

# Databáze – Přednáška 3: Vazba M:N

Michal Krátký

Katedra informatiky  
Přírodovědecká fakulta  
Univerzita Palackého v Olomouci

2026/2027

# Obsah přednášky

1. Konceptuální model
2. Datumové a časové datové typy v SQL
3. Konceptuální model, ERD
4. Relační datový model, ERD
5. Implementace v SQL
6. Dotazování tabulek ve vazbě M:N
7. Projekce

## Vazba M:N

- ▶ Pokud při stanovení kardinality vazby dvou entitních typů zjistíme, že entita může mít z obou stran vztah k více entitám druhého typu, pak má tato vazba **kardinalitu M:N (many to many)**.
- ▶ Uvažujme pokračování našeho zadání:
  - ▶ V informačním systému budeme evidovat jednotlivé **uživatele**. Pro každého uživatele budeme evidovat jedinečné číslo, jméno, příjmení, bydliště (ulici, číslo popisné, město) a email.

Každý uživatel je ve firmě pracovníkem nejvýše jednoho **oddělení**, u kterého chceme evidovat jedinečné číslo a název (který je povinný). Oddělení může mít více pracovníků.

Chceme evidovat **projekty**, na kterých může pracovat více uživatelů. U každého projektu budeme evidovat: název, začátek a konec projektu. Uživatel může pracovat na více projektech.

## Lineární zápis konceptuálního modelu

- ▶ V **lineárním zápise konceptuálního modelu** nám přibyl entitní typ Project, do kterého jsme přidali syntetický klíč idp:

Project(idp, date\_from, date\_to)

- ▶ A **vazba** mezi entitními typy User a Project:

WORKS\_AT(User, Project)

- ▶ **Kardinalita** vazby je **M:N**, protože uživatel může pracovat na **více** projektech a na jednom projektu může pracovat **více** uživatelů.
- ▶ **Členství** obou entitních typů ve vazbě je **nepovinné**, protože uživatel nemusí pracovat na **žádném** projektu, například nový zaměstnanec. Na projektu nemusí pracovat **žádný** uživatel, například v případě nově zadaného projektu.

## Entitní typ vs entita

- Mějme tyto konkrétní **entity** typu Project:

```
-- (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')
(4, 'Spisová služba', '2026-01-01', null)
```

- **Datumové (případně datumové a časové) hodnoty** píšeme (i na úrovni konceptuálního modelu) nejčastěji do apostrofů, jelikož se tento zápis používá v SQL.

## Datumové a časové datové typy v SQL

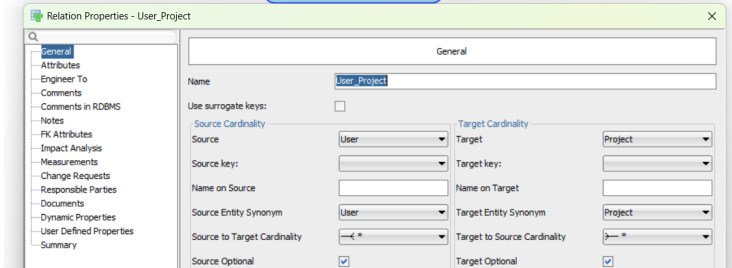
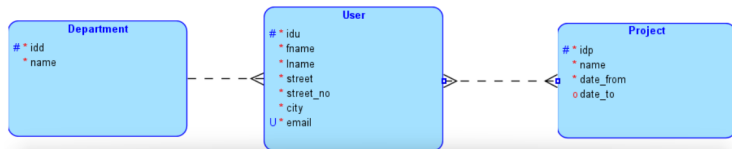
- ▶ **Standard SQL obsahuje čtyři datové typy** pro uložení data, času a intervalu:
  - ▶ **date**: pouze datum v implicitním formátu 'YYYY-MM-DD'.
  - ▶ **time**: pouze čas v rámci dne v implicitním formátu 'HH:MM:SS'.
  - ▶ **timestamp**: datum a čas v implicitním formátu 'YYYY-MM-DD HH:MM:SS'.
  - ▶ **interval**: časový interval, může vzniknout například jako rozdíl dvou hodnot timestamp.
- ▶ Specifikace dále obsahuje modifikace datových typů time a timestamp pro **definici časové zóny**.

