

Databáze – Přednáška 2: Vazba 1:N

Michal Krátký

Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

2026/2027

Obsah přednášky

1. Vazba mezi entitními typy
2. Vazba 1:N v ERD
3. Implementace vazby 1 : 1 a 1 : N v relačním datovém modelu
4. Implementace vazby v SQL
5. Spojení záznamů tabulek v SQL

Vazba mezi entitními typy

- ▶ V konceptuálním modelu se kromě entitních typů objevují ještě tzv. **vazby** mezi entitními typy – **vztahy mezi entitami**, které vyplývají z reality (zaměstnanec pracuje ve firmě, student studuje na fakultě, student studuje studijní program apod.).
- ▶ **Mějme rozšířené zadání z 1. přednášky:** V informačním systému budeme evidovat jednotlivé **uživatelé**. 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.

Lineární zápis konceptuálního modelu

- ▶ Na základě zadání tedy máme tento **lineární zápis konceptuálního modelu** (který obsahuje **entitní typy** a jejich **atributy**):

User(idu, fname, lname, street, street_no, city,
email)

Department(idd, name)

Entitní typ vs entita

► S těmito příklady konkrétních entit:

► pro entitní typ User:

```
-- (idu, fname, lname, street, street_no, city, email)
(1, 'Jan', 'Sobota', '17. listopadu', 15, 'Olomouc',
 'jan.sobota01@upol.cz')
(2, 'Yveta', 'Švecová', 'Hněvotínská', 217, 'Olomouc',
 'yveta.svecova01@upol.cz')
(3, 'Petr', 'Švec', 'Hněvotínská', 217, 'Olomouc',
 'petr.svec02@upol.cz')
```

► a entitní typ Department:

```
-- (idd, name)
(1, 'Vedení')
(2, 'Účetní oddělení')
(3, 'IT oddělení')
(4, 'Výroba')
```

Vazba mezi entitní typy

- ▶ V našem zadání může mít konkrétní entita typu User vztah k nějaké entitě typu Department. Mezi entitními typy User a Department je tedy **vazba**.
- ▶ V **lineárním zápisu** definujeme vazbu jejím názvem (velkými písmeny) a entitními typy, které jsou členy této vazby:

```
User(idu, fname, lname, street, street_no, city,  
email)
```

```
Department(idd, name)
```

```
IS_MEMBER_OF(User, Department)
```

- ▶ Všimněme si vybraného **směru vazby**: *uživatel je členem oddělení*, můžeme ale zvolit i opačný směr od Department k User: *pracoviště má zaměstnance*, HAS_MEMBER(Department, User), což ale nemusí být tak intuitivní.

Jazykové okénko: chybné označení vazby

- ▶ **Častou chybou** je označovat vazbu jako relaci. V relačním datovém modelu je ale relace tabulka, nikoli vazba mezi tabulkami.
- ▶ **Stejná chyba** se vyskytuje i v anglicky mluvícím prostředí. Například v nástroji Oracle Modeler se vazba označuje jako Relation namísto Relationship.

Kardinalita vazby – povinnost členství ve vazbě

- ▶ Parametrem vazby je **kardinalita**, která se specifikuje samostatně pro **oba entitní typy vazby**: znamená **minimální a maximální počet** entit druhého typu, se kterými entita daného typu **má vztah**.
- ▶ **Minimální počet entit značí povinnost členství** ve vazbě:
 - ▶ 0 značí **nepovinné členství** (*optional*): může existovat entita daného entitního typu, která **nemá vztah** s nějakou entitou druhého typu. Což ale nevylučuje existenci entity se vztahem k entitě druhého typu.
 - ▶ 1 značí **povinné členství** (*mandatory*): každá entita daného entitního typu **má vztah** s nějakou entitou druhého typu. Což vylučuje existenci entity bez vztahu k entitě druhého typu.

Příklad: povinnost členství ve vazbě

- ▶ V našem zadání je uvedeno: Každý uživatel je ve firmě pracovníkem **nejvýše** jednoho **oddělení** . . . , což znamená *nepovinnost* User ve vazbě IS_MEMBER_OF.
 - ▶ Srovnajme se zadáním: Každý uživatel je ve firmě pracovníkem **právě** jednoho **oddělení** . . . , které značí povinnost členství User v IS_MEMBER_OF.
- ▶ Ze zadání vyplývá, že pracoviště, například právě vytvořené, nemá žádného pracovníka, což značí *nepovinnost* členství Department v IS_MEMBER_OF.

