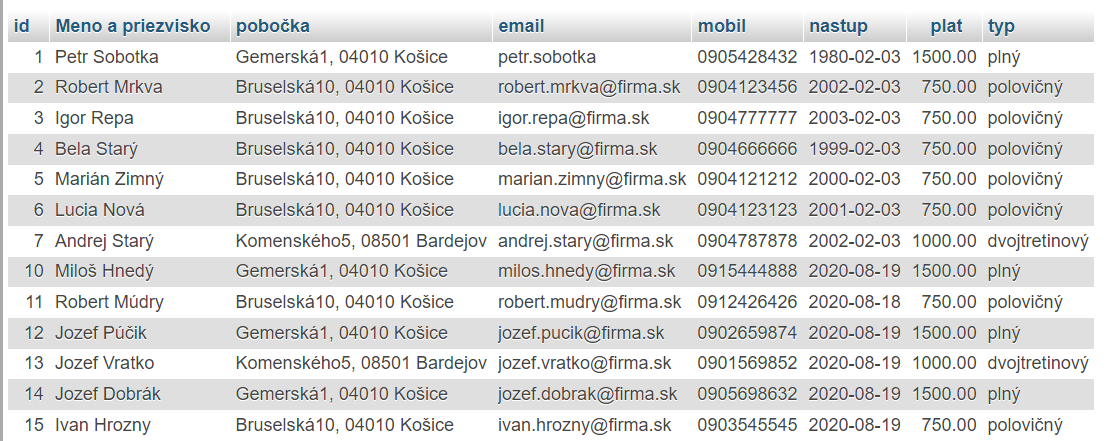
1.

Pracujte s pripraveným súborom zamestnanci.png. Obsahuje prehľad údajov, ktoré firma potrebuje evidovať o svojich zamestnancoch. Navrhnite normalizovanú databázu vo forme entitno – relačného diagramu aplikovaním troch základných normálových foriem.

Na príklade ERD diagramu vysvetlite dané pojmy: entita, atribút, dátové typy, relačné vzťahy, typy relačných vzťahov, primárny kľúč, cudzí kľúč.



**Normalizácia databázy**

**Správne vytvorené tabuľky spĺňajú 3 základné nomálové formy:**

1. **normálová forma (1NF)**

* hovorí, že všetky atribúty (stĺpce) tabuľky (entity) sú atomické, t.j. ďalej sa už nedajú deliť

Naša firma má všetky pobočky len v rámci jedného mesta a náš zákazník neuvažuje o rozšírení do iných miest.

Ak by sme v takejto tabuľke chceli vypísať všetkých zamestnancov, ktorých PSČ sa rovná určitej hodnote, nebolo by to jednoduché. Alebo keby sme chceli nájsť zamestnancov, ktorí pracujú na pobočke v Bardejove.

Je to preto, že atribút (stĺpec) „pobočka“ nie je atomický a skladá sa z niekoľkých častí:

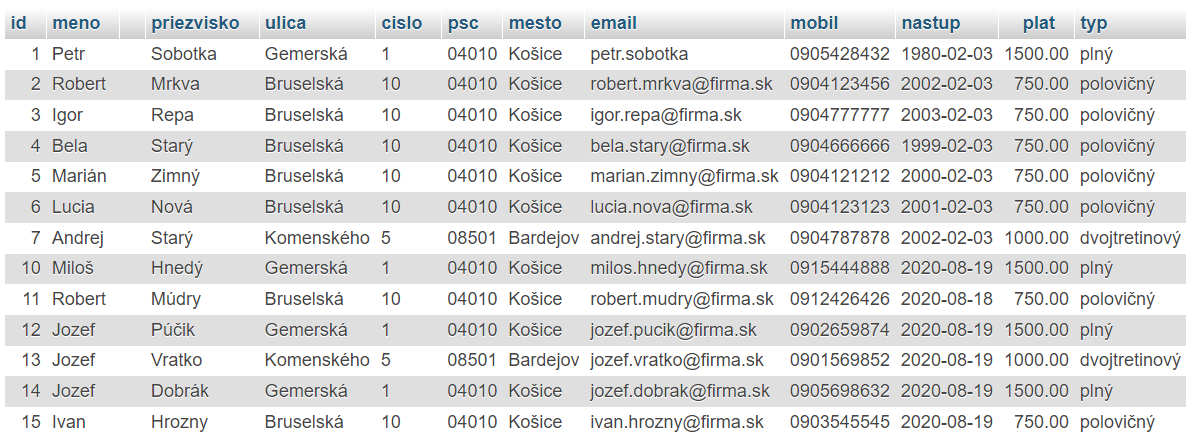
ulica,

číslo,

PSČ,

mesto.