## Implementace datumových a časových typů v některých DBS

- ▶ **PostgreSQL** téměř dokonale implementuje specifikaci SQL: hodnota date je uložena ve 4 B, timestamp v 8 B s přesností na mikrosekundy a interval dokonce nad rámec specifikace jako 16 B hodnota.
- ▶ **SQL Server** historicky implementoval datetime (jako období timestamp z SQL); samostatný typ date obsahuje až od verze 2008. Typ interval zcela chybí (nahrazuje se funkcemi).
- ▶ **Oracle** má interval dle specifikace SQL, typy date a timestamp ale shodně obsahují datum i čas s drobnými odchylkami.

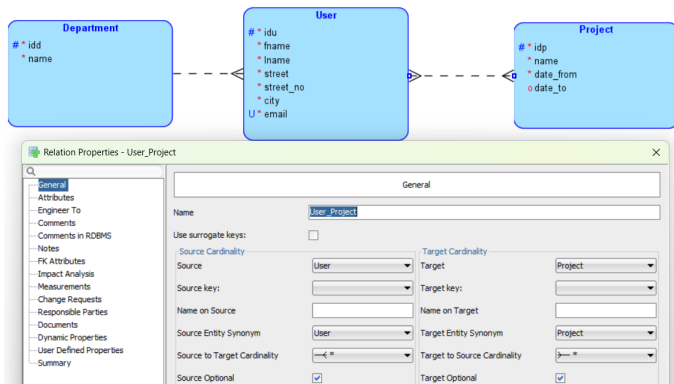
⇒ Při přechodu na jiný DBS nás nemile překvapí **rozdíly v implementaci datumových a časových datových typů**.

## Vazba M:N v ERD 1/4



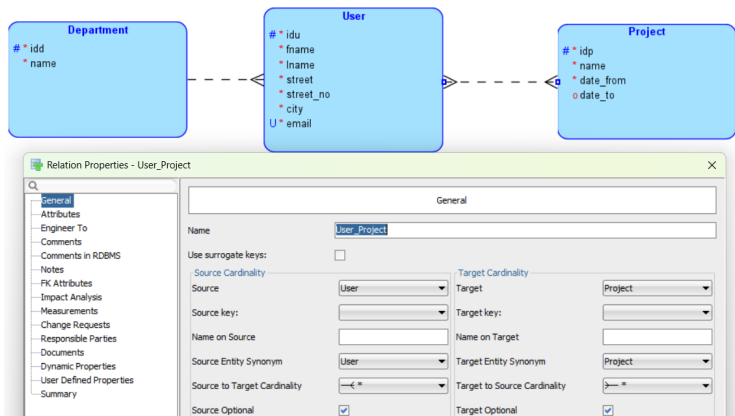
- ▶ V nástroji Oracle Modeler najdeme vazbu **M:N** pod ikonou New M:N Relation.
- ▶ Na **rozdíl** od vazby **1:N** se nemusíme starat o počáteční a koncový entitní typ vazby z pohledu strany s kardinalitou 1.

## Vazba M:N v ERD 2/4



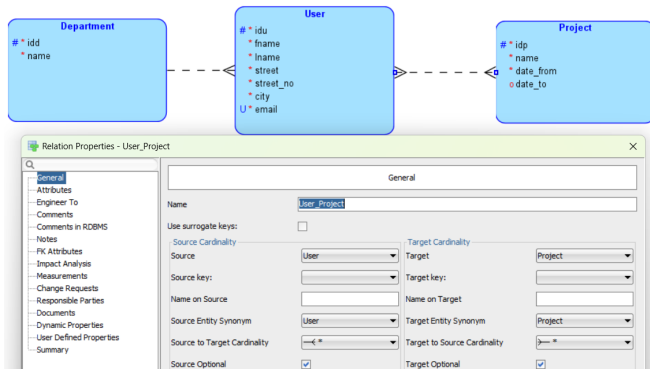
- ▶ Všimněme si symbolu pro kardinalitu  $M$  a  $N$ , tedy **vraní nohy** (**Crow's foot**) u obou entitních typů, resp. v nastavení Source to Target a Target to Source Cardinality.
- ▶ Členství obou entitních typů ve vazbě je **nepovinné** – volby Source Optional i Target Optional jsou zatrženy.

