

# Databáze – Přednáška 5: Aritmetika datumových a časových datových typů, agregační funkce, vnější spojení

Michal Krátký

Katedra informatiky  
Přírodovědecká fakulta  
Univerzita Palackého v Olomouci

2026/2027

# Obsah přednášky

1. Aritmetika datumových a časových datových typů
2. Agregáčn  funkce
3. Vn jší spojení (outer join)

## Aritmetika datumových a časových datových typů

- ▶ Aritmetika datumových a časových datových typů je **silně závislá na konkrétním databázovém systému**. Budeme probírat implementaci v PostgreSQL dle specifikace SQL92 a SQL99.
- ▶ Pro získání aktuálního data nebo času použijeme proměnné `current_date` resp. `current_timestamp`.
- ▶ **Rozdíl hodnot** typu `date` je počet dnů jako celé číslo nebo hodnota `null`, pokud je jedna z hodnot `null`:

**select**

(**current\_date** - date\_to) pocet\_dnu\_od\_ukonceni\_projektu  
**from** Project;

```
-- Vysledek:
-- pocet_dnu_od_ukonceni_projektu
-- 1755
-- 1025
-- 294
-- NULL
```

## Rozdíl hodnot date jako interval: funkce age

- Pokud potřebujeme získat jiný výsledek rozdílu hodnot typu date než počet dní, musíme použít funkci age, která vrátí rozdíl jako hodnotu interval:

```
select
```

```
    age(current_date, date_to) interval_od_ukonceni_projektu  
from Project;
```

```
-- Výsledek:
```

```
-- interval_od_ukonceni_projektu
```

```
-- 4 years 9 mons 21 days
```

```
-- 2 years 9 mons 21 days
```

```
-- 9 mons 21 days
```

```
-- NULL
```

## Rozdíl hodnot timestamp

- Rozdíl hodnot typu timestamp je hodnota typu interval nebo hodnota null, pokud je některá z hodnot null:

**select**

```
(current_timestamp - date_to) interval_od_ukonceni_projektu  
from Project;
```

```
-- Vysledek:  
-- interval_od_ukonceni_projektu  
-- 1755 days 04:00:16.488508  
-- 1025 days 04:00:16.488508  
-- 294 days 04:00:16.488508  
-- NULL
```

- V tomto případě byl atribut date\_to typu date automaticky přetypován na timestamp s nastaveným nulovým časem.

## Získání části hodnoty: funkce extract

- Pokud chceme získat jednotlivé části hodnot date, time, timestamp nebo interval použijeme funkci extract:

```
select
idp,
extract(year from date_to) date_to_year,
extract(month from date_to) date_to_month,
extract(day from date_to) date_to_day
from Project;
-- Výsledek:
-- idp  date_to_year  date_to_month  date_to_day
-- 1    2021          12             31
-- 2    2023          12             31
-- 3    2025          12             31
-- 4    NULL          NULL            NULL
```

- Dotaz: Vrať projekty, které byly zahájeny v roce 2020.

```
select * from Project
where extract(year from date_from) = 2020;
-- namísto:
-- where date_from between '2020-01-01' and '2020-12-31';
-- Výsledek:
-- idp  name                date_from  date_to
-- 1    IS banky            2020-01-01 2021-12-31
```

## Získání části hodnoty: funkce extract

- ▶ extract můžeme aplikovat i na rozdíl hodnot, protože výsledkem je hodnota interval (pokud u date použijeme funkci age):

**select**

```
idp,
extract(year from age(date_to, date_from)) years,
extract(month from age(date_to, date_from)) months,
extract(day from age(date_to, date_from)) days,
extract(hour from age(date_to, date_from)) hours,
extract(minute from age(date_to, date_from)) minutes,
extract(second from age(date_to, date_from)) seconds
from Project;
```

-- Výsledek:

```
-- idp  years  months  days  hours  minutes seconds
--   1   11    30      0      0      0.000000
--   1   11    30      0      0      0.000000
--   1   11    30      0      0      0.000000
-- NULL NULL  NULL    NULL  NULL  NULL
```