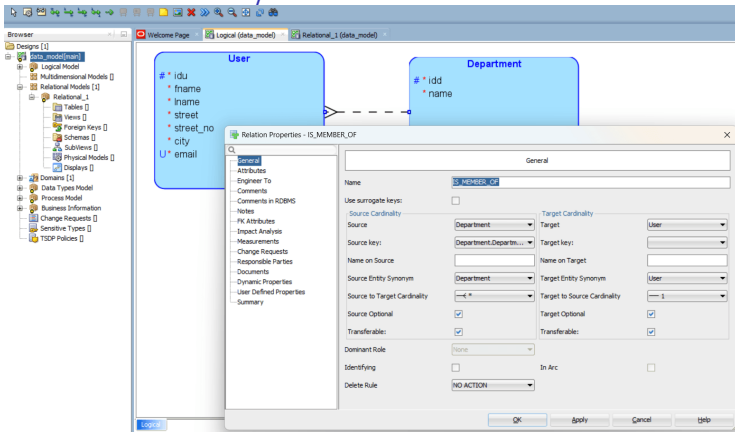
Kardinalita vazby

- ▶ Jelikož minimální kardinalita (tedy povinnost členství ve vazbě) je v SQL implementována jinak než maximální kardinalita, budeme v souvislosti s **kardinalitou většinou spíše uvažovat maximální kardinalitu**.
- ▶ Rozlišujeme tyto kardinality: **1:1**, **1:N** a **M:N** (*one-to-one*, *one-to-many*, *many-to-many*).
- ▶ Abychom dobře stanovili kardinalitu vazby, musíme se na tento počet zeptat z obou stran (entitních typů) vazby.
 - ▶ Z obou stran zjišťujeme, zda může mít entita vztah s **jednou nebo více (N)** entitami druhého entitního typu.

Příklad: kardinalita vazby

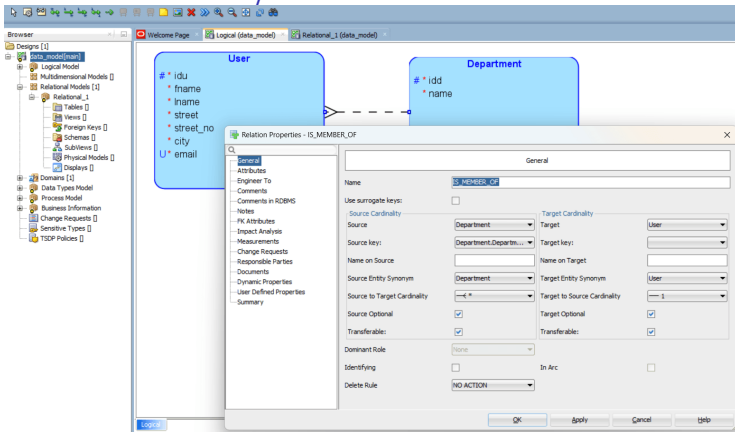
- ▶ **Zadání:** Každý uživatel je ve firmě pracovníkem nejvýše jednoho **oddělení**, u kterého chceme evidovat jedinečné číslo a název (který je povinný). Oddělení může mít více pracovníků.
 - ▶ **Vazba:** IS_MEMBER_OF(User, Department)
 - ▶ **Určení kardinality:**
 - ▶ Uživatel může pracovat nejvýše v **jednom** oddělení, oddělení může mít **více** uživatelů. $\Rightarrow$ kardinalita **1:N**.
- ▶ **Zadání:** Uživatel může být ve firmě pracovníkem více **oddělení**, u každého Oddělení může mít více pracovníků.
 - ▶ **Vazba:** IS_MEMBER_OF(User, Department)
 - ▶ **Určení kardinality:**
 - ▶ Uživatel může pracovat ve **více** odděleních, oddělení může mít **více** uživatelů. $\Rightarrow$ kardinalita **M:N**.
- ▶ **Častá chyba:** Zjednodušení úvahy (např. "více uživatelů pracuje ve více odděleních") má za následek chybné určení kardinality.