## Vazba M:N v ERD 3/4



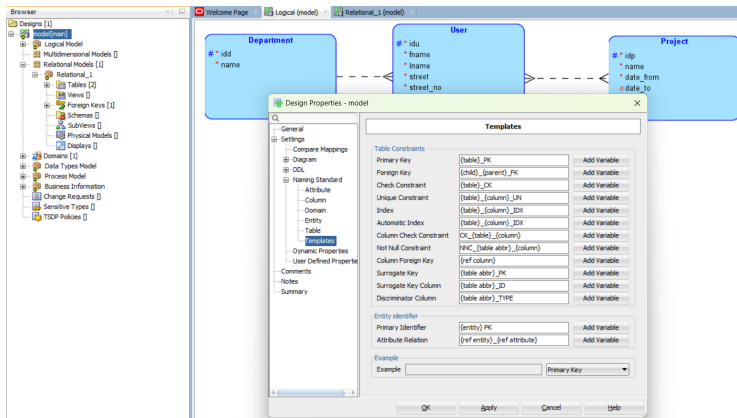
- Název vazby jsme zvolili User\_Project namísto WORKS\_AT, čímž se chystáme na implementaci vazby **M:N** v relačním datovém modelu.

## Vazba M:N v ERD 4/4



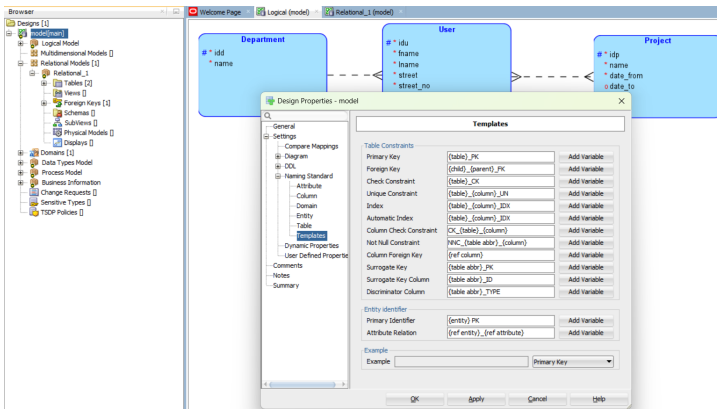
- ▶ Všimněme si **povinnosti (hodnoty) atributu** Project.date\_from – při vložení nového projektu musíme znát datum jeho začátku.
- ▶ Naopak hodnota atributu Project.date\_to je **nepovinná** – při vytváření nového projektu datum jeho ukončení zpravidla neznáme.

# Vazba M:N v relačním datovém modelu, ERD 1/7



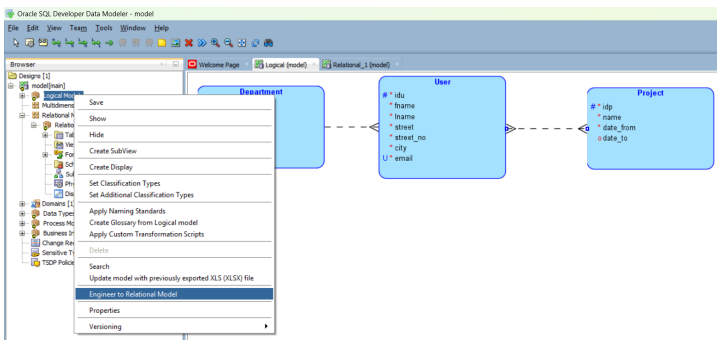
- Před převodem konceptuálního modelu v ERD na relační datový model v nástroji Oracle Modeler, nastavíme šablonu pro názvy cizích klíčů tak, aby se cizí klíč jmenoval stejně jako odkazovaný (primární) klíč.

