Výukový materiál zpracován v rámci projektu

EU peníze školám

Databáze

Petr Martiník

Registrační číslo projektu: CZ.1.07/1.5.00/21. 0418
Číslo klíčové aktivity: III/2
Název klíčové aktivity: Inovace a zkvalitnění výuky prostřednictvím ICT
Číslo materiálu: VY_32_INOVACE_281-300
Datum: 31. 10. 2013

Vzdělávací oblast:	Informatika a informační a komunikační technologie
Tematická oblast:	Databáze
Předmět:	Informatika a výpočetní technika
Výstižný popis způsobu využití, případně metodické pokyny:	Materiál obsahuje ve většině kapitol stručný popis učiva, které může sloužit jako podklad pro výklad učitele nebo pro samostatné studium či opakování žáků. Příklady a cvičení mohou být využity při výkladu, procvičování v hodinách nebo při domácí přípravě, nebo také k testování.
Klíčová slova:	Databáze, datové typy, relační databáze, SQL, Access, Excel
Druh učebního materiálu:	Pracovní sešit s doprovodnými soubory pro program Excel

Obsah

Databáze 4

1 Databáze a tabulky (VY_32_INOVACE_281) 5

2 Datové typy (VY_32_INOVACE_282) 8

3 Tabulky a datové typy – Access (VY_32_INOVACE_283) 10

4 E-R model (VY_32_INOVACE_284) 14

5 Relační databáze (VY_32_INOVACE_285) 18

6 Relační databáze – test (VY_32_INOVACE_286) 23

7 Relační databáze – Access (VY_32_INOVACE_287) 24

8 SQL – select (VY_32_INOVACE_288) 27

9 SQL – select (VY_32_INOVACE_289) 32

10 SQL – select – cvičení (VY_32_INOVACE_290) 35

11 SQL - spojování tabulek (VY_32_INOVACE_291) 36

12 SQL – cvičení (VY_32_INOVACE_292) 39

13 SQL – Access (VY_32_INOVACE_293) 40

14 SQL - skupiny, agregace (VY_32_INOVACE_294) 41

15 SQL - DML, DDL (VY_32_INOVACE_295) 43

16 Excel – třídění (VY_32_INOVACE_296) 45

17 Excel – filtrování (VY_32_INOVACE_297) 47

18 Excel – souhrny (VY_32_INOVACE_298) 49

19 Excel - kontingenční tabulky (VY_32_INOVACE_299) 50

20 Excel - souhrnné cvičení (VY_32_INOVACE_300) 51

Literatura 52

Databáze

Většina kapitol tohoto materiálu obsahuje stručný výkladový text a příklady, případně cvičení a náměty pro další studium. Mohou sloužit jako podklad pro výklad učitele a k opakování nebo samostatné přípravě žáků. Cvičení a náměty lze využít v hodinách, zadat jako domácí úkol nebo využít jako test. Materiál je určen pro výuku základů práce s databázemi. Omezuje se jen na stručný výběr základních pojmů a činností nezbytných pro pochopení struktury a operací v databázích.

Tématika databází se vyučuje v kvintě šestiletého gymnázia v povinném předmětu informatika a výpočetní technika. Předpokládá se základní znalost práce v Excelu a základní znalosti o datových typech v Pascalu, o relačních a logických operátorech a o zápisu podmínek.

Žáci by měli se měli seznámit se základními pojmy z oblasti relačních databází. Měli by navrhovat tabulky a efektivně volit datové typy, navrhnout strukturu jednoduché relační databáze, sestavit jednoduché dotazy v jazyce SQL, seznámit se s prostředním programu Access a realizovat v něm základní operace nutné pro definici databáze a práci s dotazy, provádět databázové operace v Excelu.

K závěrečným kapitolám jsou přiloženy doprovodné soubory (xlsx). Soubory a popis praktických činností vychází z programů Microsoft Excel a Microsoft Access ve verzi 2013.

Každé téma je určeno na jednu výukovou hodinu. Rozsah a časová náročnost cvičení a úkolů pro přípravu žáků je závislý na výběru učitele.

1 Databáze a tabulky (VY_32_INOVACE_281)

S pojmem databáze (databanka, báze dat,...) jste se již jistě setkali. Určitě byste dokázali vyjmenovat mnoho příkladů a oblastí, kde se s nimi můžete setkat: evidence knih a čtenářů v knihovně, evidence studentů a jejich studia ve škole, evidence cédéček v domácí diskotéce, bankovní operace na účtu, zaměstnanci a výrobky ve firmě, nákupy a prodeje v obchodě, skladová evidence, sportovní soutěže a mnoho dalších.

I vy se jistě často dostáváte do situace, kdy potřebujete něco přehledně evidovat. Pokud byste například měli několik knih ve své domácí knihovničce, sepsali byste si je asi na papír. Pokud je vaše knihovnička větší, asi byste uvažovali o počítačové evidenci. Pro menší počet knih byste zapsali seznam do textového editoru (třeba do Wordu). O každé knize byste si chtěli zapsat více informací (autor, název, cena, počet stran, rok vydání, zda ji má někdo půjčenou, ...) a bylo by dobré mít informace zachycené ve formě tabulky. Při větším počtu knih by už práce v textovém editoru nebyla přehledná a pohodlná. S tabulkami se lépe pracuje v tabulkových kalkulátorech, které kromě matematických výpočtů mají také podporu pro práci s databázovými údaji. Pokud bychom evidovali knihy v knihovně, chtěli bychom asi také evidovat čtenáře, zaměstnance knihovny a možná i další informace. Měli bychom už tabulek více. Práce v tabulkovém kalkulátoru by už měla jistá omezení a mohli bychom narazit i na problém s maximálním počtem řádků tabulky. Potom už je lépe použít specializované databázové programy.

Pod pojmem databáze si tedy můžeme představit systém strukturovaných informací, které jsou zachycené v tabulkách a tyto jsou propojené nějakými vztahy. Může být uložena na papíru (sešity, kartotéky) nebo na počítačových paměťových mediích.

V širším slova jsou součástí databáze i softwarové prostředky, které umožňují definovat strukturu informací, manipulovat s daty a přistupovat k nim. Takové programy se nazývají SŘBD – systémy řízení báze dat (DBMS – Database Manegement System). Pod pojmem databáze budeme rozumět jak vlastní data, tak tento software.

