

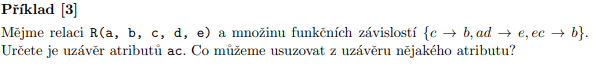
**Jde o natural join, kde se vybírají řádky, kde se rovnají sloupce a v nich hodnoty. Tudíž, jediný takový řádek se jménem Tony, kde se i R2.b rovná R1.b je -> Tony 26 13**



**SELECT R1.a, R1.c, R2.d FROM**

**R1 JOIN R2 ON R1.a = R2.a AND**

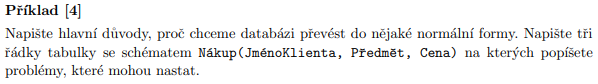
**R1.b = R2.b**



**c -> b**

**ad -> e ac += {a, c, b} Pokud určíme uzávěr, který určuje celý záznam**

**ec -> b dokážeme se dobrat klíče a určit MNF**



**Normalizací modelu databáze, myslíme reorganizaci struktury a dat v databázi, tak abychom zvýšili efektivitu, snížili redundanci dat a předešli nekonzistenci dat.**

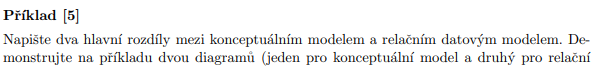
**Jméno klienta Předmět Cena**

***Novák Pračka 1000***

***Novák Pračka 7000* -> pokud bychom smazali nováka takzůstane**

***Jarmil Počítač 2000 záznam nad ním zůstane = nekonzistence***

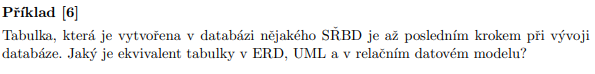
**Dále vidíme, že se data v modelu opakují = redundance a v neposlední řadě, nemáme ani jasný identifikátor záznamu v db.**



**Konceptuální model = návrh databáze**

**Relační model = konkrétní implementace, jak bude databáze implementována na serveru**

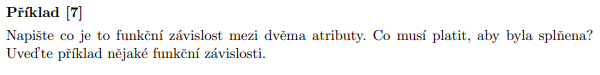
**Konceptuální model neobsahuje cizí klíče a vazby M:N v něm mohou být realizovány pouhým vztahem, zatímco Relační cizí klíče obsahuje, obsahuje indexaci v modelu, obsahuje konkrétní datové typy atributů a vazba M:N je realizováná vazební tabulkou.**



**ERD – ENTITA**

**UML – CLASS (TŘÍDA)**

**RELAČNÍ DM – RELACE**



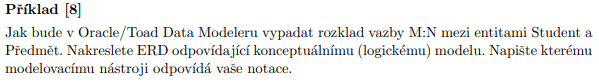
**Př. Rodne\_cislo -> jmeno, prijmeni**

**Pro každé rodné číslo existuje jedno jméno a příjmení**

**Funkční závislost, je závislost atributů mezi sebou v rámci jedné tabulky/relace.**

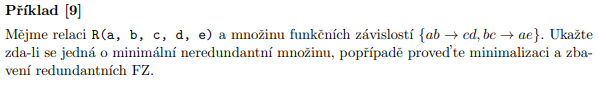
**Pomocí funkčních závislostí, můžeme (a chceme) docílit určení klíče, tedy atributu, kterým dokážeme vyjádřit jednoznačně celý záznam v tabulce.**

**Aby funkční závislost platila, musí zejména dávat smysl v reálném světě.**



**Student ---< Studuje >--- Předmět (Oracle)**

**M:N je rozložena na dvě vazby 1:N, vůči nové vazební tabulce, která vzniká tímto rozkladem.**



**Sestavíme hlavní uzávěr.**

**a, b, c, d += { a, b, c, d, e }**

**Postupným ubíráním z levých stran ověříme, zda nějaký z atributů na levé straně není redundantní.**

**a, b += { a, b, c, d }**

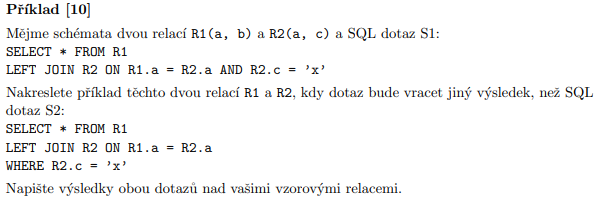
**b, c += { b, c, a, e** }

**ab -> cd bc-> ae**

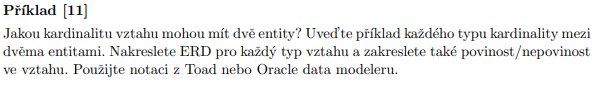
**a += { a , c, d } nelze odebrat b b += { a, e, b, c, d } c lze odebrat**