Vazba 1:N v ERD 1/4



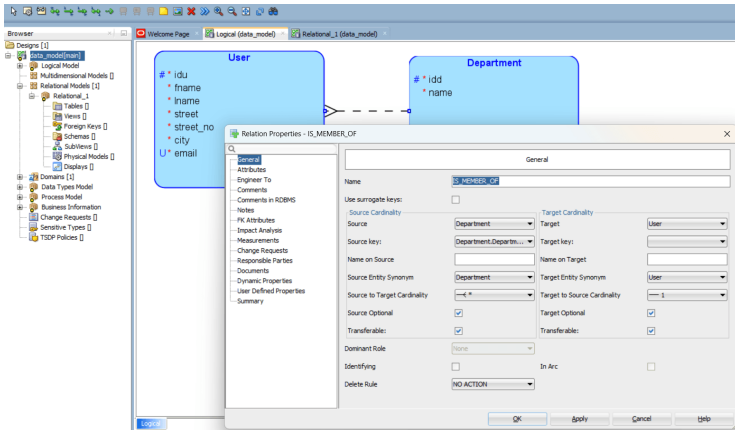
- ▶ V nástroji Oracle Modeler najdeme vazbu 1:N pod ikonou New 1:N Relation.
- ▶ Jelikož pracoviště **má více** uživatelů, táhneme vazbu od Department k User.

Vazba 1:N v ERD 2/4



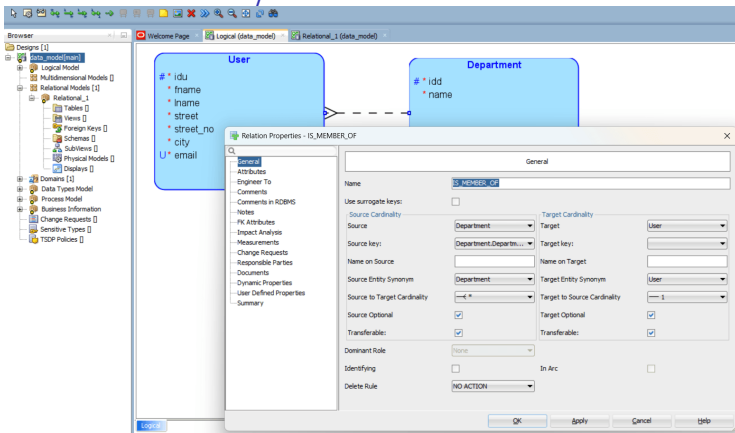
- Napíšeme název vazby IS_MEMBER_OF.
- Source to Target Cardinality u Source (Department) obsahuje dvojici symbolů pro kardinalitu N (tzv. **vrání nohu** – **Crow's foot** a symbol *).

Vazba 1:N v ERD 3/4



- Naopak Target to Source Cardinality u Target (User) obsahuje dvojici symbolů pro kardinalitu 1 (symbol – a symbol 1).

Vazba 1:N v ERD 4/4



- ▶ Volby Source Optional i Target Optional jsou zatrženy.
- ▶ Nepovinnost členství obou entitních typů ve vazbě je v Barkerově notaci ERD znázorněna **čárkovanou čarou** grafického symbolu vazby (ten má tedy dvě poloviny).

Implementace vazby 1 : 1 a 1 : N v relačním modelu I

- ▶ V relačním datovém modelu je **vazba 1:1 nebo 1:N** implementována pomocí **cizího klíče** – atributu, který se **odkazuje** na nějaký **klíč** druhé tabulky (primární klíč nebo jedinečný atribut):
- ▶ Hodnota cizího klíče pak **odpovídá** hodnotě tohoto klíče druhé tabulky: přes cizí klíč se tedy záznam **odkazuje** na záznam druhé tabulky.

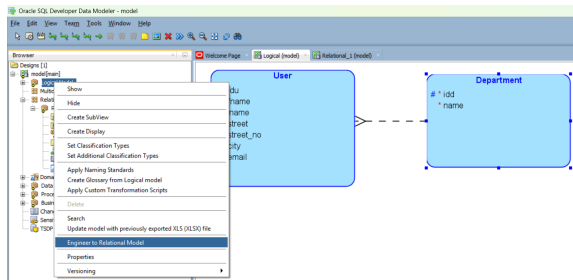
Implementace vazby 1 : 1 a 1 : N v relačním modelu II

► Umístění cizího klíče:

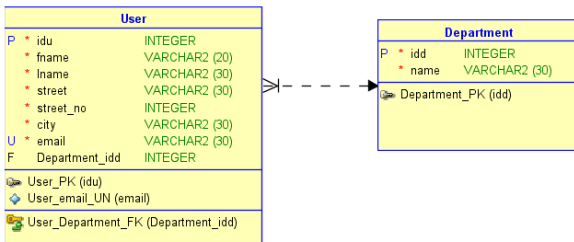
- **U vazby 1 : 1** intuitivně umístíme **cizí klíč** do jedné z tabulek vazby:
 - V tomto případě je cizí klíč **jedinečný**, může odkazovat na nejvýše jeden záznam druhé tabulky.
- **U vazby 1 : N** umístíme **cizí klíč** do tabulky **na straně N**. Cizí klíč **není jedinečný** – tabulka na straně N může obsahovat stejné hodnoty cizího klíče, tzn.:
 - Jeden záznam z tabulky na straně 1 může být odkazován více záznamy z tabulky na straně N.
 - Jeden záznam z tabulky na straně N se odkazuje na nejvýše jeden záznam tabulky na straně 1 (odkazovaný atribut je klíč).

ERD relačního datového modelu I

- ▶ Entity Relationship Diagram (**ERD**) používáme i pro vizualizaci **relačního datového modelu**.
- ▶ Nástroje pro konceptuální modelování proto často umožňují automaticky vytvořit **ERD relačního datového modelu** z konceptuálního modelu:
 - ▶ V *Oracle Modeler* použijeme položku *Engineer to Relational Model* z kontextového menu pro konceptuální model (Logical Model) a tlačítko *Engineer*.

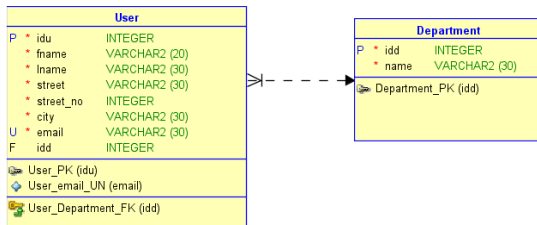


ERD relačního datového modelu II



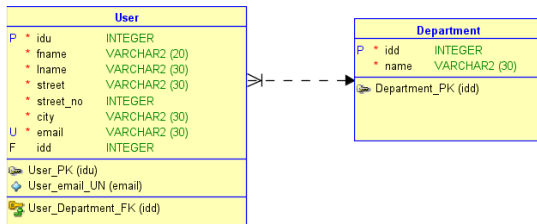
- ▶ V relačním datovém modelu nyní vidíme:
 - ▶ Nový atribut Department_idd v tabulce User, který je označen jako **F** (tedy **Foreign key**, tedy **cizí klíč**).
 - ▶ Je zvykem pojmenovat cizí klíč stejně jako klíč ve druhé tabulce vazby, což jednoduše v nástroji Oracle Modeler provedeme změnou v položce *Columns* v okně specifikace tabulky: cizí klíč pojmenujeme idd.

ERD relačního datového modelu III



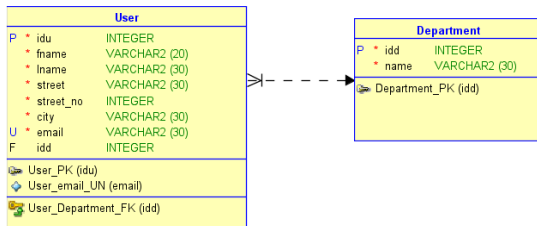
- ▶ V relačním datovém modelu nyní vidíme:
 - ▶ Pod specifikací atributů jsou jednotlivá **integritní omezení (IO)**:
 - ▶ *** _PK**: entitní IO **primary key** (tedy primární klíč),
 - ▶ *** _UN**: entitní IO **unique** (tedy jedinečnost atributu),
 - ▶ *** _FK**: nové, tzv. **referenční IO**, **foreign key** (tedy cizí klíč), v tomto případě implementuje vazbu 1 : N.

ERD relačního datového modelu IV



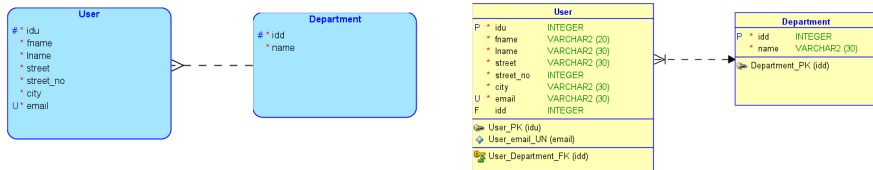
- ▶ V relačním datovém modelu nyní vidíme:
 - ▶ **Specifikace integritních omezení** – entitních (*primary key*, *unique*) i referenčních (*foreign key*) – se opakují: jsou uvedeny přímo u jednotlivých atributů a znovu v souhrnném seznamu pod atributy.
 - ▶ Jedná se o náznak dvojího způsobu definice integritních omezení v příkazu `create table` (inline u atributu vs. samostatně na konci).