Databázové programy mají velký historický význam. Na prvních počítačích v padesátých letech minulého století se kromě vědeckotechnických výpočtů realizovaly právě databázové úlohy. Této oblasti využití se říkalo hromadné zpracování dat a pro tyto účely byl vyvinut programovací jazyk COBOL. První relační databázový systém byl v 80. letech dBase, dále vznikaly různé systémy odvozené od tohoto systému nebo i systémy nové (FoxPro, Clipper, Paradox). Formáty uložení dat v těchto systémech se používají dodnes. Kancelářské balíky Microsoft Office obsahují v rozšířených verzích databázi Access, podobně v OpenOffice je nástroj Base. Tyto databáze se

hodí spíše k domácímu nebo kancelářskému použití. Tam, kde se zpracovává velké množství dat a jsou vysoké požadavky na jejich zabezpečení, se používají systémy Oracle, Informix, Progress a další.

Pokud bychom hledali přesnější definici pojmu databáze a dalších pojmů zde uvedených, zjistili bychom, že databázová terminologie není příliš jednotná a také to, že některé pojmy jsou definovány dost složitě. Databázové systémy jsou založeny na přísných matematických pravidlech. Jejich studiem se možná někteří z vás budou zabývat. I v databázovém programu Access narazíme na velké množství informací a možností. Zde se budeme zabývat pouze výběrem základních zjednodušených principů nutných pro rozumný návrh struktury a práci s databázovými informacemi.

S databázemi se běžně setkáváte při práci na počítači, většinou si to ani neuvědomujete. V konkrétních databázových programech je už struktura databáze nadefinována a je připraveno prostředí tak, aby práce byla pro uživatele jednoduchá a přehledná. My se nebudeme zabývat databázemi z pohledu uživatele, ale půjde nám o návrh struktury z pohledu tvůrce databázové aplikace. Pokud bychom chtěli databázový program kompletně připravit pro běžného uživatele, museli bychom se ještě zabývat tvorbou formulářů a sestav, případně programováním. Informace musí být také zabezpečeny tak, aby k nim někdo neměl neoprávněný přístup (systém uživatelů a přístupových práv), ale také aby nedošlo ke ztrátě nebo narušení konzistence dat.

Databázi můžeme tedy chápat jako více vzájemně propojených tabulek. Základním pojmem tedy je tabulka.

Adresář

Jméno	Příjmení	Adresa	Telefonní číslo	Datum narození
František	Novák	Hlučín, Dr. Ed. Beneše 7	606254789	2.3.1976
Barbora	Veselá	Praha, Milady Horáková 5	777256789	5.7.1991
Tomáš	Koutný	Ostrava, Nádražní 20	708214529	5.12.1963

Knihovna

Autor	Název	Rok vydání	Cena	Počet stran	Ilustrace	Půjčeno
Verne	Děti kapitána Granta	1953	120	453	ano	
Němcová	Babička	1978	200	487	ne	Andrea
Malchárková	Modrá barva duhy	2011	220	136	ne	
Balabán	Možná, že odcházíme	2012	285	178		Martin

Ve sloupcích tabulky jsou atributy (vlastnosti, položky, pole,...) – jaké informace si chceme o každém objektu (člověk, kniha,...) pamatovat. Řádky tvoří záznamy (record, databázová věta,...).

Obsahují vždy informace o jednom objektu. Záznamy zadávají a editují uživatelé programu. Atributy musí být nadefinovány při tvorbě databáze.

Při definici databáze založíme databázi, v ní vytvoříme jednotlivé tabulky a v každé tabulce nadefinujeme atributy. Každý atribut musí mít určen identifikátor (název v záhlaví), datový typ (typ informace), můžeme přidat omezení hodnot a další vlastnosti.

Možná si vzpomínáte z programování v Pascalu, že identifikátory proměnných nesměly obsahovat mezery, háčky a čárky nad písmeny. V databázi Access nejsme příliš omezeni a můžeme volit názvy identifikátorů podobně jako ve výše uvedených tabulkách. Pro rozumnou práci je dobré volit identifikátory výstižné, krátké, bez mezer, používat pouze písmena anglické abecedy a číslice, případně podtržítko. Příprava databázového programu tak bude rychlejší a snížíme pravděpodobnost překlepů a dalších možných chyb. Vždy se dá zařídit, aby uživatel viděl názvy vlastností ve srozumitelnější podobě. V jedné tabulce nemohou být dva atributy se stejnými identifikátory.

2 Datové typy (VY_32_INOVACE_282)

Pro každý atribut musíme definovat datový typ. Tak jako v programovacích jazycích je důvodem pro rozlišení datových typů časová a paměťová efektivita. Pro každý typ jsou navíc definovány speciální operace a funkce. Datové typy v databázích se v mnohém podobají datovým typům v jazyce Pascal (integer, real, boolean, char, string) nebo v jiných programovacích jazycích. Základním rozdílem je fakt, že databáze nemají strukturované datové typy, tzn. atributy se nemohou skládat z jednotlivých prvků (složek). Existují sice databázové systémy, které toto umožňují, ale práce se strukturovanými typy není v databázích příliš komfortní. Databáze jsou rozšířeny o datové typy, které se zde často používají (datum nebo čas, měna). Jednotlivé datové typy mohou mít různé názvy a vlastnosti v různých databázových systémech. Většinou jsou ale k dispozici následující typy.

Číselné typy se dělí na celočíselné a reálné. Zatímco s celočíselnými typy jsou výpočty přesné, u reálných typů dochází k zaokrouhlovacím chybám. Připomeňme si, že celočíselným typem v jazyce Pascal je integer, reálným typem typ real. Možná už také víte, že může být i více celočíselných typů (liší se paměťovými nároky a rozsahem hodnot) a reálných typů (liší se také paměťovými nároky a rozsahem hodnot a navíc přesností, tj. počtem platných číslic). Pro celá a reálná čísla mohou být v databázích zavedeny samostatné typy nebo je k dispozici číselný typ a v nastavení jeho vlastnosti se určí, zda jde o celočíselné nebo reálné hodnoty. Jsou data, kterým běžně říkáme číslo, ale ve skutečnosti to číslo není a místo číselného typu musíme definovat textový typ. Jsou to například rodné číslo, telefonní číslo, číslo občanského průkazu a podobně. Za číslo považujme vždy jen takové informace, se kterými má smysl provádět matematické operace (například je sečíst nebo vypočítat aritmetický průměr).