Správny návrh tabuľky, ktorá bude rešpektovať 1NF by vyzeral nasledovne:



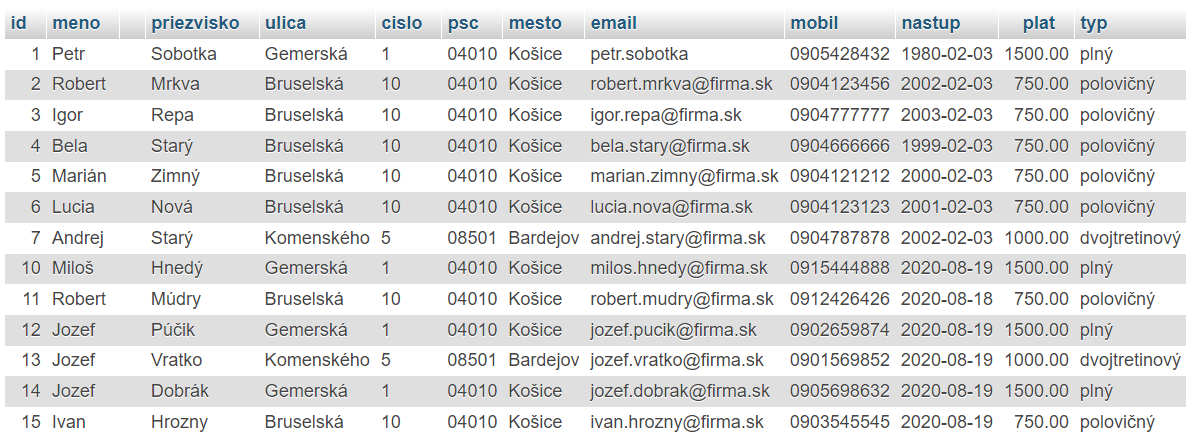
1. **normálová forma (2NF)**

* tabuľka spĺňa 2NF práve vtedy keď spĺňa 1NF a naviac každý atribút, ktorý nie je primárnym kľúčom je na primárnom kľúči úplne závislý,
* v našom prípade pobočka nie je nijako závislá na primárnom kľúči – id zamestnanca,
* ani popis úväzku nie je priamo závislý na type úväzku
* v takom prípade je riešením rozpad na viacero tabuliek – konkrétne vznikne tabuľka „pobočka“ a tabuľka „úväzok“
* v tabuľke pobočka je adresa pobočky závislá na primárnom kľúči – id,
* v tabuľke úväzok je typ úväzku závislý na primánom kľúči – id.

**zamestnanec**

**uvazok**

**pobocka**







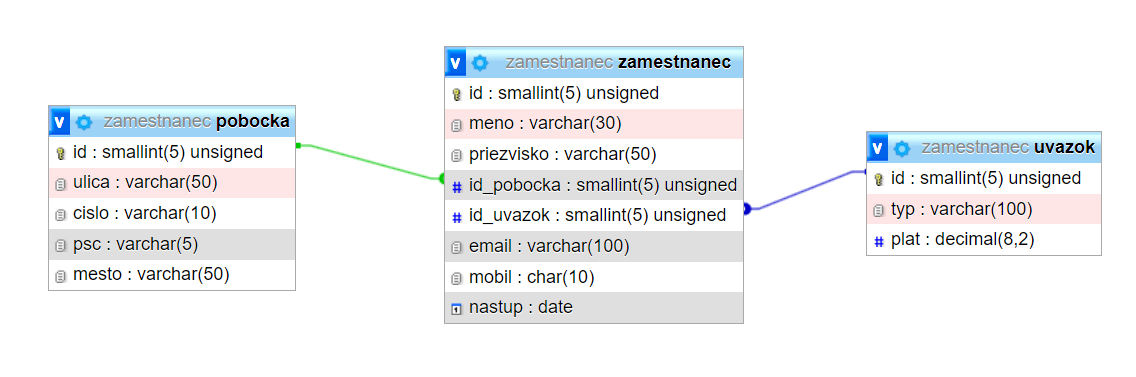
1. **normálová forma (3NF)**

* relačné tabuľky spĺňajú tretiu normálovú formu (3NF), ak spĺňajú 2NF a všetky nekľúčové atribúty musia byť navzájom nezávislé
* v tabuľke zamestnanec vidíme, že výška platu závisí od typu úväzku
* riešením je znova dekompozícia

