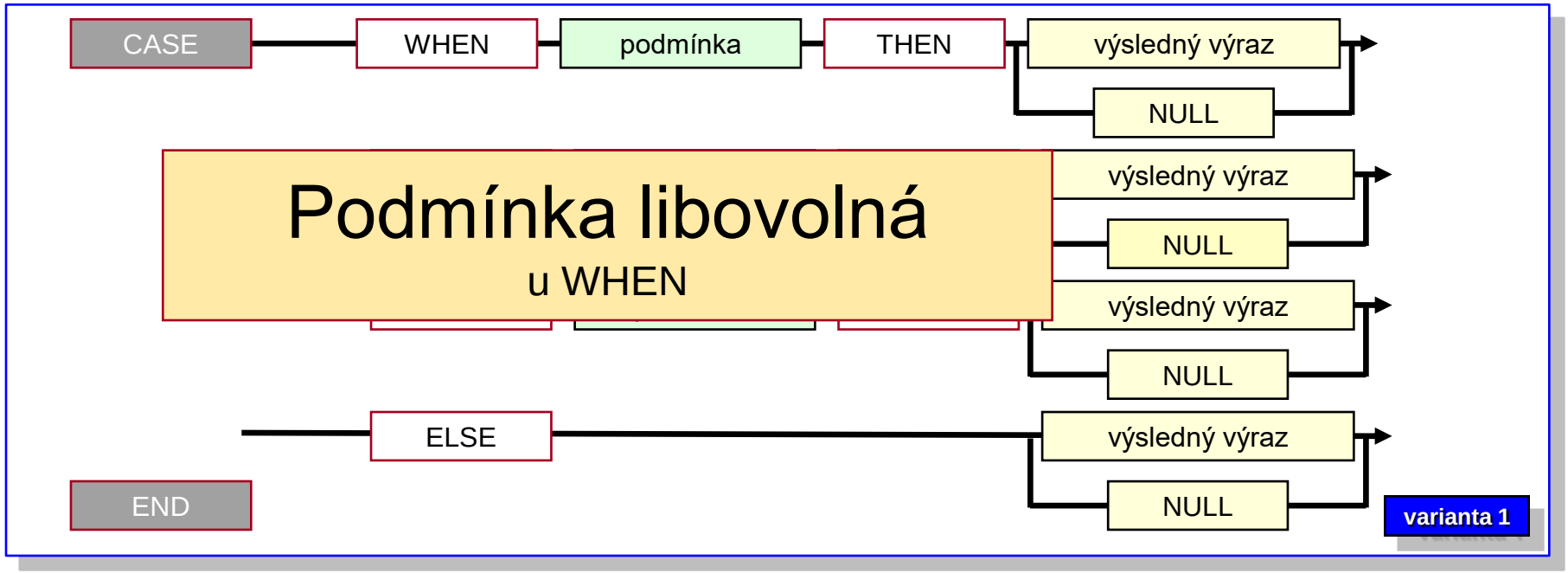
Suma1	Suma2
406	64

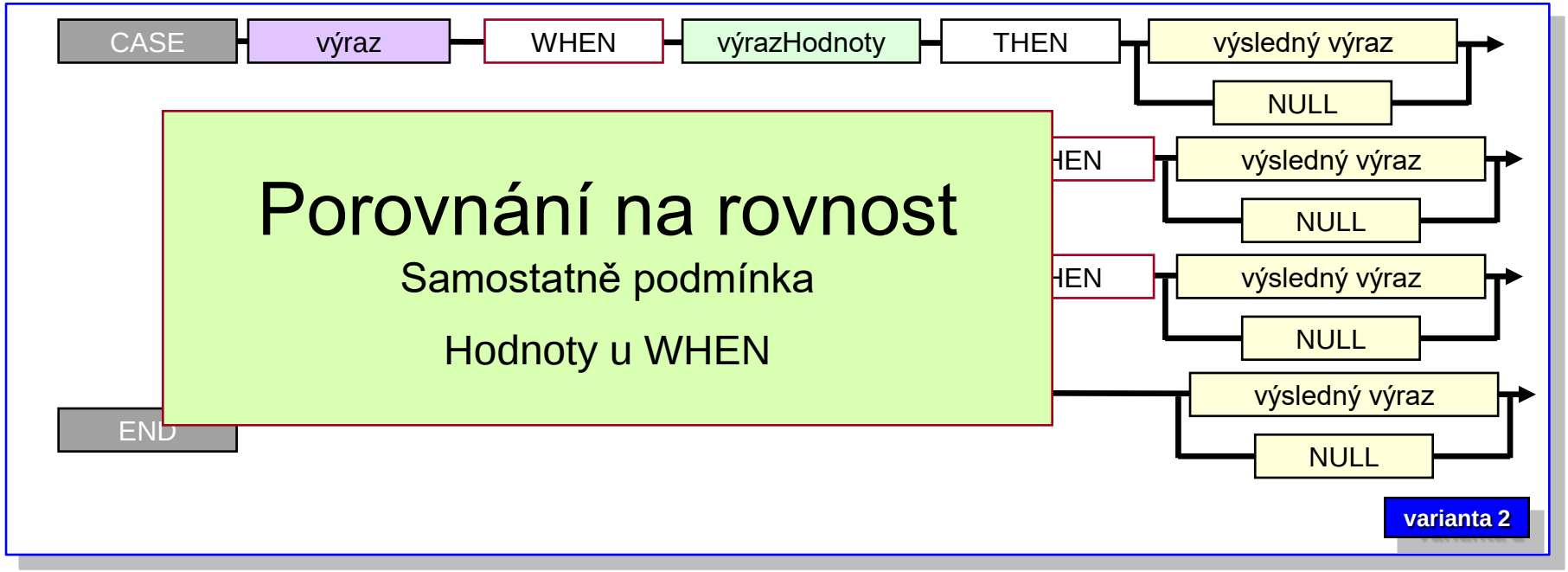
- a nebo to uděláme takhle

```
Select  Sum(
                Case Poh  When   'z'   Then  Castka1
                          When  'm'   Then  Castka2
                 End)       Suma
From    Lid;
```

Suma
460

Podmíněný operátor Case





a teď malé šarády

Operátor Case - Coalesce v agregaci SUM

- zjistěte **součet** částek z tabulky
- avšak
- pro ženy načítejte Částku1
- pro muže Částku2

Lid			
Jm	Poh	Castka1	Castka2
Anna	z	100	1
Bobo	m	2	20
Dana	z	300	3
Emil	m	4	40

- řešte

Suma
460

Suma1	Suma2
406	64

- a nebo to uděláme takhle

```
Select  Sum(
                Case Poh  When   'z'   Then  Castka1
                          When  'm'   Then  Castka2
                 End)       Suma
From    Lid;
```

Suma
460

Operátor Case - Coalesce v agregaci SUM

- zjistěte **součet** částek z tabulky
- avšak
- jméno max 3 znaky - Částku1
- jinak - Částku2

Lid			
Jm	Poh	Castka1	Castka2
Ada	z	100	1
Bobo	m	2	20
Dana	z	300	3
Emil	m	4	40

- řešte

Suma
163

```
Select  Sum(Case
                When Length(Jm) <=3 Then  Castka1
                Else                      Castka2
            End)      Suma
From   Lid;
```

Suma
163

Operátor Case - Coalesce v agregaci SUM

- zjistěte **součet** částek z tabulky
- avšak
- jméno max 3 znaky - Částku1
- jinak - Částku2

Lid			
Jm	Poh	Castka1	Castka2
Ada	z	100	1
Bobo	m	2	20
Dana	z	300	3
Emil	m	4	40

- řešte

Suma
163

```
Select  Sum(Case
                When Length(Jm) <=3 Then  Castka1
                Else                      Castka2
            End)      Suma
From   Lid;
```

Suma
163

Standardní a nestandartní funkce a operátory pro NULL

IFNull (___, ___)

IFNull (Zdroj, Náhrada)

- pokud zdroj je NULL
- navrací náhradní hodnotu
- pokud zdroj není NULL
- navrací zroj

=

NVL (Zdroj, Náhrada)