Také textové typy jsou dva. Řetězce (v Pascalu string) jsou posloupnosti libovolných znaků. Používají se pro jednořádkové texty s maximální délkou většinou 255 znaků. Hodnota atributu v každém záznamu potom zabírá 256 bajtů. V mnoha případech nám ale postačí kratší řetězce (například jméno určitě nebude delší než 15 znaků, označení třídy ve škole nebude delší než 5 znaků, ale naopak atribut pro název knihy nelze rozumně zkrátit). U řetězců bychom vždy měli přemýšlet o maximální možné délce a nastavit ji ve vlastnostech tak, abychom šetřili pamětí. Druhým textovým typem je typ označovaný memo (poznámka). Tyto texty mohou být i delší než 255 znaků (v databázi Access až 64 000 znaků) a mohou být rozčleněny na odstavce.

Z Pascalu znáte ještě datový typ boolean, jehož ekvivalenty jsou i v databázích. Připomeňme si, že se jedná o logický datový typ a atributy tohoto typu mohou nabývat pravdivostních hodnot pravda (true) nebo nepravda (false).

Speciální databázové datové typy jsou měna a datum (případně čas).

Zajímavým datovým typem je typ počítadlo (counter). Hodnoty tohoto typu jsou kladná celá čísla, která generuje databázový systém při založení záznamu. Uživatel je nemůže zadat ani změnit. Navíc jsou generovány tak, aby byly unikátní v rámci jedné tabulky, tzn. aby hodnota v každém záznamu byla jiná. Využívají se jako klíče, které budou popsány v dalších kapitolách.

V dnešních databázích lze používat další typy informací (hypertextové odkazy, obrázky, zvuky, videa,...), pro které jsou definovány speciální datové typy. Těmi se zde nebudeme zabývat.

Navrhněte strukturu tabulky pro učitele, ve které si bude evidovat informaci o studentech, které učí. Pokuste se vymyslet atributy tak, aby tabulka obsahovala všechny výše uvedené datové typy.

Tabulka by mohla mít strukturu:

jmeno	řetězec, délka 15	křestní jméno studenta
prijmeni	řetězec, délka 20	příjmení studenta jména vždy rozdělujte na dva atributy jméno, příjmení
narozen	datum	datum narození
predmet	řetězec, délka 3	vyučovaný předmět - zkratka
absence	celé číslo	počet zameškaných hodin
neomluveno	celé číslo	počet neomluvených hodin
ukoly	logická hodnota	zda student splnil všechny úkoly
klasifikace	celé číslo	známka z předmětu
prumer	reálné číslo	průměr na posledním vysvědčení
zaloha	měna	záloha vybraná na nákup pomůcek a literatury
vydaje	měna	skutečná cena nákupu
prace	memo	poznámky o práci v hodinách (aktivita, kázeň,...)

Tabulka by samozřejmě mohla obsahovat mnoho dalších atributů. Pokud by účelem evidence žáků bylo něco jiného, než klasifikace (např. školní výlet, stravování,...), byly by atributy jiné.

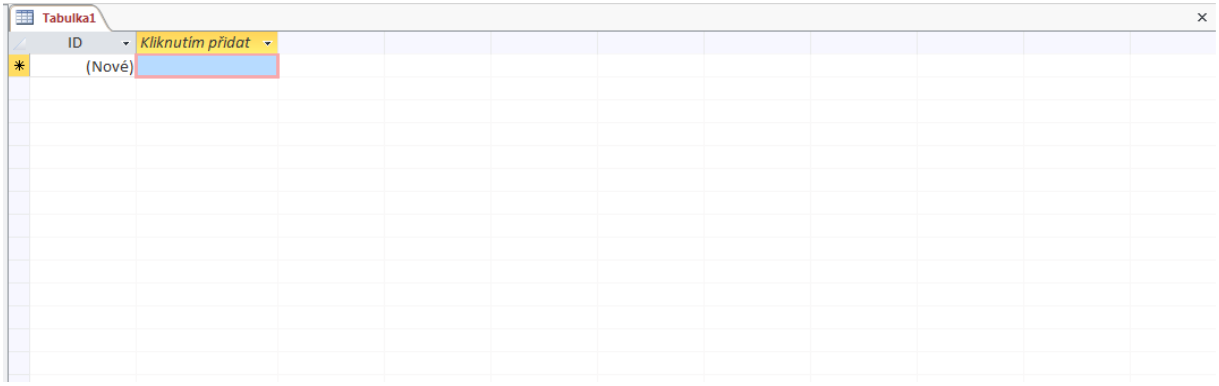
Zvolte si téma (a účel) tabulky a navrhněte datové typy podle předchozího příkladu. Náměty: knihy, seznam přátel, zaměstnanci firmy, pacienti u lékaře, inventární seznamy, evidence úkolů, atd.).

3 Tabulky a datové typy – Access (VY_32_INOVACE_283)

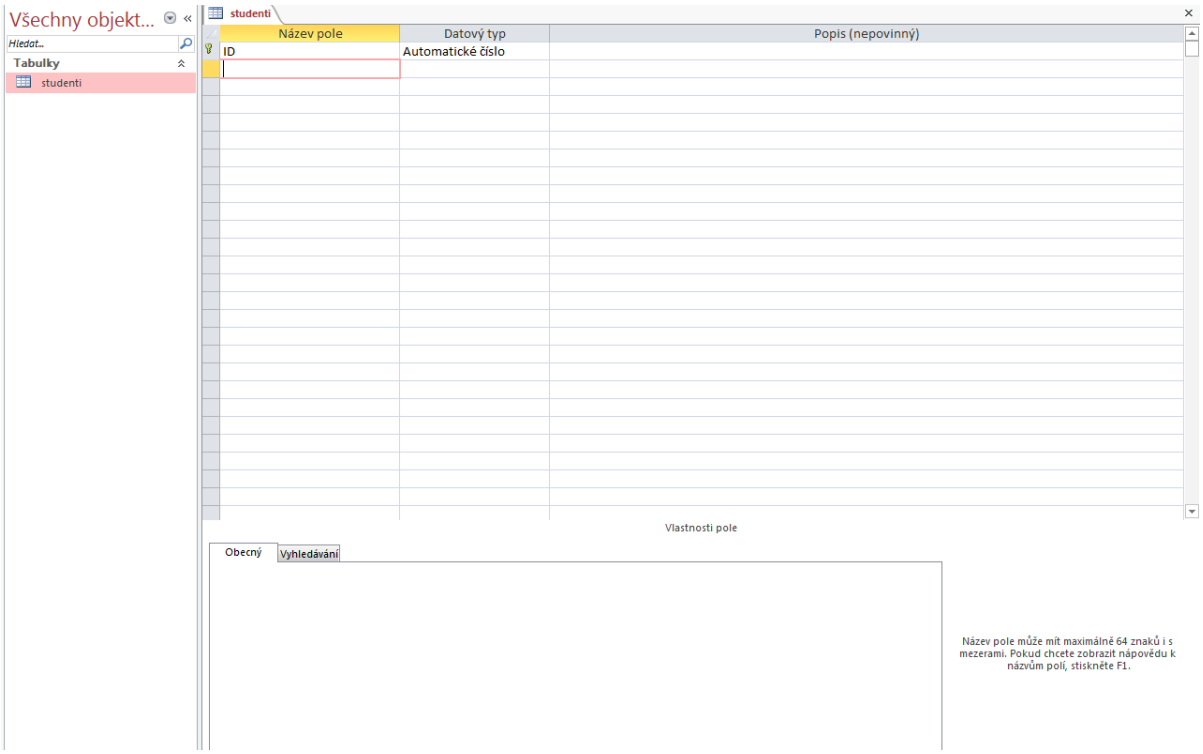
V této kapitole si ukážeme práci s databází a jednou tabulkou v prostředí programu Microsoft Access. Použijeme k tomu tabulku studentů z předchozí kapitoly.