**uvazok**



**Entitno-relačný diagram pre databázu zamestnanci:**



**Relačné databázy – základné pojmy**

Tri základné pojmy:

* entita (tabuľka - **Zamestnanec**)
* atribúty (stĺpce)
* relačné vzťahy – väzby (zamestnanec má jeden typ úväzku, ale jeden typ úväzku môže mať N zamestnancov, t.j. 1:N, zamestnanec pracuje na jedenej pobočke, ale na jednej pobočke môže pracovať N zamestnancov, t.j. 1:N )
* primárny kľúč (PRIMARY KEY) – je taká hodnota, ktorá nám jednoznačne určuje daný záznam
* cudzí kľúč (FOREIGN KEY) - cudzí kľúč v jednej tabuľke ukazuje na primárny kľúč v inej tabuľke

***Dátové typy:***

***Reťazcové dátové typy***

Najčastejšie používané

* **VARCHAR(x)** – slúži na ukladanie ľubovoľných reťazcov znakov (tzv. stringov) dĺžky najviac x
* **CHAR(x)** – sa tiež používa na ukladanie ľubovoľných reťazcov, tu však majú pevnú dĺžku x.

***Číselné dátové typy***

Najčastejšie používané

* **INT** (alebo neskrátené INTEGER) slúži na ukladanie celých čísel
* **DEC(x,y)** (alebo aj DECIMAL(x,y)) – desatinné čísla s celkovou dĺžkou (t.j. počtom cifier pred i za desatinnou čiarkou či bodkou)

***Dátové typy pre dátum a čas***

Najčastejšie používané

* **DATE** slúži na ukladanie dátumu vo formáte YYYY-MM-DD
* **DETETIME** na ukladanie dátumu a času : YYYY-MM-DD HH:MM:SS

**Relačné vzťahy - príklady**

**1:1**

* jednému záznamu v jednej tabuľke zodpovedá jeden záznam v inej tabuľke
* táto relácia sa veľmi nepoužíva, slúži skôr na sprehľadnenie tabuliek
* Napríklad: vzťah vodič – automobil – jeden vodič môže riadiť práve jeden automobil a jeden automobil môže byť riadený práve jedným vodičom

**1:N**

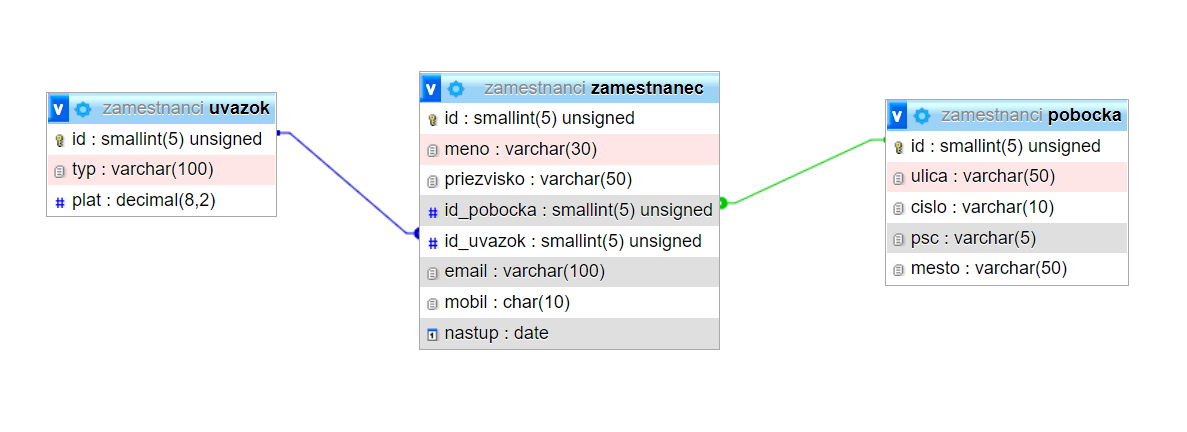
* jednému záznamu v jednej tabuľke zodpovedá N záznamov v inej tabuľke
* napr.: triedny učiteľ a študent ... 1 triedny učiteľ môže mať N študentov, ale 1 študent nemôže mať N triednych učiteľov
* Iný príklad: v autobuse môže naraz cestovať viac cestujúcich, ale jeden cestujúci môže v jednom okamihu cestovať iba jedným autobusom

**M:N**

* M záznamom v jednej tabuľke zodpovedá N záznamov v inej tabuľke
* napr.: študenti a predmet .... 1 študent študuje viacero predmetov, ale aj 1 predmet študuje viacero študentov
* Iný príklad: jeden autor môže napísať viac kníh a jednu knihu môže napísať viac autorov (kolektív autorov)

2.