# Vazba M:N v relačním datovém modelu, ERD 2/7



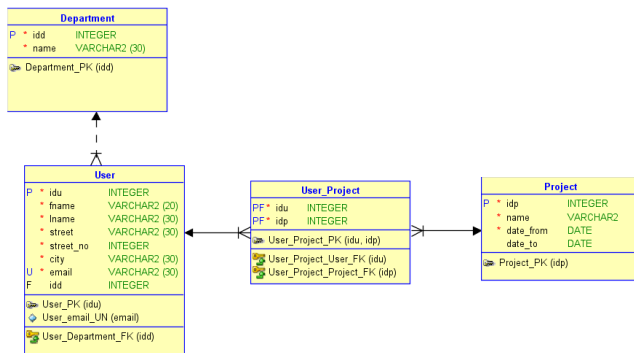
- V našem návrhu, v tomto případě model, klikneme pravým tlačítkem a zvolíme Properties/Settings/Naming Standard/Templates a u Column Foreign Key přepíšeme {ref table}\_{ref column} na {ref column}.

## Vazba M:N v relačním datovém modelu, ERD 3/7



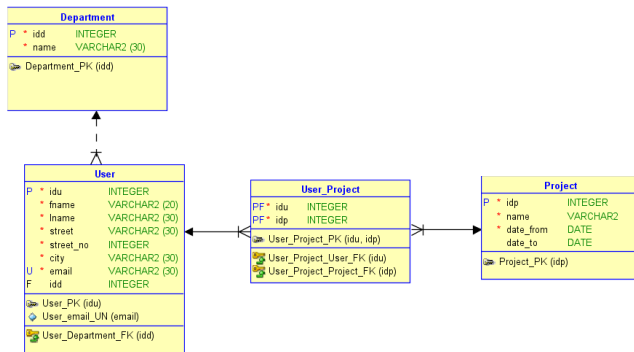
- Převědeme konceptuální model v ERD na relační datový model – pravým tlačítkem klikneme na Logical Model (konceptuální model) a zvolíme Engineer to Relational Model.

# Vazba M:N v relačním datovém modelu, ERD 4/7



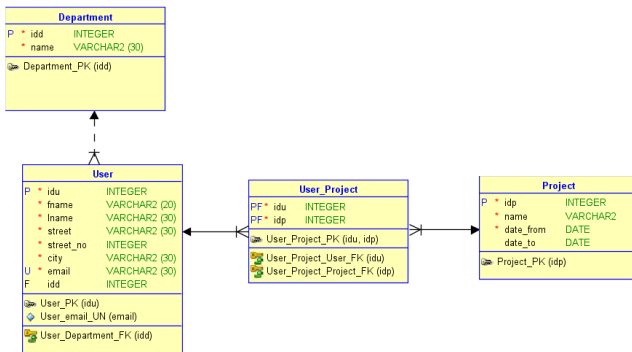
- ▶ V ERD vidíme následující vlastnosti **implementace vazby M:N v relačním datovém modelu**:
  - ▶ Vazba M:N je implementována novou tabulkou, tzv. **vazební tabulkou (junction/associative table)**, která má vazby **1:N** k oběma původním tabulkám.

## Vazba M:N v relačním datovém modelu, ERD 5/7



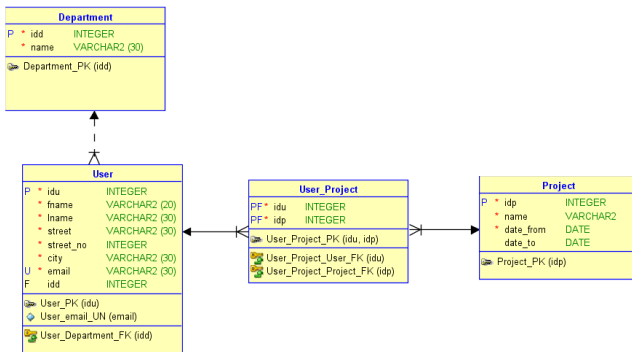
- **Vazba M:N** je implementována novou (tzv. **vazební**) tabulkou s vazbami **1:N** k oběma původním tabulkám:
  - Obě vazby **1:N** mají vždy kardinalitu **N** na straně vazební tabulky a **1** na straně původních tabulek.