Po spuštění tohoto systému vidíme hned jeden rozdíl oproti jiným programům, se kterými jsme zatím pracovali. Pokud jsme spustili program Word, mohli jsme psát a formátovat text (nazývá se pracovně Dokument) aniž by byl uložen v nějakém konkrétním souboru. Podobně se pracuje v Excelu (Sešit), v Power Pointu (Prezentace), v Malování (Bez názvu), v Pascalu (Untitled) a v jiných programech. V Accessu musíme hned v úvodu vytvořit nový soubor na disku. V nabídce bychom mohli vybrat některou z šablon, ve kterých jsou připravené struktury často používaných databází (Kontakty, Správa úkolů, Sledování majetku,...). V databázích založených na těchto šablonách bychom mohli měnit strukturu a pracovat s informacemi jako uživatelé. Mohli bychom samozřejmě otevřít také existující databázové soubory. My ale chceme vytvořit zcela novou databázi s vlastní strukturou, takže zvolíme možnost Prázdná databáze. V dalším okně zvolte umístění a název databáze (zvolme název vyuka) a tlačítkem Vytvořit se založí soubor. Jedná se o soubor s příponou accdb, do kterého se budou ukládat všechna nastavení i data.

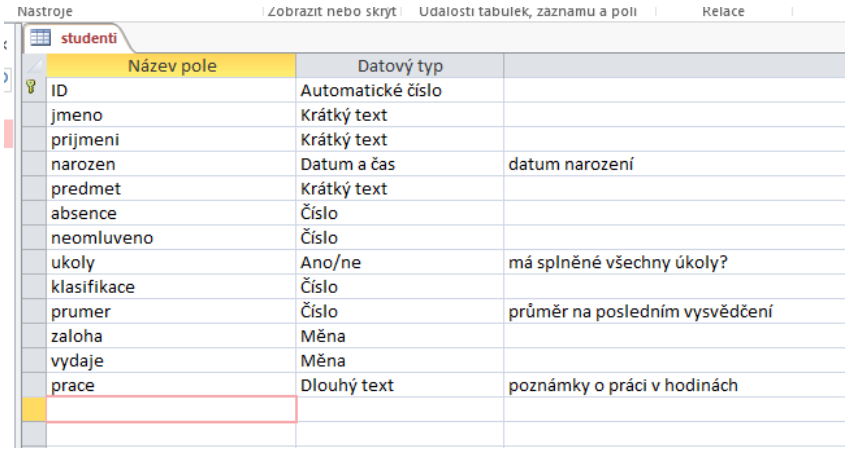
V každé databázi musí být definovaná alespoň jedna tabulka, takže nás program ihned přesunul do definice nové tabulky.



Tabulku lze zobrazit v různých režimech zobrazení, které můžeme přepínat tlačítkem Zobrazení vlevo nahoře. Nacházíme se v zobrazení Datový list, přepneme se do režimu Návrhové zobrazení. Systém po nás vyžaduje Název tabulky, zadejme název studenti.



V Návrhovém zobrazení budeme ve sloupci Název pole zapisovat pod sebe identifikátory jednotlivých atributů (atributy se v Accessu nazývají pole), ve sloupci Datový typ budeme vybírat typ atributu, v popisu si můžeme napsat libovolnou poznámku. V dolní části Vlastnosti pole doplníme v některých případech další nastavení. První řádek s názvem pole ID zatím ponechte a zadejte strukturu tabulky.



Název pole	Datový typ	
ID	Automatické číslo	
jmeno	Krátký text	
prijmeni	Krátký text	
narozen	Datum a čas	datum narození
predmet	Krátký text	
absence	Číslo	
neomluveno	Číslo	
ukoly	Ano/ne	má splněné všechny úkoly?
klasifikace	Číslo	
prumer	Číslo	průměr na posledním vysvědčení
zaloha	Měna	
vydaje	Měna	
prace	Dlouhý text	poznámky o práci v hodinách

Datové typy v Accessu mají následující názvy:

- Číslo – celá i reálná čísla,
- Krátký text – řetězce,
- Dlouhý text – memo (poznámky),
- Ano/ne – logický datový typ, hodnoty true a false,

- Datum a čas,
- Měna,
- Automatické číslo – počítadlo.

U některých polí nastavíme jejich vlastnosti. U polí typu Krátký text nastavíme Velikost pole. Maximální velikost pole je 255 znaků. Abychom šetřili paměť, měli bychom hodnotu rozumně zmenšit (jmeno 15, prijmeni 20, predmet 3). Výběrem velikosti pole u číselných typů určíme, zda jde o celočíselný nebo reálný typ. Celočíselné jsou typy bajt, celé číslo a dlouhé celé číslo, reálné mají názvy jednoduchá přesnost a dvojitá přesnost. Tyto číselné typy se liší paměťovými nároky a možným rozsahem hodnot, reálné typy navíc přesností (počtem platných číslic). Pro klasifikaci nám postačí typ bajt (zabírá v paměti 1 bajt, možné hodnoty jsou od 0 do 255). Doufejme, že tento typ můžeme použít i pro pole absence a neomluveno. Pokud bychom potřebovali i záporná čísla, zvolíme typ celé číslo a v případě, že by nám jeho rozsah nestačil, dáme dlouhé celé číslo. U reálných typů (v našem případě prumer) nám téměř vždy bude stačit jednoduchá přesnost. Dalších vlastností a nastavení si zde nebudeme všímat. Změny, které nastavujeme v návrhovém zobrazení, musíme ukládat průběžně (Uložit – CTRL+S) nebo se nás na uložení zeptá systém při zavření tabulky.