Vytvorte databázu s názvom ‚zamestnanci‘ na lokálnom počítači. Na základe entitno-relačného diagramu ERD\_zamestnanci.png upravte súbory zamestnanec.txt, uvazok.txt a pobocka.txt. Naplňte danú databázu dátami.



Pracujeme na lokálnom počítači - vytvoríme databázu s názvom zamestnanci a upravíme  pripravené súbory zamestnanec.txt, uvazok.txt, pobocka.txt.

Riešenie:

CREATE TABLE pobocka

(id SMALLINT(5) UNSIGNED NOT NULL AUTO\_INCREMENT PRIMARY KEY,

ulica VARCHAR(50) NOT NULL,

cislo VARCHAR(10) NOT NULL,

psc VARCHAR(5),

mesto VARCHAR(50)

);

INSERT INTO pobocka VALUES

(NULL, 'Gemerská', 1, '04010', 'Košice'),

(NULL, 'Bruselská' ,10, '04010', 'Košice'),

(NULL, 'Komenského',5, '08501', 'Bardejov'),

(NULL, 'Masaryková', 15, '07101', 'Michalovce');

CREATE TABLE uvazok

(id SMALLINT(5) UNSIGNED NOT NULL AUTO\_INCREMENT PRIMARY KEY,

typ VARCHAR(100) NOT NULL,

plat decimal(8,2) NOT NULL

);

INSERT INTO uvazok

VALUES

(null, 'plný', 1500),

(null, 'polovičný', 750),

(null, 'dvojtretinový', 500);

CREATE TABLE zamestnanec

(

id smallint(5) UNSIGNED NOT NULL AUTO\_INCREMENT PRIMARY KEY,

meno varchar(30) NOT NULL,

priezvisko varchar(50) NOT NULL,

id\_pobocka smallint(5) UNSIGNED DEFAULT 1,

id\_uvazok smallint(5) UNSIGNED NOT NULL DEFAULT 1,

email varchar(100) DEFAULT NULL,

mobil char(10) DEFAULT NULL,

nastup date NOT NULL

);

INSERT INTO zamestnanec (meno, priezvisko, id\_pobocka, id\_uvazok, email, mobil, nastup)

VALUES

( 'Petr', 'Sobotka', 1,1, 'petr.sobotka', '0905428432', '1980-02-03'),

( 'Robert', 'Mrkva', 2,1, 'robert.mrkva@firma.sk', '0904123456', '2002-02-03'),

( 'Igor', 'Repa', 2,2, 'igor.repa@firma.sk', '0904777777', '2003-02-03'),

( 'Bela', 'Starý', 2, 1,'bela.stary@firma.sk', '0904666666', '1999-02-03'),

( 'Marián', 'Zimný', 2, 1,'marian.zimny@firma.sk', '0904121212', '2000-02-03'),

( 'Lucia', 'Nová', 2, 1,'lucia.nova@firma.sk', '0904123123', '2001-02-03'),

( 'Andrej', 'Starý', 3, 1,'andrej.stary@firma.sk', '0904787878', '2002-02-03'),

( 'Boris', 'Hraško', 4, 1,'boris.hrasko@firma.sk', '0904585858', '2005-02-03'),

( 'Ferko', 'Mrkvička', 4,3, 'ferko.mrkvicka@firma.sk', NULL, '2020-08-18'),

( 'Miloš', 'Hnedý', 1, 1,'milos.hnedy@firma.sk', NULL, '2020-08-19'),

( 'Robert', 'Múdry', 2, 2,'robert.mudry@firma.sk', NULL, '2020-08-18'),

( 'Jozef', 'Púčik', 1, 3,'jozef.pucik@firma.sk', NULL, '2020-08-19'),

( 'Jozef', 'Vratko', 3, 3, 'jozef.vratko@firma.sk', NULL, '2020-08-19'),

( 'Jozef', 'Dobrák', 1, 3,'jozef.dobrak@firma.sk', NULL, '2020-08-19'),

( 'Ivan', 'Hrozny', 2, 3,'ivan.hrozny@firma.sk', NULL, '2020-08-19'),

( 'Albert', 'Einstein', NULL, 1,NULL, NULL, '2020-08-20');

Teória:

**Vytváranie tabuľky CREATE TABLE**

Zistili sme nasledujúce informácie o žiakoch a tie si chceme zapísať prehľadnejšie do tabuľky.