## Délka trvání hodnoty interval

- Pokud chceme převést hodnotu interval na celkový počet roků, měsíců, dní, hodin, minut a vteřin, musíme použít specifické výpočty, případně použít `extract – epoch`, která vrátí počet vteřin intervalu:

```
select
  idp,
  (extract(year from age(date_to, date_from)) * 12 +
   extract(month from age(date_to, date_from))) month_count,
  extract(epoch from age(date_to, date_from)) / 3600 hour_count
from Project;
```

```
-- Výsledek:
-- idp  month_count  hour_count
-- 1    23          17406.000000000000
-- 2    23          17406.000000000000
-- 3    23          17406.000000000000
-- 4    NULL        NULL
```

# Dotazy I

- **Dotaz:** Vrať projekty delší než 1 rok (včetně).

```
select * from Project
where extract(year from age(date_to, date_from)) >= 1;
```

-- Výsledek:

| -- idp | name     | date_from  | date_to    |
|--------|----------|------------|------------|
| -- 1   | IS banky | 2020-01-01 | 2021-12-31 |
| -- 2   | IS firmy | 2022-01-01 | 2023-12-31 |
| -- 3   | IS úřadu | 2024-01-01 | 2025-12-31 |

- **Dotaz:** Vrať projekty, které začaly dříve než před rokem.

```
select * from Project
where extract(year from age(current_date, date_from)) < 1;
```

-- Výsledek:

| -- idp | name           | date_from  | date_to |
|--------|----------------|------------|---------|
| -- 4   | Spisová služba | 2026-01-01 | NULL    |

## Dotazy II

- **Dotaz:** Vrať projekty, které začaly dříve než před 13 měsíci.

```
select * from Project
where
  (current_date - interval '13 months') < date_from;
```

```
-- Výsledek:
-- idp  name                date_from      date_to
-- 4     Spisová služba     2026-01-01     NULL
```

- **Protože:**

```
select
  (current_date - interval '13 months'), date_from
from Project;
```

```
-- Výsledek:
-- Column1                date_from
-- 2025-09-21 00:00:00     2020-01-01
-- 2025-09-21 00:00:00     2022-01-01
-- 2025-09-21 00:00:00     2024-01-01
-- 2025-09-21 00:00:00     2026-01-01
```

## Agregací funkce

- ▶ **Agregací funkce**, které používáme v **projekci** příkazu `select`, počítají následující hodnoty pro hodnoty atributů tabulky sestavené pomocí `from` nebo `from - join` a selekce (podmínky za klauzulí `where`)<sup>1</sup>:
  - ▶ `count` – počet hodnot,
  - ▶ `sum` – součet hodnot,
  - ▶ `avg` – průměrná hodnota,
  - ▶ `min` – minimální hodnota,
  - ▶ `max` – maximální hodnota.

---

<sup>1</sup>Označíme si tuto tabulku jako **výstupní**.

## count

- ▶ **Agregační funkce** count je mezi ostatními funkcemi specifická – můžeme ji aplikovat na kompletní záznamy pomocí symbolu \* i na jednotlivé atributy.
- ▶ Rozdíl je překvapivě významný. Zatímco count(\*) počítá počet záznamů výstupní tabulky:

```
select count(*) from "User";
-- count
-- 3
```

- ▶ count(<attribute\_name>) vrací počet hodnot atributu výstupní tabulky, které nejsou null:

```
select count(idd) from "User";
-- count
-- 2
```

**Výsledky se liší:** jeden uživatel nemá nastaveno oddělení.

## count

- ▶ count (a ostatní agregační funkce) můžeme kombinovat.
- ▶ **Dotaz:** Vrať počet uživatelů i počet uživatelů s nastaveným oddělením:

```
select count(*) pocet_uzivatelu,
       count (idd) pocet_uzivatelu_s_oddelenim
from "User";
```

```
-- pocet_uzivatelu      pocet_uzivatelu_s_oddelenim
-- 3                   2
```

- ▶ Ale **není možné** je kombinovat s projekcí na atributy:

```
select idu, fname, count(*) pocet_uzivatelu
from "User";
```

```
-- Chyba:
-- column "User.idu" must appear in the GROUP BY
-- clause or be used in an aggregate function
-- LINE 1: select idu, fname, count(*) pocet_uzivatelu
```

## sum, avg, min, max

- **Dotaz:** agregační funkce: sum, avg, min, max.