Přehled číselných typů

typ	paměť	rozsah	desetinná místa
Bajt	1 bajt	0 – 255	
Celočíselný	2 bajty	-32 768 - 32 767	
Dlouhý celočíselný	4 bajty	-2 147 483 648 – 2 147 483 647	
Jednoduchá přesnost	4 bajty	-3,402823E38 – 3,402823E38	7
Dvojitá přesnost	8 bajtů	-1,79769313486232E308 - 1,79769313486232E308	15

Kdykoli můžeme tabulku přepnout do zobrazení datového listu. Zde pracujeme jako uživatelé a můžeme editovat data. V tabulce je vždy připraven jeden prázdný řádek pro založení nového záznamu. Zadejte alespoň 5 studentů. Nemusíte nutně vyplňovat všechny informace, určitě ale bude důležité zadat jméno a příjmení. Při editaci záznamů v roli uživatele se informace ukládají do souboru automaticky. Všímejte si chování pole ID (Automatické číslo). Zkoušejte reakci programu na nesmyslné hodnoty (záporná klasifikace, nesmyslné datum, delší text než jsme zadali v definici struktury tabulky, text místo čísla,...). Logické hodnoty se zadávají pomocí zaškrtávacího políčka. Problém je pouze se zadáváním hodnot typu memo (prace), na které nemáme dostatek prostoru. I ostatní pole by mohla být uspořádána přehledněji. K tomu se dají

v databázi nadefinovat tzv. formuláře. Pokud chceme záznam odstranit, označíme příslušný řádek a zvolíme odstranit. Můžeme označit a odstranit i více záznamů najednou.

V tabulce se kdykoli můžeme přepnout (i tehdy, když už obsahuje data, nebo po jejím zavření a opětovném otevření) do návrhového zobrazení a změnit její strukturu. Můžeme přidat nové pole na konec seznamu. Jednotlivá pole by měla být seřazena v rozumném pořadí. Například nebylo by dobré pořadí jmeno, absence, klasifikace, zaloha, prijmeni, neomluveno, vydaje,.... Nová pole můžeme vkládat nejen nakonec. Pomocí položky Vložit řádky v nabídce se vytvoří prázdný řádek a zde můžeme nadefinovat nové pole. Označením více řádku lze takto vložit i více prázdných řádků za sebou. Pořadí polí lze měnit označením šedého čtverečku na začátku řádku a tažením ho přemístit na jiné místo. Takto můžeme označit a přesunout i více řádků.

Označením a volbou Odstranit řádky můžeme pole nebo více polí z tabulky odstranit. Samozřejmě v tomto případě dojde ke ztrátě příslušných dat. Můžeme také změnit název pole. V naší tabulce se nic podstatného nestane, ale pokud bychom měli rozsáhlejší aplikaci, ve které jsou nadefinované objekty, které spolupracují s danou tabulkou, mohlo by dojít k problémům. V tabulce lze také měnit vlastnosti polí. Zvětšením velikosti pole u typů Krátký text se data nezmění. Pokud ho ale zmenšíme, mohou být existující informace zkráceny. Podobné problémy mohou být i u změny velikosti pole číselných typů nebo u změny datového typu.

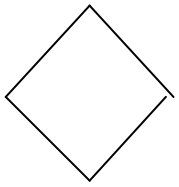
Struktura tabulek se dá kdykoli změnit. Abychom se vyhnuli možným problémům, je důležité mít strukturu tabulek dobře rozmyšlenou předem.

4 E-R model (VY_32_INOVACE_284)

Databáze není jen jedna tabulka, ale více tabulek propojených nějakými vztahy. Při návrhu struktury databáze si musíme nejprve ujasnit o čem budeme chtít evidovat informace, jaké informace chceme zapisovat a jaké jsou mezi nimi vztahy. Pro popis úseku reality, který nás zajímá, se používá grafické schéma, které se nazývá E-R model (entitně-relační model), který ještě není určen pro počítačové zpracování. V další kapitole si ho převedeme do počítačové databáze. Základními pojmy jsou entita (o čem evidujeme informace), atribut (jaké informace o každém prvku entity) a relace (vztahy mezi entitami). Popisují se pomocí grafických značek propojených spojnicemi:

entita

atribut



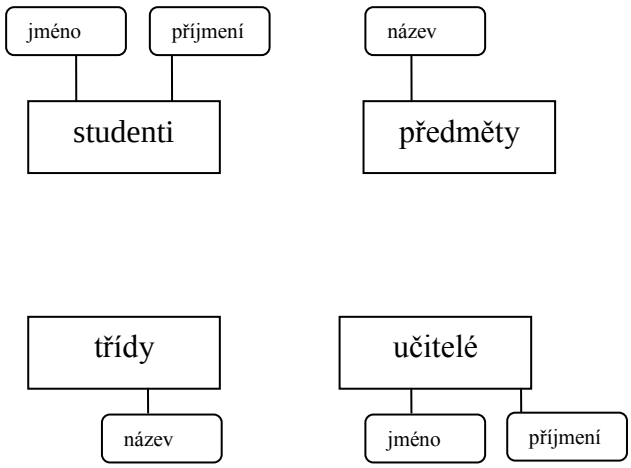
relace

Sestavte E-R model pro evidenci výuky ve škole.

Úkol máme definován velice volně. Budeme ho dále zpřesňovat a vysvětlíme si na něm základní pojmy. Model by mohl být mnohem složitější.