CREATE TABLE ziaci

(

id  SMALLINT(5),

meno  VARCHAR(10),

priezvisko  VARCHAR(15),

vek  SMALLINT(5),

ucebny\_odbor  VARCHAR(50),

rocnik  SMALLINT(3),

triedny\_ucitel  VARCHAR(40)

);

***Výsledok: hlavička tabuľky***



**Vloženie údajov INSERT INTO**

INSERT INTO ziaci

(id, meno, priezvisko, vek, ucebny\_odbor, rocnik, triedny\_ucitel)

VALUES

(1, 'Ivo', 'Tento', 28, '39176 M Technické a informatické služby', 2, 'Ján Múdry'),

(2, 'Emil', 'Ba', 28, '2569 M Staviteľstvo', 1, 'Edo Veselý'),

(3, 'Otto', 'Vítko', 28, NULL, NULL, NULL),

(4, 'Zuzana', 'Nová', 25, NULL, 1, 'Edo Veselý'),

(5, 'Jana', 'Červená', 26, '2561 M Informačné a sieťové technológie', 2, 'Ján Múdry'),

(6, 'Emil', 'Novák', 24, '5698 M Elektrotechnika', 1, 'Edo Veselý'),

(7, 'Eva', 'Čierna', 25, '2561 M Informačné a sieťové technológie', 3, 'Zuzana Múdra'),

(8, 'Šimon', 'Veľký', 23, '5698 M Elektrotechnika', 2, 'Ján Múdry'),

(9, 'Eva', 'Čierna', 25, '2561 M Informačné a sieťové technológie', 3, 'Zuzana Múdra');

***Výsledok: naplnenie tabuľky údajmi, ktoré máme k dispozícií***

Obrázok, na ktorom je text

Automaticky generovaný popis

Dátové ***typy:***

***Reťazcové dátové typy***

1. Najčastejšie používané

* **VARCHAR(x)** – slúži na ukladanie ľubovoľných reťazcov znakov (tzv. stringov) dĺžky najviac x
* **CHAR(x)** – sa tiež používa na ukladanie ľubovoľných reťazcov, tu však majú pevnú dĺžku x.

1. Obrázok, na ktorom je text

   Automaticky generovaný popisĎalšie

***Číselné dátové typy***

1. Najčastejšie používané

* **INT** (alebo neskrátené INTEGER) slúži na ukladanie celých čísel
* **DEC(x,y)** (alebo aj DECIMAL(x,y)) – desatinné čísla s celkovou dĺžkou (t.j. počtom cifier pred i za desatinnou čiarkou či bodkou)