**select**

**sum**(date\_to - date\_from) sum\_day\_count,

**avg**(date\_to - date\_from) avg\_day\_count,

**min**(date\_to - date\_from) min\_day\_count,

**max**(date\_to - date\_from) max\_day\_count

**from** Project;

-- Výsledek:

-- sum\_day\_count avg\_day\_count min\_day\_count max\_day\_count

-- 2189 729.67 729 730

- **Protože:**

**select** idp, (date\_to - date\_from) day\_count

**from** Project;

-- Výsledek:

-- idp day\_count

-- 1 730

-- 2 729

-- 3 730

-- 4 NULL

## Podmíněná agregace

- ▶ V agregační funkci můžeme použít podmínku case, která umožní počítat jen určité hodnoty atributu.
- ▶ **Dotaz:** Vrať počet uživatelů z Olomouce a počet uživatelů z Prahy.

```
select
  count(case when city='Olomouc' then 1 end) pocet_olomouc,
  count(case when city='Praha' then 1 end) pocet_praha
from "User";
```

```
-- Výsledek:
-- pocet_olomouc      pocet_praha
-- 3                  0
```

- ▶ Pokud je hodnota atributu city Olomouc / Praha, case vrátí 1, což znamená inkrementaci počtu hodnot.
- ▶ Všimněme si, že jsme takto vrátili dvě hodnoty vypočítané z jediného atributu.

## Vnější spojení (outer join)

- ▶ Zásadní vlastností **vnitřního spojení (inner join nebo join)** je, že zařazuje do výsledku všechny dvojice záznamů spojovaných tabulek, pro které spojovací podmínka nabývá hodnoty true.
- ▶ Pokud je hodnota cizího klíče spojované tabulky null, pak tento záznam nebude zařazen do výsledku (podmínka je vždy vyhodnocena jako unknown).
- ▶ **Uvažujme dotaz:** Vrať uživatele (id, jméno a příjmení), jejich oddělení (název) a jejich projekty (id a název).
- ▶ Tuto úlohu **nemůžeme řešit pomocí vnitřního spojení**, protože uživatel, který má nějaké projekty, ale nemá žádné oddělení, je eliminován operací User  $\bowtie$  Department.
- ▶ Proto je zavedeno **vnější spojení (outer join)** ve variantě levé (left outer join –  $\bowtie\!\!\!\!\!\leftarrow$ ) a pravé (right outer join –  $\rightarrow\!\!\!\!\!\bowtie$ ).

## left/right (outer) join

- ▶ Do výsledku outer join jsou **zařazeny všechny dvojice záznamů** obou tabulek, pro které je hodnota **spojovací podmínky true**, tedy stejně jako pro vnitřní spojení.
- ▶ **Ale také záznamy z levé/pravé tabulky**, pro left / right (outer) join, které **nemají vazbu** na žádný záznam z tabulky druhé. V takovém případě jsou hodnoty atributů druhé tabulky null.
- ▶ Tyto operace **nejsou zjevně komutativní** – rozlišují levou a pravou stranu.