ERD relačního datového modelu V



- ▶ V relačním datovém modelu nyní vidíme:
 - ▶ Jelikož v konceptuálním modelu jsme **povinnost v členství** obou entitních typů ve vazbě uvedli jako nepovinné, cizí klíč `User.idd` je označen jako nepovinný.
 - ▶ Přesně tedy vyjadřujeme, že uživatel nemusí mít oddělení. V relačním datovém modelu však nemáme prostředky pro vyjádření případné povinné účasti na straně 1 (tj. že oddělení musí mít alespoň jednoho uživatele).

ERD relačního datového modelu VI



- Oracle Modeler pro zakreslení vazeb mezi tabulkami relačního datového modelu používá nyní **jinou notaci ERD** (zvanou *Information Engineering*):
 - Symbol pro N z konceptuálního modelu, **vraní noha (Crow's Foot)**, obsahuje navíc svislou čáru.
 - Symbol pro 1 z konceptuálního modelu je nyní **plná černá šipka**.
 - Naznačení povinnosti/nepovinnosti ve vazbě **zůstává** pomocí plné/čárkované čáry.

Vytvoření tabulky Department

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

```
create table Department  
(  
    id int primary key,  
    name varchar(30) not null  
);
```

Vložení více záznamů do tabulky

- Vložíme několik záznamů pomocí SQL příkazu pro manipulaci s daty (JMD/DML) insert, použijeme variantu pro vložení více záznamů:

```
insert into Department(idd, name)  
values  
    (1, 'Vedení'), (2, 'Účetní oddělení'),  
    (3, 'IT oddělení'), (4, 'Výroba');
```

- **Poznámka ke specifikaci SQL:** Ačkoli je možnost vložení více záznamů jedním příkazem insert definována již ve specifikaci **SQL92**, Microsoft SQL Server ji podporuje od verze 2008 a Oracle Database dokonce až od verze 23c.

Implementace vazby v SQL I

- ▶ **Referenční integritní omezení**, tedy naznačení, že atribut je **cizí klíč**, se v SQL, při specifikaci atributu, zapisuje pomocí:

`references <table_name>(attribute_name)`

- ▶ Pro již vytvořenou tabulku můžeme nový atribut, **cizí klíč**, vložit pomocí příkazu jazyka pro definici dat (JDD/DDL)
`alter table:`

```
alter table "User" add column idd int  
references Department(idd);
```

Implementace vazby v SQL II

- Pokud definujeme cizí klíč přímo v příkazu jazyka pro definici dat (JDD/DDDL) create table, pak jednoduše píšeme:

```
create table "User"  
(  
    idu int primary key,  
    fname varchar(20) not null,  
    lname varchar(30) not null,  
    street varchar(30) not null,  
    street_no int not null,  
    city varchar(30) not null,  
    email varchar(30) unique not null,  
    idd int references Department(idd)  
);
```

Hodnota cizího klíče po přidání atributu

- Hodnota cizího klíče je nyní null:

```
select * from "User";
```

```
-- idu  fname  lname  street  street_no  city  email  idd
-- 1     Jan    Sobota 17. listopadu 15  Olomouc jan.sobota01@upol.cz  NULL
-- 2     Yveta  Švecová Hněvotínská 215  Olomouc yveta.svecova01@upol.cz  NULL
-- 3     Petr    Švec    Hněvotínská 217  Olomouc petr.svec02@upol.cz  NULL
```

- Přidáme tedy dvěma uživatelům jejich oddělení¹:

```
update "User" set idd=4 where idu=1;
update "User" set idd=3 where idu=2;
```

```
-- idu  fname  lname  street  street_no  city  email  idd
-- 3     Petr    Švec    Hněvotínská 217  Olomouc petr.svec02@upol.cz  NULL
-- 1     Jan    Sobota 17. listopadu 15  Olomouc jan.sobota01@upol.cz  4
-- 2     Yveta  Švecová Hněvotínská 215  Olomouc yveta.svecova01@upol.cz  3
```

¹Příkazy jsou odděleny symbolem ; můžeme je tedy oba označit a spustit najednou.