**b += { b, c, d, a, e } a lze odebrat c += {a, e, c } b nelze**

**Minimální neredundantní forma ( b -> cd, b -> ae ) = ( b -> aecd)**



**Pokud LEFT JOIN obsahuje podmínku where, tak se už nebude chovat jako LEFT JOIN, stává se z něj normální JOIN. Zatím co první dotaz vypíše všechny záznamy, kde R2.c je ‚x‘ i s NULL atributy.**



**Kardinality mohou být: 1:N, 1:1, M:N**

**Poté z hlediska povinností ve vztahu mohou být ještě povinné/nepovinné (mandatory)**

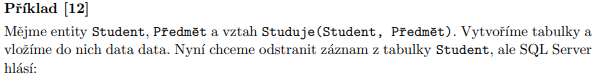
**1:N -> -----------< - nepovinnost -> - - - - ------<**

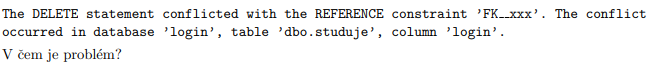
**1:1 -> ------------ - nepovinnost -> - - - - --------**

**M:N -> >---------< - nepovinnost -> >- - - ------<**

**------< |REL| >------**

**Nepovinnost samozřejmě může být oboustranná.**

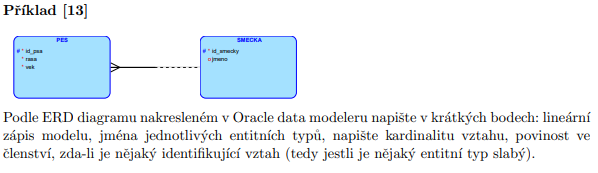




**Pokoušíme se odstranit záznam, na nějž je navázán záznam v tabulce studuje pomocí cizího klíče.**

**Tabulka studuje vznikla rozložením M:N vazby mezi Studentem a Předmětem a její primární klíč je tvořen složením cizích klíčů ze zmíněných tabulek.**

**Nastavením ON\_DELETE, by se této chybě dalo vyvarovat, protože by bylo stanoveno chování, při smazání záznamu, na který je jiný záznam navázán.**



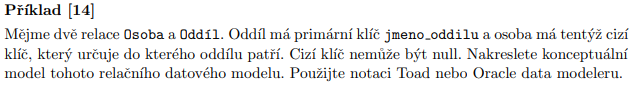
**Identifikující vztah zde není.**

**Vazba je: Smečka 1 : N Pes**

**Vazba by se četla, jako: Smečka nemusí mít žádného psa, ale pes musí mít smečku.**

**Pes(id\_psa, rasa, vek, *id\_smecky*)**

**Smecka(id\_smecky, jmeno)**



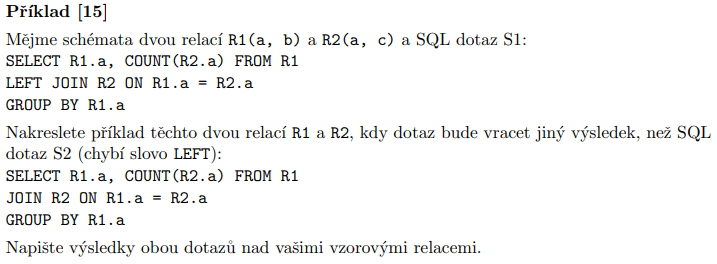
**Osoba Oddil**

**---------------------- -------------------------**

**# \*Id\_os | | #\*Jmeno\_oddilu**

**\*Jmeno |>--------------- |**

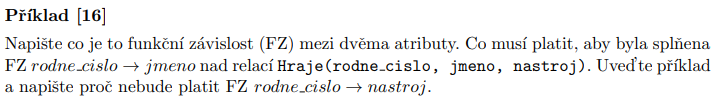
**\*Prijmeni | |**



**První vypíše i výsledky z druhé tabulky, kde některé hodnoty jsou NULL, stará se jen o „LEVOU“ tabulku.**

**Druhý vypíše jen FULL JOIN, kdy data jsou v obou tabulkách.**

**Dále je groupne podle R1.a**



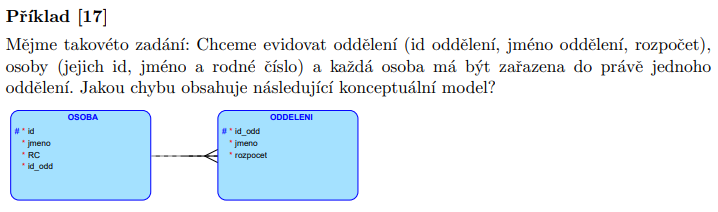
**Funkční závislost, je závislost atributů mezi sebou v rámci jedné tabulky/relace.**

**Pomocí funkčních závislostí, můžeme (a chceme) docílit určení klíče, tedy atributu, kterým dokážeme vyjádřit jednoznačně celý záznam v tabulce.**