## Dotaz 1a

- ▶ **Dotaz:** Vrať všechny uživatele (id, jméno a příjmení) a jejich oddělení (id a jméno).
- ▶ Ve výsledku join chybí záznamy uživatelů bez oddělení (uživatel s idu 3):

```
select u.idu, u.fname, u.lname, d.idd, d.name
from "User" u
join Department d on u.idd = d.idd;
```

```
-- Výsledek:
-- idu  fname      lname      idd      name
-- 2     Yveta      Švecová   3        IT oddělení
-- 1     Jan        Sobota    4        Výroba
```

## Dotaz 1b

- ▶ **Dotaz:** Vrať všechny uživatele (id, jméno a příjmení) a jejich oddělení (id a jméno).
- ▶ left (outer) join vede na korektní výsledek, uživatel s id 3 je ve výsledku s hodnotami atributů Department null:

```
select u.idu, u.fname, u.lname, d.idd, d.name
from "User" u
left join Department d on u.idd = d.idd;
```

```
-- Výsledek:
-- idu  fname    lname    idd      name
-- 2     Yveta     Švecová  3        IT oddělení
-- 1     Jan       Sobota   4        Výroba
-- 3     Petr      Švec     NULL     NULL
```

## Dotaz 1c

- ▶ **Dotaz:** Vrať všechny uživatele (id, jméno a příjmení) a jejich oddělení (id a jméno).
- ▶ right (outer) join můžeme použít, ale musíme dát tabulku User na pravou stranu operace:

```
select u.idu, u.fname, u.lname, d.idd, d.name
from Department d
right join "User" u on u.idd = d.idd;
```

```
-- Výsledek:
-- idu  fname  lname  idd      name
-- 2     Yveta  Švecová 3      IT oddělení
-- 1     Jan    Sobota  4      Výroba
-- 3     Petr   Švec    NULL    NULL
```

## Funkce coalesce

- ▶ Pokud chceme hodnotu null ve výsledku nahradit jinou hodnotou, můžeme použít funkci coalesce.
- ▶ Pokud je hodnota prvního parametru null, coalesce vrátí hodnotu druhého parametru, jinak vrátí hodnotu prvního parametru.
- ▶ **Dotaz:** Vrať všechny uživatele (id, jméno a příjmení) a jejich oddělení (id a jméno). Pokud uživatel nemá zadáno oddělení, vrať jako jméno oddělení řetězec 'Žádné oddělení'.

```
select u.idu, u.fname, u.lname, d.idd,
       coalesce(d.name, 'Žádné oddělení')
from "User" u
left outer join Department d on u.idd = d.idd;
```

-- Výsledek:

| -- idu | fname | lname   | idd  | coalesce       |
|--------|-------|---------|------|----------------|
| -- 2   | Yveta | Švecová | 3    | IT oddělení    |
| -- 1   | Jan   | Sobota  | 4    | Výroba         |
| -- 3   | Petr  | Švec    | NULL | Žádné oddělení |

## Dotaz 2a

- ▶ **Dotaz:** Vrať všechna oddělení (id a jméno) a jejich uživatele (id, jméno a příjmení).
- ▶ Použití (inner) join nevede ke správnému výsledku, protože přijdeme o dvě oddělení, která nemají žádné uživatele (idd 1 a 2):

```
select d.idd, d.name, u.idu, u.fname, u.lname
from Department d
join "User" u on u.idd = d.idd;
```

```
-- Výsledek:
-- idd  name      idu      fname      lname
-- 3     IT oddělení  2         Yveta      Švecová
-- 4     Výroba     1         Jan        Sobota
```

## Dotaz 2b

- ▶ **Dotaz:** Vrať všechna oddělení (id a jméno) a jejich uživatele (id, jméno a příjmení).
- ▶ Správné řešení musí použít left (outer) join, ve výsledku jsou pak všechny záznamy tabulky na levé straně, tedy Department.

```
select d.idd, d.name, u.idu, u.fname, u.lname
from Department d
left join "User" u on u.idd = d.idd;
```

-- Výsledek:

| --   | idd             | name | idu   | fname   | lname |
|------|-----------------|------|-------|---------|-------|
| -- 1 | Vedení          | NULL | NULL  | NULL    |       |
| -- 2 | Účetní oddělení | NULL | NULL  | NULL    |       |
| -- 3 | IT oddělení     | 2    | Yveta | Švecová |       |
| -- 4 | Výroba          | 1    | Jan   | Sobota  |       |