IsNull (Zdroj, Náhrada)

Coalesce (Zdroj, Náhrada)

Select Coalesce (158, 1000);	→ 158
Select Coalesce (NULL, 1000);	→ 1000
Select Coalesce ('Modrá', 'Bez Barvy');	→ Modrá
Select Coalesce (NULL, 'Bez Barvy');	→ Bez Barvy
Select Coalesce (1/0, **0**);	→ 0 1/0 → NULL
Select Coalesce (1/0, 'Malér')	→ Malér

- vypište tabulku seřazanou dle jmen
- dle příjmení

Clovek		
Pr	Jm	Neco
Balbin	Pepa	
Danek	Franta	
	Zdenek	
	Hynek	
Kubat	Mirek	

```
Select      Pr,
            Jm
From        Clovek
 Order By Pr;
```

Pr	Jm
	Zdenek
	Hynek
Balbin	Pepa
Danek	Franta
Kubat	Mirek

- vypište tabulku seřazanou dle jmen
- dle příjmení
- **avšak**
- není -li příjmení, pak dle jmen

- řešte

Clovek		
Pr	Jm	Neco
Balbin	Pepa	
Danek	Franta	
	Zdenek	
	Hynek	
Kubat	Mirek	

Pr	Jm
	Zdenek
	Hynek
Balbin	Pepa
Danek	Franta
Kubat	Mirek

Pr	Jm
Balbin	Pepa
Danek	Franta
	Hynek
Kubat	Mirek
	Zdenek

```
Select      Pr,
            Jm,
            Coalesce(Pr, Jm) PrJm
From        Clovek
Order By    PrJm;
```

Clovek		
Pr	Jm	PrJm
Balbin	Pepa	Balbin
Danek	Franta	Danek
	Hynek	Hynek
Kubat	Mirek	Kubat
	Zdenek	Zdenek

Operátor Case - Coalesce v klauzuli Order

- vypište tabulku seřazanou dle jmen
- dle příjmení
- **avšak**
- není -li příjmení, pak dle jmen

- řešte

```
Select      Pr,
            Jm

From        Clovek
Order By    Coalesce(Pr, Jm);
```

Clovek		
Pr	Jm	Neco
Balbin	Pepa	
Danek	Franta	
	Zdenek	
	Hynek	
Kubat	Mirek	

Pr	Jm
Balbin	Pepa
Danek	Franta
	Hynek
Kubat	Mirek
	Zdenek

Clovek	
Pr	Jm
Balbin	Pepa
Danek	Franta
	Hynek
Kubat	Mirek
	Zdenek

Operátor Case - Coalesce v klauzuli Order

- vypište tabulku seřazanou dle jmen
- dle příjmení
- avšak
- není -li příjmení, pak dle jmen

- řešte

```
Select      Pr,
            Jm

From        Clovek
Order By    Coalesce(Pr, Jm, 'ZZZZ');
```

Clovek

Pr	Jm	Neco
Balbin	Pepa	
Danek	Franta	
	Zdenek	
	Hynek	
Kubat	Mirek	

Pr	Jm
Balbin	Pepa
Danek	Franta
	Hynek
Kubat	Mirek
	Zdenek

Clovek

Pr	Jm
Balbin	Pepa
Danek	Franta
	Hynek
Kubat	Mirek
	Zdenek

Operátor Case - Coalesce v klauzuli Order

- vypište tabulku seřazanou dle jmen
- dle příjmení
- avšak
- není -li příjmení, pak dle jmen

- řešte

- a jak to prosím probíhá ?

```
Select      Pr,
            Jm

From        Clovek
Order By    Coalesce(Pr, Jm, 'Aaa');
```

Clovek		
Pr	Jm	Neco
Balbin	Pepa	
Danek	Franta	
	Zdenek	
	Hynek	
Kubat	Mirek	

Pr	Jm
Balbin	Pepa
Danek	Franta
	Hynek
Kubat	Mirek
	Zdenek

Clovek	
Pr	Jm
Balbin	Pepa
Danek	Franta
	Hynek
Kubat	Mirek
	Zdenek

Operátor Case - Coalesce v klauzuli Order

- vypište tabulku seřazanou dle jmen
- dle příjmení
- avšak
- není -li příjmení, pak dle jmen