## Vazba M:N v relačním datovém modelu, ERD 6/7



- ▶ **Vazba M:N** je implementována novou (tzv. **vazební**) tabulkou s vazbami **1:N** k oběma původním tabulkám:
  - ▶ Vazební tabulka obsahuje vždy primární klíče původních tabulek jako **cizí klíče** (zde idu a idp) a dohromady tvoří **primární klíč**.
  - ▶ Jelikož primární klíč nemůže mít hodnotu null, **vazební tabulka** má povinné členství v obou vazbách 1:N.

# Vazba M:N v relačním datovém modelu, ERD 7/7



- Všimněme si detailů notace **Information Engineering** v ERD pro relační datový model:
  - Symbol pro kardinalitu 1 je znázorněn **černou plnou šipkou**.
  - Čára vazby nemá polovinu čárkovanou pro nepovinné členství a polovinu plnou pro povinné členství – čára je plná po celé délce z důvodu povinného členství tabulky s cizím klíčem.

## Tabulka Project

- ▶ Nejprve vytvoříme tabulku Project pomocí SQL příkazu jazyka pro definici dat (JDD/DDDL) create table:

```
create table Project
(
    idp int primary key,
    name varchar(100) not null,
    date_from date not null,
    date_to date
);
```

- ▶ A vložíme čtyři záznamy:

```
insert into Project
values
(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'),
(4, 'Spisová služba', '2026-01-01', null);
```

## Vazební tabulka User\_Project I

- ▶ Vytvoříme **vazební tabulku** User\_Project pomocí SQL příkazu jazyka pro definici dat (DDL) create table:

```
create table User_Project  
(  
    idu int references "User"(idu),  
    idp int references Project(idp),  
    primary key(idu, idp)  
);
```

- ▶ Všimněme si definice entitního integritního omezení mimo definici atributu, v tomto případě je to dokonce nutné – **primární klíč je složený** ze dvou atributů.
- ▶ Cizí klíče nemusíme v tomto případě označovat not null – je automaticky vyřešeno jejich zahrnutím do primárního klíče.

## Vazební tabulka User\_Project II

- Dokonce můžeme integritní omezení pojmenovat:

```
create table User_Project  
(  
    idu int references "User"(idu),  
    idp int references Project(idp),  
    constraint pk_user_project primary key(idu, idp)  
);
```

- Například, pokud jej potřebujeme v budoucnu, z různých důvodů, zrušit příkazem SQL jazyka pro definici dat (DDL) alter table:

```
alter table User_Project  
    drop constraint pk_user_project;
```

## Dotazování tabulek ve vazbě M:N I

- ▶ Pro dotazování tabulek ve **vazbě M:N**, tedy například: „**vrať uživatele a jejich projekty**“, opět používáme konstrukci **join** (**spojení**)<sup>1</sup> příkazu jazyka pro dotazování dat (DQL) **select**.
- ▶ Nemůžeme ale spojit přímo původní tabulky vazby, protože neobsahují žádné cizí klíče implementující tuto vazbu:

```
select * from "User" u  
join Project p on u.? = p.?
```

---

<sup>1</sup>Spojení označujeme symbolem ⋈.

## Dotazování tabulek ve vazbě M:N II

- ▶ Musíme tedy nejprve spojit tabulku User a vazební tabulku User\_Project a implementovat tak dotaz:  
„vrať uživatele a id jejich projektů“:

```
select * from "User" u  
join User_Project up on u.idu = up.idu;
```

- ▶ Výsledkem je nyní prázdná množina, jelikož tabulka User\_Project je prázdná.