## Dotaz 2c

- ▶ **Dotaz:** Vrať všechna oddělení (id a jméno) a jejich uživatele (id, jméno a příjmení).
- ▶ **Můžeme přepsat** outer join pomocí inner join?

```
select d.idd, d.name, u.idu, u.fname, u.lname
from Department d
join "User" u on u.idd = d.idd or u.idd is null;
```

-- Výsledek:

| -- idd | name            | idu | fname | lname   |
|--------|-----------------|-----|-------|---------|
| -- 1   | Vedení          | 3   | Petr  | Švec    |
| -- 2   | Účetní oddělení | 3   | Petr  | Švec    |
| -- 3   | IT oddělení     | 3   | Petr  | Švec    |
| -- 3   | IT oddělení     | 2   | Yveta | Švecová |
| -- 4   | Výroba          | 1   | Jan   | Sobota  |
| -- 4   | Výroba          | 3   | Petr  | Švec    |

- ▶ **Nemůžeme:** uživatel bez oddělení (idu=3) se ve výsledku zopakuje pro každé oddělení, protože pro všechny tyto dvojice záznamů je výsledek spojovací podmínky true.

## Dotaz 3a

- ▶ **Dotaz:** Vrať počet uživatelů oddělení s názvem 'Vedení'.
- ▶ V řešení využijeme opakování záznamů v případě vazby 1 : N mezi spojovanými tabulkami. Záznam oddělení je ve výsledku spojení Department ⋈ User tolikrát, kolik uživatelů má zadáno toto oddělení.
- ▶ Řešení pomocí join je v tomto případě korektní, protože výsledek je pro oddělení, která nemají uživatele, prázdný:

```
select count(*) pocet_uzivatelu
from Department d
join "User" u on u.idd = d.idd
where d.name = 'Vedení';
```

```
-- Výsledek:
-- pocet_uzivatelu
-- 0
```

## Dotaz 3b

- ▶ **Dotaz:** Vrať počet uživatelů oddělení s názvem 'Vedení'.
- ▶ V kontextu seskupování záznamů (následující téma) je rozumnější použít left (outer) join. V takovém případě musíme použít count(u.idu), které vrátí 0, namísto count(\*), které vrátí 1.

```
select count(u.idu) pocet_uzivatelu
from Department d
left join "User" u on u.idd = d.idd
where d.name = 'Vedení';
```

```
-- Výsledek:
-- pocet_uzivatelu
-- 0
```

```
-- Protože:
-- idd  name    idu      fname    lname
-- 1    Vedení  NULL      NULL     NULL
```

## Dotaz 4a

- ▶ **Dotaz:** Vrať **všechny** uživatele (id, jméno a příjmení), jejich oddělení (id a název) a jejich projekty (id a název).
- ▶ Nastavme uživateli s idu 2 skupinu jako nenastavenou, tento uživatel pracuje na dvou projektech:

```
update "User" set idd = null where idu = 2;
```

- ▶ Řešení s join vyřadí uživatele s idu 2 bez oddělení, přestože má nějaké projekty.

```
select u.idu, u.fname, u.lname, d.idd, d.name, p.idp, p.name
from "User" u
join Department d on u.idd = d.idd
join User_Project up on u.idu = up.idu
join Project p on up.idp = p.idp;
```

-- Výsledek:

| -- idu | fname | lname  | idd | name   | idp | name           |
|--------|-------|--------|-----|--------|-----|----------------|
| -- 1   | Jan   | Sobota | 4   | Výroba | 1   | IS banky       |
| -- 1   | Jan   | Sobota | 4   | Výroba | 2   | IS firmy       |
| -- 1   | Jan   | Sobota | 4   | Výroba | 3   | IS úřadu       |
| -- 1   | Jan   | Sobota | 4   | Výroba | 4   | Spisová služba |