1. Obrázok, na ktorom je stôl

   Automaticky generovaný popisĎalšie

***Dátové typy pre dátum a čas*Obrázok, na ktorom je stôl

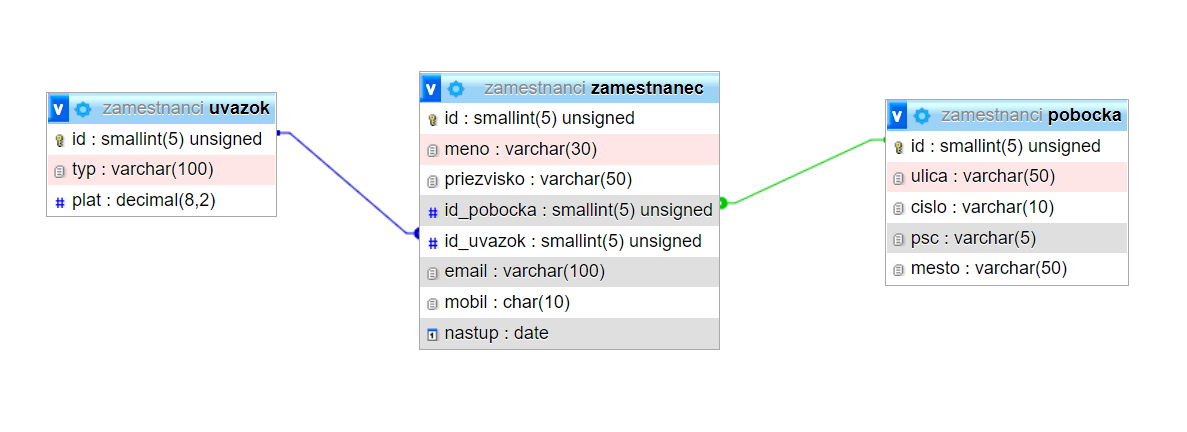
Automaticky generovaný popis**

**Základné pravidlá:**

* Názvy stĺpcov ani tabuliek spravidla neobsahujú medzery – v prípade potreby ich možno nahradiť podčiarkovníkmi (napr. datum\_narodenia)
* vyhýbame sa používaniu interpunkčných znamienok v názvoch tabuliek alebo stĺpcov
* veľkými písmenami budeme písať tie časti, ktoré sú súčasťou jazyka SQL (v našom prípade napr. CREATE TABLE alebo dátové typy VARCHAR, DATE, DEC, INT, atď.)
* malými písmenami naznačíme, že nejde o súčasť jazyka, ale o nami zvolené názvy (v našom prípade napr. názov tabuľky ziaci alebo názvy stĺpcov meno, priezvisko, vek, ucebny\_odbor, atď.)
* jazyk SQL nie je case-sensitive, to znamená že neberie ohľad na veľkosť písmen. V praxi to znamená, že výraz *select* má rovnaký význam ako *SELECT*
* neplatí to pre raťazcové hodnoty, ktoré uvádzame v úvodzovkách “ “

3.

Náš zákazník, pre ktorého spravujeme databázu zamestnancov si od nás vyžiadal výstupnú zostavu ohľadom jednotlivých pobočiek. Obsahovať má názov pobočiek, počet zamestnancov, priemernú mzdu, mesačné náklady a ročné náklady na jednotlivých pobočkách. Vytvorte zodpovedajúci dopyt.



Pracujeme s tabuľkami pobocka, zamestnanec, uvazok.

Riešenie:

SELECT p.ulica AS 'Názov pobočky',

COUNT(z.id) AS 'Počet zamestnancov',

AVG(u.plat) AS 'Priemerný plat',

SUM(u.plat) AS 'Mesačné náklady',

SUM(u.plat) \* 12 AS 'Ročné náklady'

FROM zamestnanec z

JOIN pobocka p ON z.id\_pobocka=p.id

JOIN uvazok u ON z.id\_uvazok = u.id

GROUP BY ulica;

Teória:

Agregácie – niekedy nás zaujímajú nielen vlastnosti jednotlivých záznamov, ale i vlastnosti nejakej ich množiny

* agregačné funkcie vracajú hodnotu, ktorá je vypočítaná z hodnôt v stĺpci
* hodnota NULL je ignorovaná
* pomocou agregačných funkcií je možné získať počet všetkých záznamov
* minimum/maximum z hodnôt
* priemernú hodnotu
* súčet

**COUNT** („počet“)

* vracia počet riadkov, ktoré vyhovujú daným kritériám
* použitie na: text, čísla a dátum

Napr.:

SELECT COUNT(\*) as pocet\_zamestnancov FROM zamestnanec;

**MAX** (skratka pre „maximum“) – nájde najväčšiu hodnotu

**MIN** (skratka pre „minimum“) – nájde najmenšiu hodnotu

* použitie na: text, čísla a dátum

Napr.: chceme zistiť dátum nástupu prvého zamestnanca

SELECT MIN(nastup) as datum\_nastupu\_prveho\_zamestnanca FROM zamestnanec;

**AVG** (skratka pre „average“)

* vracia priemernú hodnotu z hodnôt nachádzajúcich sa v číselnom stĺpci
* použitie na: čísla

Napr.: chceme získať priemerný plat zamestnancov

SELECT AVG(plat) AS priemerny\_plat

FROM zamestnanec z

JOIN uvazok u ON z.id\_uvazok = u.id;

Funkcia **CAST** slúži na konvertovanie z jedného dátového typu na iný.