## Vložení záznamů do vazební tabulky

- ▶ Abychom přiřadili projekt uživateli, musíme **vložit záznam do vazební tabulky** SQL příkazem jazyka pro manipulaci s daty (DML) `insert`.
- ▶ Nebo **vložíme více záznamů** tak, že uživatel s `idu=1` je na všech čtyřech projektech, uživatel s `idu=2` na dvou projektech a uživatel s `idu=3` není na žádném projektu:

```
insert into User_Project(idu, idp)  
values  
    (1, 1), (1, 2), (1, 3), (1, 4),  
    (2, 1), (2, 2);
```

## Dotazování tabulek ve vazbě M:N III

- Vraťme se k dotazu: „**vrať uživatele a id jejich projektů**“.

```
select * from "User" u
join User_Project up on u.idu = up.idu;
```

- **Výsledek:**

| idu | fname | lname   | street        | street_no | city    | ... | idd | idu | idp |
|-----|-------|---------|---------------|-----------|---------|-----|-----|-----|-----|
| 1   | Jan   | Sobota  | 17. listopadu | 15        | Olomouc | ... | 4   | 1   | 1   |
| 1   | Jan   | Sobota  | 17. listopadu | 15        | Olomouc | ... | 4   | 1   | 2   |
| 1   | Jan   | Sobota  | 17. listopadu | 15        | Olomouc | ... | 4   | 1   | 3   |
| 1   | Jan   | Sobota  | 17. listopadu | 15        | Olomouc | ... | 4   | 1   | 4   |
| 2   | Yveta | Švecová | Hněvotínská   | 215       | Olomouc | ... | 3   | 2   | 1   |
| 2   | Yveta | Švecová | Hněvotínská   | 215       | Olomouc | ... | 3   | 2   | 2   |

- **Pozorování:**

- Počet záznamů ve výsledku je stejný (6) jako počet záznamů ve vazební tabulce, jelikož každý záznam vazební tabulky musí odkazovat na nějaký záznam tabulky User.
- Záznam uživatele s idu=1 je ve výsledku dotazu 4×, záznam uživatele s idu=2 2×, zatímco záznam uživatele s idu=3 se ve výsledku nevyskytuje.

## Dotazování tabulek ve vazbě M:N IV

- ▶ Vraťme se k dotazu: „**vrať uživatele a jejich projekty**“.
- ▶ Nyní potřebujeme na výsledek operace  $\text{User} \bowtie \text{User\_Project}$  aplikovat další spojení, tentokrát s tabulkou `Project`.
- ▶ Nyní je to možné, protože výsledek  $\text{User} \bowtie \text{User\_Project}$  obsahuje `idp`, tedy odkaz na záznamy v tabulce `Project`.
- ▶ **SQL dotaz:**

```
select * from "User" u
join User_Project up on u.idu = up.idu
join Project p on up.idp = p.idp;
```

## Dotazování tabulek ve vazbě M:N V

### ► SQL dotaz:

```
select * from "User" u
join User_Project up on u.idu = up.idu
join Project p on up.idp = p.idp;
```

### ► Výsledek:

| idu | fname | lname   | ... | idd | idu | idp | idp | name           | date_from  | ... |
|-----|-------|---------|-----|-----|-----|-----|-----|----------------|------------|-----|
| 1   | Jan   | Sobota  | ... | 4   | 1   | 1   | 1   | IS banky       | 2020-01-01 | ... |
| 1   | Jan   | Sobota  | ... | 4   | 1   | 2   | 2   | IS firmy       | 2022-01-01 | ... |
| 1   | Jan   | Sobota  | ... | 4   | 1   | 3   | 3   | IS úřadu       | 2024-01-01 | ... |
| 1   | Jan   | Sobota  | ... | 4   | 1   | 4   | 4   | Spisová služba | 2026-01-01 | ... |
| 2   | Yveta | Švecová | ... | 3   | 2   | 1   | 1   | IS banky       | 2020-01-01 | ... |
| 2   | Yveta | Švecová | ... | 3   | 2   | 2   | 2   | IS firmy       | 2022-01-01 | ... |

### ► Pozorování:

- Počet záznamů ve výsledku je stejný (6) jako počet záznamů vazební tabulky, jelikož každý záznam vazební tabulky odkazuje na jeden záznam tabulky User a jeden záznam tabulky Project.