## Dotaz 4b

- **Dotaz:** Vrať **všechny** uživatele (id, jméno a příjmení), jejich oddělení (id a název) a jejich projekty (id a název).
- Řešení s left (outer) join vrátí i uživatele s idu 2 bez oddělení.

```
select u.idu, u.fname, u.lname, d.idd, d.name, p.idp, p.name
from "User" u
left outer join Department d on u.idd = d.idd
join User_Project up on u.idu = up.idu
join Project p on up.idp = p.idp;
```

```
-- Výsledek:
-- idu  fname  lname  idd      name      idp      name
-- 1      Jan     Sobota  4        Výroba    1        IS banky
-- 1      Jan     Sobota  4        Výroba    2        IS firmy
-- 1      Jan     Sobota  4        Výroba    3        IS úřadu
-- 1      Jan     Sobota  4        Výroba    4        Spisová služba
-- 2      Yveta    Švecová NULL      NULL      1        IS banky
-- 2      Yveta    Švecová NULL      NULL      2        IS firmy
```

- Ve výsledku ale není uživatel s idu 3.

## Dotaz 4c

- ▶ **Dotaz:** Vrať **všechny** uživatele (id, jméno a příjmení), jejich oddělení (id a název) a jejich projekty (id a název).
- ▶ Jelikož uživatel s idu 3 nemá žádný projekt, User  $\bowtie$  User\_Project a User\_Project  $\bowtie$  Project vyřadí tento záznam z výsledku, proto musíme v obou případech použít left (outer) join.

```
select u.idu, u.fname, u.lname, d.idd, d.name, p.idp, p.name
from "User" u
left outer join Department d on u.idd = d.idd
left join User_Project up on u.idu = up.idu
left join Project p on up.idp = p.idp;
```

-- Výsledek:

| -- idu | fname | lname   | idd  | name   | idp  | name           |
|--------|-------|---------|------|--------|------|----------------|
| -- 1   | Jan   | Sobota  | 4    | Výroba | 1    | IS banky       |
| -- 1   | Jan   | Sobota  | 4    | Výroba | 2    | IS firmy       |
| -- 1   | Jan   | Sobota  | 4    | Výroba | 3    | IS úřadu       |
| -- 1   | Jan   | Sobota  | 4    | Výroba | 4    | Spisová služba |
| -- 2   | Yveta | Švecová | NULL | NULL   | 1    | IS banky       |
| -- 2   | Yveta | Švecová | NULL | NULL   | 2    | IS firmy       |
| -- 3   | Petr  | Švec    | NULL | NULL   | NULL | NULL           |

## Dotaz 4d

- Proč nevedou ke správnému řešení následující SQL dotazy?

```
select u.idu, u.fname, u.lname, d.idd, d.name, p.idp, p.name
from "User" u
left join Department d on u.idd = d.idd
join User_Project up on u.idu = up.idu
left join Project p on up.idp = p.idp;
```

```
select u.idu, u.fname, u.lname, d.idd, d.name, p.idp, p.name
from "User" u
left join Department d on u.idd = d.idd
left join User_Project up on u.idu = up.idu
join Project p on up.idp = p.idp;
```

- Pokud uživatel nemá projekt, tedy v User\_Project není záznam s jeho idu, pak tento záznam nemůže být ve výsledku User ⋈ User\_Project ani User\_Project ⋈ Project.

## full (outer) join

- ▶ full (outer) join  $\bowtie$  vrátí výsledek join, left outer join i right outer join.
- ▶ **Dotaz:** Vrať uživatele (id, jméno a příjmení), jejich oddělení (id a název) a jejich projekty (id a název). Ve výsledku budou všichni uživatelé i oddělení.

```
select u.idu, u.fname, u.lname, d.idd, d.name
from "User" u
full join Department d on u.idd = d.idd;
```

```
-- Výsledek:
-- idu  fname    lname    idd      name
-- NULL NULL     NULL     1        Vedení
-- NULL NULL     NULL     2        Účetní oddělení
-- NULL NULL     NULL     3        IT oddělení
-- 1     Jan      Sobota   4        Výroba
-- 2     Yveta    Švecová  NULL     NULL
-- 3     Petr     Švec     NULL     NULL
```

- ▶ Pouze jeden uživatel (s idu 1) má nastaveno oddělení.