Napr.: chceme získať výsledok v tvare desatinného čísla na dve desatinné miesta

SELECT CAST(AVG(plat) AS decimal(8,2)) AS priemerny\_plat

FROM zamestnanec z

JOIN uvazok u ON z.id\_uvazok = u.id;

**SUM**

* vráti celkový súčet hodnôt v stĺpci
* použitie na: čísla

Napr.: chceme získať celkovú sumu potrebnú na výplaty pre zamestnancov

select sum(plat) as mzdove\_naklady

from zamestnanec z

JOIN uvazok u ON z.id\_uvazok = u.id;

Zoskupovanie hodnôt

**GROUP BY** – zoskupiť podľa

* používa sa s agregačnými funkciami
* určuje zoskupovanie údajov vo výstupe podľa určených pravidiel

Napr.: Chceme vypísať mzdové náklady podľa typu úväzku

SELECT uvazok, SUM(plat)

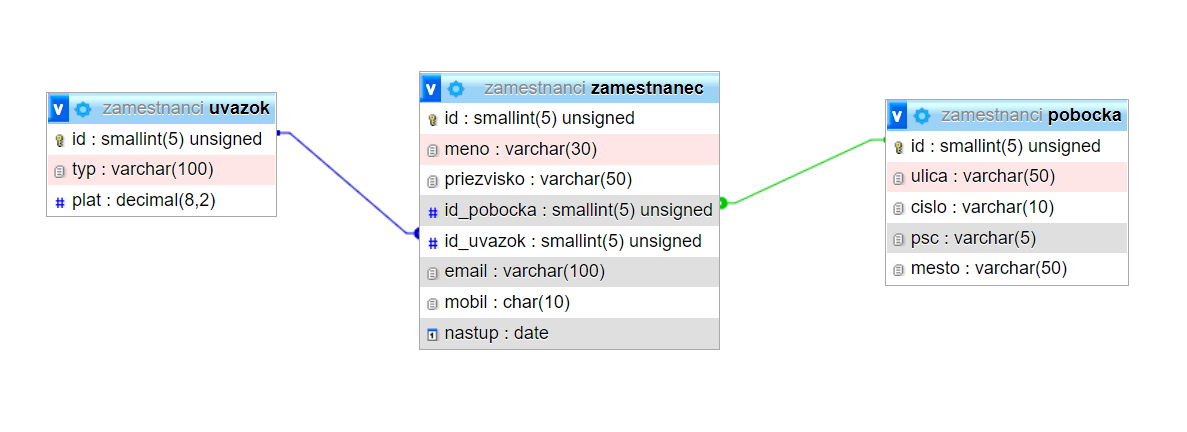
FROM zamestnanec z

JOIN uvazok u ON z.id\_uvazok = u.id

GROUP BY uvazok;

4.

Firma potrebuje zaevidovať do databázy zamestnancov dátum ukončenia pracovného pomeru u zamestnanca Jozefa Púčika k aktuálnemu dátumu. V databázovej štruktúre o zamestnancoch sa však nenachádza stĺpec ukončenie pracovného pomeru. Navrhnite kde a ako doplniť atribút „ukoncenie“, sledujte tabuľky zamestnanec, uvazok a pobocky,  popíšte postup v phpMyAdmin a zároveň aktualizujte údaj.



Pracujeme s tabuľkami pobocka, zamestnanec, uvazok.

Riešenie:

**Vloženie nového stĺpca do tabuľky**

ALTER TABLE *Názov\_tabuľky* ADD *Názov\_poľa* dátový typ;

Vložíme do tabuľky nový stĺpec ukoncenie

ALTER TABLE zamestnanec ADD ukoncenie DATE;

Samozrejme potrebujeme údaje doplniť, t.j.

Jozef Púčik ukončil pracovný pomer k aktuálnemu dátumu.

UPDATE zamestnanec

SET ukoncenie = CURRENT\_DATE

WHERE id = 13;

Teória:

**Úpravy tabuľky – DROP, ALTER TABLE, UPDATE**

**Vymazanie tabuľky**

DROP TABLE *Názov\_tabuľky*;

**Premenovanie tabuľky**

ALTER TABLE *Názov\_tabuľky* RENAME *Nový\_názov*;

**Premenovanie stĺpca tabuľky**

ALTER TABLE *Názov\_tabuľky* CHANGE *Pôvodný\_názov* *Nový\_názov* dátový typ;

**Vloženie nového stĺpca do tabuľky**

ALTER TABLE *Názov\_tabuľky* ADD *Názov\_poľa* dátový typ;

**Aktualizácia stĺpca tabuľky**

UPDATE *Názov\_tabuľky* SET stlpec1 = hodnota1, stlpec2 = hodnota2, ...