# Komutativita operace vnitřního spojení

- ▶ Jelikož operace vnitřní spojení je **komutativní**, můžeme dotaz uchopit z druhé strany: „**vrať projekty a jejich uživatele**“.
- ▶ Což intuitivně povede na jiné pořadí spojení tabulek:

```
select * from Project p
join User_Project up on p.idp = up.idp
join "User" u on up.idu = u.idu;
```

- ▶ Se stejným výsledkem (jen s jiným pořadím atributů):

| idp | name           | date_from  | ... | idu | idp | idu | fname | lname   | ... | idd |
|-----|----------------|------------|-----|-----|-----|-----|-------|---------|-----|-----|
| 1   | IS banky       | 2020-01-01 | ... | 1   | 1   | 1   | Jan   | Sobota  | ... | 4   |
| 2   | IS firmy       | 2022-01-01 | ... | 1   | 2   | 1   | Jan   | Sobota  | ... | 4   |
| 3   | IS úřadu       | 2024-01-01 | ... | 1   | 3   | 1   | Jan   | Sobota  | ... | 4   |
| 4   | Spisová služba | 2026-01-01 | ... | 1   | 4   | 1   | Jan   | Sobota  | ... | 4   |
| 1   | IS banky       | 2020-01-01 | ... | 2   | 1   | 2   | Yveta | Švecová | ... | 3   |
| 2   | IS firmy       | 2022-01-01 | ... | 2   | 2   | 2   | Yveta | Švecová | ... | 3   |

## Chyby ve spojovací podmínce

- ▶ Pozor na chybné primární / cizí klíče ve spojovací podmínce!
- ▶ Ignorování vazební tabulky:

```
select * from "User" u
join Project p on u.idu = p.idp;
```

Vede na **chybný výsledek**, protože v podmínce porovnáváme id uživatele s id projektu.

- ▶ Podobně, **drobná chyba** ve spojovací podmínce  
User\_Project ⋈ Project:

```
select * from "User" u
join User_Project up on u.idu = up.idu
join Project p on up.idu = p.idp;
```

Vede také na **chybný výsledek**, protože v podmínce opět porovnáváme id uživatele s id projektu namísto `up.idp=p.idp`

## Počet operací spojení v dotazu je libovolný I

- ▶ Při znalosti operace spojení a jejího výsledku, pak můžeme přidat další spojení.
- ▶ Uvažujme dotaz: „**vrať uživatele, jejich oddělení a jejich projekty**“.
- ▶ Postupně tedy spojíme tabulky a výsledky spojení:

$((\text{User} \bowtie \text{Department}) \bowtie \text{User\_Project}) \bowtie \text{Project}$

do následujícího SQL dotazu:

```
select * 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;
```

## Počet operací spojení v dotazu je libovolný II

### ► Výsledek:

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

### ► Výsledek obsahuje opět 6 záznamů:

- Protože všichni uživatelé, kteří mají nějaké projekty, jsou také v nějakém oddělení.
- Připomeňme, že do výsledku spojení jsou zařazeny **pouze dvojice záznamů** z obou spojovaných tabulek, pro které je spojovací podmínka vyhodnocena jako true.

## Počet operací spojení v dotazu je libovolný III

- ▶ **Pořadí operací join** můžeme v takovém dotazu **měnit jen do určité míry**, musíme například zachovat `User ⋈ User_Project` před `User_Project ⋈ Project`, abychom nejprve získali idp projektů uživatelů před samotnými záznamy projektů.
- ▶ `User ⋈ Department` můžeme v dotazu uvést jako poslední, nikoli jako první, jelikož záznamy oddělení pro spojování s `User_Project` a `Project` nepotřebujeme; přesně tedy můžeme provést spojení v tomto pořadí:  
 $((User \bowtie User\_Project) \bowtie Project) \bowtie Department$
- ▶ **Dotaz:**  

```
select * from "User" u
join User_Project up on u.idu = up.idu
join Project p on up.idp = p.idp
join Department d on u.idd = d.idd;
```
- ▶ Výsledek bude stejný, jen hodnoty atributů `Department` budou uvedeny jako poslední.