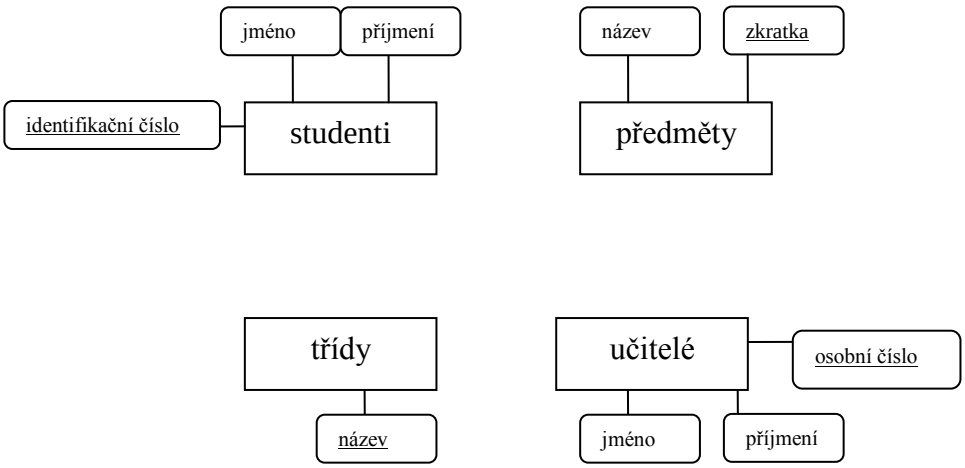
Entita jsou skupiny objektů stejných vlastností, které chceme evidovat. V našem příkladu si určitě budeme chtít vést informace o studentech, třídách do kterých chodí, učitelích, kteří je učí a vyučovacích předmětech. Budeme mí tedy čtyři entity, které nazveme studenti, třídy, učitelé a předměty. Graficky zachytíme takto.

Dále přidáme atributy. Atribut je vlastnost entity. Jsou to informace, které chceme evidovat o každém prvku entity. Jak už jsme viděli v předchozích příkladech, může jich být velké množství. Tady budeme zapisovat jen ty nejdůležitější, který by měly být určitě zadány. Ostatní doplníme až při detailnějším zadávání struktury do tabulky.

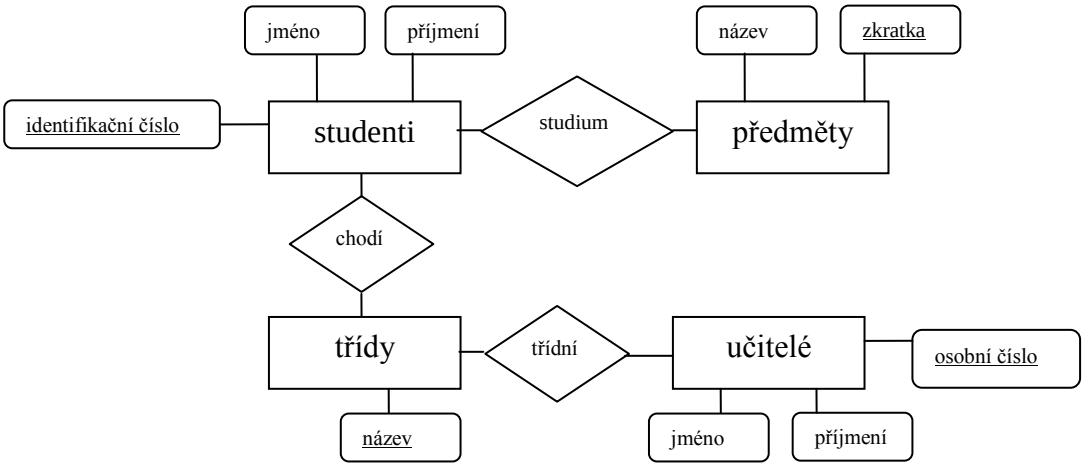


Důležitým atributem je klíč. Klíč je atribut, který jednoznačně identifikuje objekt v rámci jedné entity. Klíč může být tvořen i skupinou více atributů, my se vždy budeme snažit o jednoatributový klíč. Jeho hodnota musí být v rámci entity unikátní, což znamená, že žádné dva prvky entity nemohou mít jeho hodnotu stejnou. Nemůžeme tedy vzít jako klíč příjmení v entitě studenti, protože mohou být dva nebo více studentů se stejným příjmením. Nepomohla by nám ani kombinace příjmení a jména. Musíme zvolit nějakou informaci, která je u každého studenta jiná. Tím je například rodné číslo. Pro zjednodušení další práce zvolme raději identifikační číslo studenta, které mu bude přiděleno při nástupu na školu. Podobně zavedeme jako klíč osobní čísla u učitelů. Každá třída se určitě jmenuje jinak, takže můžeme vzít název přímo jako klíč. Podobně bychom to mohli udělat u entity předměty, každý předmět se určitě jmenuje jinak. Tady ale zvolíme jiné řešení. Názvy předmětů jsou dlouhé a mají velké paměťové nároky. Jak uvidíme v další kapitole, hodnoty klíčů se na rozdíl od ostatních atributů zapisují do databází vícekrát. Měli bychom se tedy snažit o to, aby byly co nejkratší. Zaberou potom méně paměti a také jejich psaní je rychlejší a pravděpodobností překlepu je nižší. Pro předměty proto zavedeme jako klíč atribut zkratka, do kterého budeme zapisovat zkratky předmětů. U entity třídy můžeme název jako klíč ponechat, názvy tříd nejsou příliš dlouhé. Názvy klíčů budeme podtrhávat.

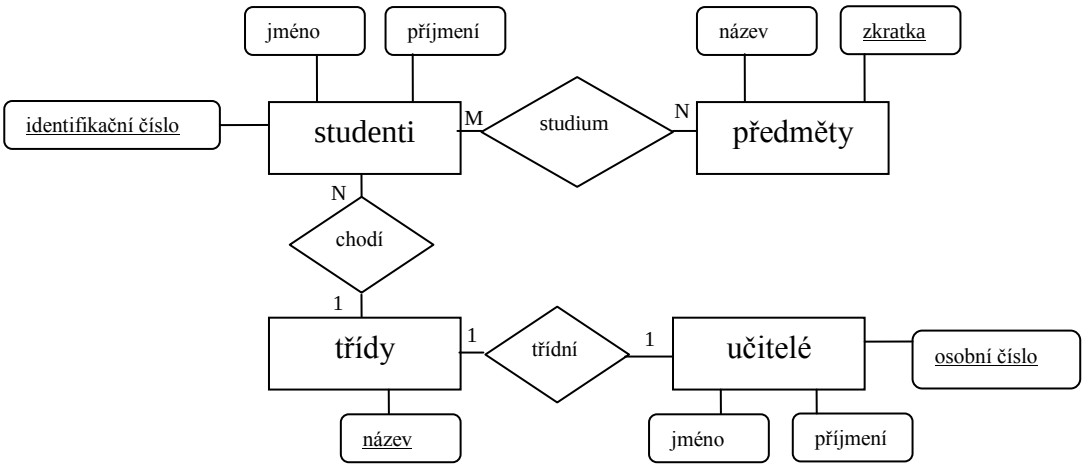
Klíče budeme definovat u každé tabulky. Musíme vybrat takový atribut, jehož hodnoty jsou v každém řádku jiné. Snažíme se o co nejkratší klíče. Pokud nenajdeme žádný vhodný atribut, můžeme zavést atribut typu počítadlo, kde se o jednoznačné celočíselné hodnoty postará databázový systém.