- řešte

```
Select      Pr,
            Jm

From        Clovek
Order By    Coalesce(Pr, Jm, 'Aaa');
```

Clovek			
	Pr	Jm	Neco
Balbin	Balbin	Pepa	
Danek	Danek	Franta	
Zdenek		Zdenek	
Hynek		Hynek	
Kubat	Kubat	Mirek	
Aaa			

Clovek		
	Pr	Jm
Aaa		
Balbin	Balbin	Pepa
Danek	Danek	Franta
Hynek		Hynek
Kubat	Kubat	Mirek
Zdenek		Zdenek

- tabulka TabX má 3 Sloupce
- některá data jsou NULL
- proveďte součet za sloupce A, B, C

TabX		
A	B	C
100	0	0
200	10	9
	20	8
	30	7
300		6
		5

- řešte

```
Select  Sum (A), Sum(B), Sum(C)
  From TabX;
```

600	**50**	**5**

600	**50**	**5**

- tabulka TabX má 3 Sloupce
- některá data jsou NULL
- proveďte součet za sloupec A
- **ovšem**
- **A je Null** : načtěte sloupec B
- **B je Null** : načtěte sloupec C
- **všechny** sloupce **Null**
- odečtěte jedničku

TabX		
A	B	C
100	0	0
200	10	9
	20	8
	30	7
300		6
		5

600	**50**	**5**

Operátor Case - Coalesce v agregaci SUM

- tabulka TabX má 3 Sloupce
- některá data jsou NULL
- proveďte součet za sloupec A
- **ovšem**
- **A je Null** : načtěte sloupec B
- **B je Null** : načtěte sloupec C
- **všechny** sloupce **Null**
- odečtěte jedničku

TabX		
A	B	C
100	0	0
200	10	9
	20	8
	30	7
300		6
		5

- řešte

600	**50**	**5**

```
Select  Sum (
                Case When A is not Null Then   A
                     When B is not Null Then   B
                     When C is not Null Then   C
                     Else                      -1
                 End)    Suma
  From TabX;
```

Suma
654

```
Select  Sum (Coalesce ( A, B, C, -1))    Suma
  From TabX;
```

Suma
654

Operátor Case - Coalesce v klauzuli Where

- vypište pobočky plnící >= 200%
- Prodej/Plan >= 2

- řešte

Pobocka		
Pobo	Plan	Prodej
Brno	2000	4500
Davle	100	1500
Hana	200	420
Chlum	500	780
Jenec	0	20
Kladno	1000	2200

```
Select   Pobo, Plan, Prodej, Round(Prodej/Plan),1) P
 From    Pobocka
 Where   Prodej/Plan >=2.0;
```

Error`

Rádek pro Jenec

20/0 -- Nepovoleno

```
Select   Pobo, Plan, Prodej, Round(Prodej/Plan,1) P
 From    Pobocka
 Where   Case When   Plan = 0   Then    False
                Else     Prodej/Plan >=2.0
          End;
```

Pobo	Plan	Prodej	P
Brno	2000	4500	2.3
Hana	200	420	2.1
Kladno	1000	2200	2.2

Operátor Case - Coalesce v klauzuli Where

- vypište pobočky plnící >= 200%
- když Plan < 1 000
- stačí plnění 150 %

- řešte

Pobocka		
Pobo	Plan	Prodej
Brno	2000	4500
Davle	100	1500
Hana	200	420
Chlum	500	780
Jenec	0	20
Kladno	1000	2200

```
Select   Pobo, Plan, Prodej, Round(Prodej/Plan,1) P
 From    Pobocka
 Where  Case When   Plan = 0   Then    False
              When   Plan < 1000
                     Then Prodej/Plan >=1.5
              Else          Prodej/Plan >=2.0
         End;
```

Pobo	Plan	Prodej	P
Brno	2000	4500	2.3
Hana	200	420	2.1
Chlum	500	780	1.6
Kladno	1000	2200	2.2

i s každým CASE

je jednou

KONEC