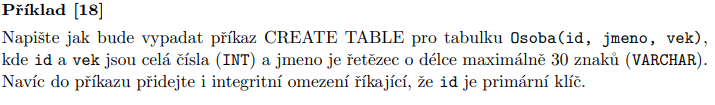
**Aby funkční závislost platila, musí zejména dávat smysl v reálném světě.**

**rodne\_cislo -> jmeno ,muselo by být jasně dáno, že podle rodného čísla se dohledám daného jména, což i v realitě platí a je to tudíž validní funkční závislost.**

**Rodne\_cislo -> nastroj, nastroj by musel být při narození přiřazen k osobě s daným rodným číslem, což nedává smysl, osoba muže hrát na více nástrojů, tudíž to není validní FZ muselo by dojít na dekompozici.**



**Kardinalita vztahu je naopak, oddělení se skládá z osob, ne osoba z oddělení, id\_odd by nemělo v konceptuálním modelu co dělat, vztah je nepovinný, měl by být povinný.**

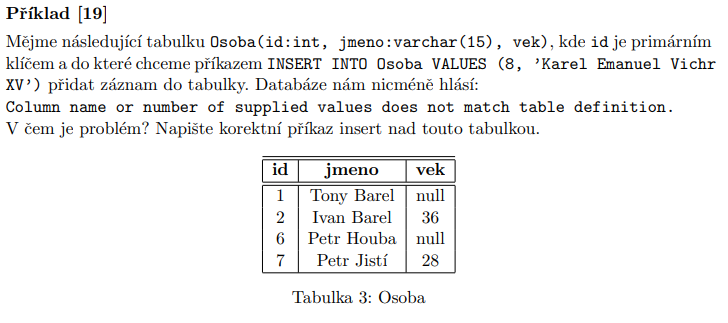


**CREATE TABLE Osoba (**

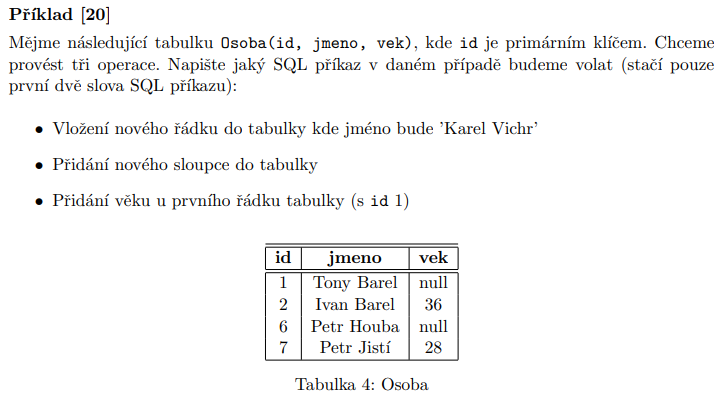
**Id INT primary key,**

**Jmeno VARCHAR(30),**

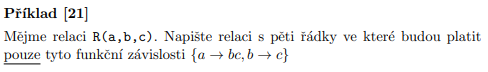
**Vek INT );**



**Pokud by bylo id nastaveno jako Auto increment, tak by vadilo počáteční číslo, jelikož nespecifikujeme pořadí, dle jakého chceme hodnoty do tabulky vkládat. V našem případě ovšem jde o to, že nevkládáme do tabulky potřebné 3 hodnoty, i hodnota null musí být vložena.**



* **INSERT INTO Osoba(8, Karel Vichr, 50);**
* **ALTER TABLE ADD … TYPE;**
* **UPDATE Osoba SET vek=… WHERE id=1;**



**a b c**

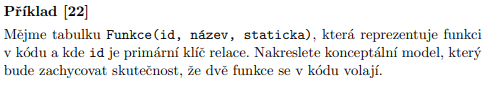
**1 73801 Frýdek-Místek**

**2 73102 Ostrava**

**3 78966 Opava**

**4 25666 Poruba**

**5 31288 Brno**



**Univerzální relační schéma** – „široká“ nepřehledná tabulka, která obsahuje všechny atributy (např. modelovaného systému)

**Funkční závislost** – ze znalosti hodnot nějaké množiny hodnot atributů (a samozřejmě obsahu databáze) znám i množinu hodnot jiných atributů.

**Armstrongovy axiomy** – odvozovací pravidla pro funkční závislosti.

**Uzávěry množiny atributů** – hodnoty kterých všech atributů jsem schopný získat na základě určité dané množiny atributů

**Klíč schématu** – atributy, kterými jednoznačně identifikuji celý záznam, tzn. pokud znám hodnoty těchto atributů, umím v univerzálním schématu dohledat obsah celého záznamu

**Minimální neredundantní pokrytí** – rozložím FZ na elementární FZ, z levých stran odstraním nadbytečné atributy, a nakonec celé nadbytečné FZ.