Posledním pojmem je relace, což je vztah mezi entitami. Relace v databázích jsou vždy binární, dávají do vztahu dvě entity. Do našeho modelu přidáme relace, které nazveme třídní (kdo je ve které třídě učitelem), chodí (kdo chodí do které třídy) a studium (jaké předměty studují jednotliví studenti).



Musíme ještě určit typ relace. Typ může být 1:1, 1:N nebo M:N. Relace třídní je typu 1:1, protože každá třída má pouze jednoho třídního a učitel může být třídním pouze v jedné třídě. Relace chodí je typu 1:N, protože student chodí do jedné třídy, ale do třídy chodí více studentů. Poslední relace studium je typu M:N, protože každý student studuje více předmětů a jeden předmět studuje více studentů. Označení N a M chápejme ve smyslu „více“, nejedná se o nějaký konkrétní počet.



Relací může být mnohem více. Dvě různé entity může dokonce propojovat více relací (např. kromě třídního učitele může být i jeho zástupce). Mohli bychom zavést novou relaci mezi studenty a učiteli, ve které bychom chtěli zachytit jakého třídního učitele má každý student. Tato relace by ale byla nadbytečná (redundantní), tento vztah už je zachycen prostřednictvím entity třídy. Entita může být i ve vztahu sama se sebou (např. někteří začínající učitelé mohou mít svého uvádějícího učitele). Určitě se nám podaří vymyslet vztahy, které spojují i více než dvě entity. Například klasifikace je vztah mezi studentem, učitelem, předmětem a další entitou známky. Takové relace by ale nešly převést do počítačového databázového systému. Proto je musíme převést na binární relace. To lze udělat jednoduchým způsobem tak, že z relace uděláme entitu a propojíme ji relacemi 1:N s příslušnými entitami.

Při návrhu modelu je nezbytné se seznámit se skutečnou situací. Zejména je nutné si ujasnit typy relací. Například relaci třídní jsme navrhli 1:1. Kdyby ale učitel mohl být třídním ve více třídách, museli bychom relaci upravit na typ 1:N. Pokud bychom předpokládali, že může být v jedné třídě více třídních učitelů, šlo by o relaci M:N.

Vyberte si oblast a účel evidence a navrhněte E-R model. Použijte alespoň 3 entity a relace všech typů.

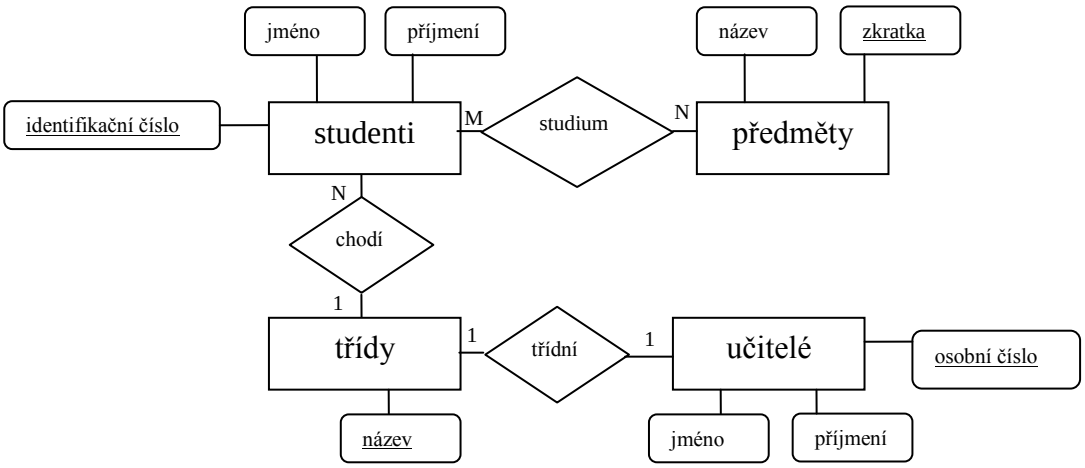
5 Relační databáze (VY_32_INOVACE_285)

E-R model popisuje strukturu reality bez vazby na počítačové zpracování. Entity, atributy i relace je ale potřeba převést do počítače. K tomuto účelu se používaly různé modely (hierarchický, síťový), které ale vykazovaly nedostatky. Rozdíl mezi jednotlivými modely je v zachycení relací. Dnes se nejčastěji používá relační model, jehož výhodou je, že se v něm snadno modifikuje struktura databáze, data lze kdykoli uvést do jakýchkoli vztahů. Nověji se používají modely objektové.

Relační model je charakteristický tím, že:

- vše je zachyceno v tabulkách (entity i relace),
- relace se realizují pomocí hodnot klíčů.

Převod E-R modelu do tabulek relační databáze si ukážeme na příkladu z minulé kapitoly. Názvy atributů při převodu upravíme. V tomto příkladu budeme pro pochopení zapisovat v každé tabulce i několik záznamů.



Entity převedeme jednoduše. Každé entitě odpovídá jedna tabulka. Uvádíme pouze klíč a základní atributy, další bychom jednoduše doplnili.

studenti

cislo	jmeno	prijmeni
123	Anna	Nováková
203	Jan	Dvořák
126	Petr	Černohorský
242	Marie	Komárková

tridy

nazev
1.A
1.B
2.A
2.B

ucitele

oscislo	Jmeno	prijmeni
39	Milan	Solich
12	Petr	Martiník
5	Věra	Smutná
14	Anděla	Nebeská
25	Vladimír	Koutný

predmety

zkratka	nazev
M	matematika
F	fyzika
IVT	informatika a výpočetní technika

Relace převedeme podle toho, jakého jsou typu.

Relace 1:1 je v našem příkladu relace třídní. V tomto případě můžeme do tabulky tridy přidat atribut pro zaznamenání třídního učitele a budeme do něj zapisovat hodnotu klíče příslušeného učitele.

tridy

nazev	tridni
1.A	39
1.B	5
2.A	25
2.B	14

Třídním v 1.A je tedy Milan Solich, protože hodnota jeho klíče v tabulce ucitele je 39 a podobně to platí pro další třídy a učitele.