Důsledek referenční integrity pro mazání záznamů

- DBS nyní hlídá nastavené **referenční IO** (nebo **referenční integritu**): pokud se pokusíme smazat záznam z tabulky Department, na který se odkazuje nějaký záznam z tabulky User přes cizí klíč, DBS to nepovolí.

```
delete from Department where idd=4;
```

```
-- update or delete on table "department" violates foreign key
-- constraint "User_idd_fkey" on table "User"
-- DETAIL: Key (idd)=(4) is still referenced from table "User".
```

- Smazání záznamu z Department, na který se žádný záznam z User neodkazuje, ale nic nebrání:

```
delete from Department where idd=2;
-- DELETE 1
```

Spojení záznamů tabulek v SQL I

- ▶ V SQL spojujeme záznamy dvou tabulek konstrukcí **join** (**spojení**) příkazu jazyka pro dotazování dat (DQL) **select**:

```
select * from <table1>  
join <table2> on <boolean condition>;
```

- ▶ Výsledkem je **tabulka**, která nejprve obsahuje atributy tabulky `table1` a pak atributy tabulky `table2`, takové že ve výsledku jsou obsaženy všechny možné dvojice záznamů těchto tabulek, pro které je výraz **boolean condition** vyhodnocen jako **true**.

Spojení záznamů tabulek v SQL II

- ▶ Nemělo by nás tedy překvapit, že následujícím dotazem získáme kartézský součin tabulek User a Department²:

```
select * from "User" u  
join Department d on true;  
-- tedy 3 x 4 = 12 záznamů
```

- ▶ Ačkoliv se kartézský součin tabulek User a Department může v SQL zapsat i jednodušeji takto:

```
select * from "User", Department;
```

²Všimněme si použití zkratk jmen tabulek u a d.

Spojení záznamů tabulek ve vazbě

- Spojení se typicky používá pro **spojení dvojic záznamů** ve vztahu z **tabulek ve vazbě**. Booleovská podmínka za on tedy obsahuje porovnání cizího klíče z jedné tabulky s klíčem tabulky druhé.
- **Dotaz:** Vypište všechny uživatele a jejich oddělení.

```
select * from "User" u
join Department d on u.idd=d.idd;
```

```
-- idu  fname  lname  street  street_no  ...  idd  idd  name
-- 2     Yveta  Švecová Hněvotínská  215  ...  3    3    IT oddělení
-- 1     Jan    Sobota  17. listopadu  15   ...  4    4    Výroba
```

- Proč výsledek **neobsahuje** záznam třetího uživatele s `idd=null`? Výsledkem porovnání s hodnotou `null` nemůže být nikdy `true`. Zjednodušeně tedy můžeme říct, že výsledek obsahuje **pouze uživatele, kteří mají nějaké oddělení**.

Spojení záznamů tabulek ve vazbě II

- ▶ Z požadavku na hodnotu true podmínky za on vyplývá:
- ▶ Operace se přesně jmenuje **vnitřní spojení (inner join)**³.
Později si ukážeme i další typy operace spojení.
- ▶ Vnitřní spojení je **komutativní**, následující dotazy vrací stejný výsledek, který se **liší** pouze **pořadím atributů** obou tabulek:

```
select * from "User" u
join Department d on u.idd=d.idd;
-- idu  fname  lname  ...  idd  idd  name
-- 2     Yveta  Švecová ...  3     3     IT oddělení
-- 1     Jan    Sobota  ...  4     4     Výroba
```

```
select * from Department d
join "User" u on d.idd=u.idd;
-- idd  name  idu  fname  lname  ...  idd
-- 3     IT oddělení  2     Yveta  Švecová ...  3
-- 4     Výroba      1     Jan    Sobota  ...  4
```

- ▶ Stejně tak nezáleží na pořadí atributů v podmínce za on.

³V SQL příkazu také můžeme psát inner join namísto join. 

Vazba 1 : N a výsledek spojení

- ▶ Pokud spojujeme tabulky s vazbou 1 : N , ve výsledku dochází k opakování záznamů ze strany 1 (tedy záznamů, na které se odkazuje více řádků z druhé tabulky).

- ▶ Nejprve přidáme uživatele s idu=3 do oddělení 4:

```
update "User" set idd=4 where idu=3;
```

- ▶ Ve výsledku dotazu se 2× opakuje oddělení (4, 'Výroba'), protože jsou k němu přiřazeni dva uživatelé:

```
select * from "User" u
join Department d on u.idd=d.idd;
```

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