WHERE podmienka;

5.

Pracujte s tabuľkou „student“. Pre vedenie školy je potrebné vytvoriť výstupnú zostavu s nasledujúcimi údajmi: meno, priezvisko, dátum narodenia v tvare DD.MM.YYYY, pričom všetky údaje budú v jednom stĺpci.



Riešenie:

SELECT

CONCAT(meno, ' ', priezvisko, ' (', date\_format(datum\_narodenia, '%d.%m.%Y' ), ')') AS student

from student;

alebo

SELECT CONCAT(meno, ' ', priezvisko,' (',day(datum\_narodenia),'.', month(datum\_narodenia), '.', year(datum\_narodenia),')') AS student

FROM student;



**Teória:**

**Stĺpcové funkcie**

Databázový systém spravidla neslúži na formátovanie dát (o to sa zväčša starajú iné prostredia, ktoré z databázy len vytiahnu dáta). Ukážeme si však niekoľko príkladov:

**CONCAT** (z anglického „concatenate“ – spojiť - slengovo „lepidlo“) – z dvoch reťazcov vytvorí jeden tak, že druhý pripojí za prvý

SELECT CONCAT(meno, priezvisko)

FROM student;

Odpoveď:

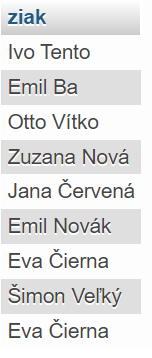


Medzi meno a priezvisko by ešte bolo treba vložiť jednu medzeru a nový stĺpec pomenovať aliasom:

SELECT CONCAT(meno, " ", priezvisko) AS ziak

FROM student;

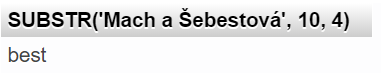
Odpoveď:



**SUBSTR** (skratka od „substring“ - podreťazec) – má tri parametre – reťazec *r*, z ktorého ideme podreťazec vyberať, poradie *z* začiatočného znaku podreťazca a dĺžku *d* podreťazca

select SUBSTR('Mach a Šebestová', 10, 4);

Odpoveď:



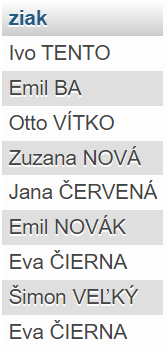
**UCASE** (UPPER – zo slovného spojenia „upper case“ – veľkými písmenami) – zmení všetky malé písmená na veľké

**LCASE** (LOWER – zo slovného spojenia „lower case“ – malými písmenami) – zmení všetky veľké písmená na malé

SELECT CONCAT(meno, " ", UCASE(priezvisko)) from student;

alebo

SELECT CONCAT(meno, " ", UPPER(priezvisko)) from student;



Keďže chceme aj informácie o žiakoch, ktorý priemer nemajú zatiaľ vyplnený, môžeme využiť funkciu **COALESCE** – obsahuje ľubovoľný počet argumentov a vráti prvú neprázdnu hodnotu. Ak sú všetky hodnoty prázdne, tak nemá na výber a vráti NULL.

SELECT

CONCAT(meno, ' ', priezvisko, ' (', COALESCE(priemer,'nevyplnený'), ')') AS student

from student;

Ak by sme chceli vypisovať priemer len na jedno desatinné miesto, môžeme použiť funkciu ROUND („zaokrúhli“), ktorá zaokrúhli číslo zo svojho prvého argumentu na počet miest v druhom argumente:

SELECT CONCAT(meno, ' ', priezvisko, ' (', ROUND(priemer,1), ')') AS student from student;

**Práca s dátumami** – Dátum máme uložený vo formáte **yyyy-mm-dd** a ten môžeme vo výpise meniť použitím funkcie

**date\_format(**'**date**'**,** '**format**'**)**, kde

date – je dátum, ktorý chceme formátovať

format – špecifický spôsob výpisu

Popis jednotlivých typov formátov:

%a - deň v týždni

%b - názov mesiaca

%c - číslo mesiaca v roku

%d -číslo dňa v mesiaci

%m - mesiac

%Y - rok

%y - posledné dva čísla roku

napr.

SELECT

CONCAT(meno, ' ', priezvisko, ' (',

COALESCE(date\_format(datum\_narodenia, '%d.%m.%Y'), 'Neznámy dátum narodenia'), ')'

)

from ziaci;