# Počet operací spojení v dotazu je libovolný IV

- Nastavením oddělení uživatele s idu=1 z idd=4 na null:

```
update "User" set idd=null where idu=1;
```

- Dostaneme ve výsledku stejného dotazu pouze 2 záznamy:

| idu | fname | lname   | ... | idd | idd | name        | idu | idp | idp | name     | ... |
|-----|-------|---------|-----|-----|-----|-------------|-----|-----|-----|----------|-----|
| 2   | Yveta | Švecová | ... | 3   | 3   | IT oddělení | 2   | 1   | 1   | IS banky | ... |
| 2   | Yveta | Švecová | ... | 3   | 3   | IT oddělení | 2   | 2   | 2   | IS firmy | ... |

- Protože záznam uživatele s idu=1 vyřadí už první spojení User ⋈ Department, tento uživatel nemá žádné oddělení: porovnání s idd=null není nikdy true, ale hodnota unknown.

# Projekce

- ▶ **Projekce**, kterou píšeme mezi klíčová slova `select` a `from`, značí definici atributů výsledku dotazu:

```
select <projekce> from <table>  
where <selekce>;
```

- ▶ Zatím jsme použili `*` pro všechny atributy nebo agregační funkci `count(*)` pro počet záznamů výsledku.
- ▶ Ačkoli projekce může být překvapivě komplikovaná, jak si ukážeme později, nyní si ukážeme přidání jednotlivých atributů nebo konstrukci `<tabulka>.*` pro všechny atributy tabulky `<tabulka>`.

## Projekce, dotazy I

- ▶ Uvažujme dotaz: „**vrať id, jména a příjmení uživatelů a id a názvy jejich projektů**“.
- ▶ Jedná se o drobnou modifikaci dotazu: „**vrať uživatele a jejich projekty**“.
- ▶ Do předchozího řešení tedy pouze přidáme projekci na požadovaných pět atributů:

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

- ▶ **Výsledek:**

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

## Projekce, dotazy II

- Stejný dotaz, ale chceme vypsát hodnoty všech atributů tabulky Project:

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

- Výsledek:

| idu | fname | lname   | idp | name           | date_from  | date_to    |
|-----|-------|---------|-----|----------------|------------|------------|
| 1   | Jan   | Sobota  | 1   | IS banky       | 2020-01-01 | 2021-12-31 |
| 1   | Jan   | Sobota  | 2   | IS firmy       | 2022-01-01 | 2023-12-31 |
| 1   | Jan   | Sobota  | 3   | IS úřadu       | 2024-01-01 | 2025-12-31 |
| 1   | Jan   | Sobota  | 4   | Spisová služba | 2026-01-01 | NULL       |
| 2   | Yveta | Švecová | 1   | IS banky       | 2020-01-01 | 2021-12-31 |
| 2   | Yveta | Švecová | 2   | IS firmy       | 2022-01-01 | 2023-12-31 |

## Projekce, dotazy III

- Stejný dotaz, ale ve výsledku chceme vytvořit nový atribut username **konkatenací** (`||` – spojení řetězců<sup>2</sup>) hodnot atributů `User.fname` a `User.lname`:

```
select u.idu, (u.fname || ' ' || u.lname) username, p.*
from "User" u
join User_Project up on u.idu = up.idu
join Project p on up.idp = p.idp;
```

- **Výsledek:**

| idu | username      | idp | name           | date_from  | date_to    |
|-----|---------------|-----|----------------|------------|------------|
| 1   | Jan Sobota    | 1   | IS banky       | 2020-01-01 | 2021-12-31 |
| 1   | Jan Sobota    | 2   | IS firmy       | 2022-01-01 | 2023-12-31 |
| 1   | Jan Sobota    | 3   | IS úřadu       | 2024-01-01 | 2025-12-31 |
| 1   | Jan Sobota    | 4   | Spisová služba | 2026-01-01 | NULL       |
| 2   | Yveta Švecová | 1   | IS banky       | 2020-01-01 | 2021-12-31 |
| 2   | Yveta Švecová | 2   | IS firmy       | 2022-01-01 | 2023-12-31 |

---

<sup>2</sup>SQL Server používá operátor `+`