Tuto relaci bychom mohli také zachytit v tabulce ucitele přidáním atributu trida, do kterého budeme zapisovat klíč (tedy název) třídy.

ucitele

oscislo	jmeno	prijmeni	trida
39	Milan	Solich	1.A
12	Petr	Martiník	
5	Věra	Smutná	1.B
14	Anděla	Nebeská	2.B
25	Vladimír	Koutný	2.A

Z obou možností vybereme pouze jednu. Výhodnější je v tomto případě první možnost. U každé třídy bude totiž třídní učitel určitě určen, takže hodnota pole tridni bude vyplněna. Ve druhé možnosti bude v atributu trida řada prázdných polí, protože ne každý učitel je třídním učitelem

v nějaké třídě. Tabulka by byla méně přehledná, ale také paměťově neefektivní (i nevyplněné hodnoty zabírají na disku paměťový prostor o velikosti, která je dána datovým typem).

U relací typu 1:1 si tedy můžeme vybrat, do které tabulky informaci přidáme. Vybíráme tu tabulku, kde lze předpokládat méně nevyplněných hodnot.

U relací typu 1:N už si vybírat nemůžeme. V našem příkladu student chodí do jedné třídy, ale do třídy chodí více studentů. Informaci lze přidat jedině do tabulky studenti. Přidáme atribut trida a do něj budeme zapisovat klíče tříd.

studenti

cislo	jmeno	prijmeni	trida
123	Anna	Nováková	1.A
203	Jan	Dvořák	1.B
126	Petr	Černohorský	2.A
242	Marie	Komárková	1.A

U relací M:N (studium) nelze přidat atribut ani do jedné z tabulek. Proto musíme vytvořit novou tabulku, ve které zachytíme pomocí klíčů studenty a předměty, které studují.

studium

student	predmet
123	M
123	F
123	IVT
203	M
203	F

Novou tabulku můžeme vytvořit i pro typy relací 1:1 a 1:N zejména v případech, když by v příslušných atributech zůstávalo mnoho prázdných polí.

Finální struktura tabulek z našeho příkladu je následující:

ucitele

oscislo	jmeno	prijmeni
39	Milan	Solich
12	Petr	Martiník
5	Věra	Smutná
14	Anděla	Nebeská
25	Vladimír	Koutný

tridy

nazev	tridni
1.A	39
1.B	5
2.A	25
2.B	14

studenti

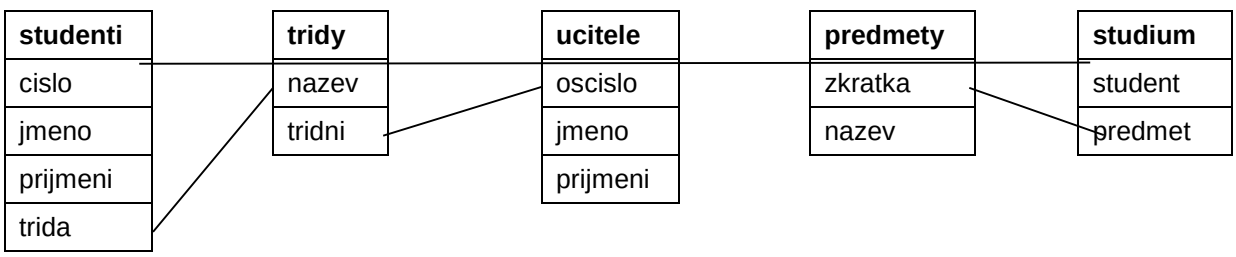
cislo	jmeno	prijmeni	trida
123	Anna	Nováková	1.A
203	Jan	Dvořák	1.B
126	Petr	Černohorský	2.A
242	Marie	Komárková	1.A

studium

student	predmet
123	M
123	F
123	IVT
203	M
203	F

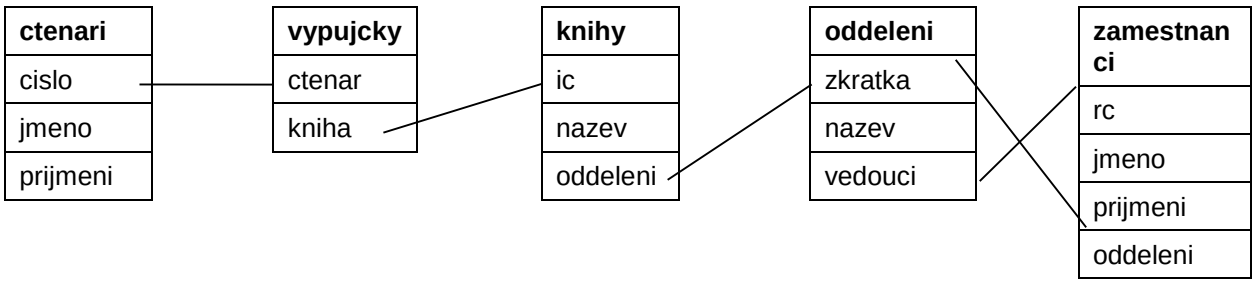
V tabulkách vidíme, že vícekrát se zapisují pouze hodnoty klíčů. Hodnoty ostatních atributů jsou zapsány vždy pouze v jedné tabulce. Proto jsme se snažili, aby klíče měly co nejmenší paměťové nároky. Tabulky se mohou zdát nepřehledné. Pro uživatele databáze lze ale připravit takové prostředí, že místo hodnot klíčů vidí a přiřazuje srozumitelnější informace (například jméno a příjmení studenta) a klíče si přiřazuje databázový systém.

Strukturu relačních databází budeme zapisovat jednodušeji. Nebudeme zapisovat jednotlivé záznamy (ty sloužily pouze pro pochopení). Zapíšeme pouze tabulky, jejich atributy a relace naznačíme spojnicemi příslušných atributů.



Při vyplňování tabulek by mohlo docházet k problémům. Například v atributu trida v tabulce studenti by měl být vždy hodnota, která se nachází v jednom záznamu v tabulce tridy (artibut nazev). Systém by měl pohlídat, aby hodnoty neexistující v tabulce tridy nebylo možné zadat. Také změnou hodnoty atributu nazev v tabulce tridy by mohlo dojít k tomu, že atributu trida v tabulce studenti už nebude odpovídat žádný záznam. Databáze by nám tedy neměla změnu dovolit nebo by měla automaticky změnit i související hodnotu. Také by se mohlo stát, že odstraníme záznam, se kterým souvisí nějaký údaj v jiné tabulce. Těmto problémům lze předcházet vhodným nastavením tzv. referenční integrity.

Sestavte strukturu relační databáze pro evidenci knih, oddělení, zaměstnanců a čtenářů v knihovně.



Sestavte schéma relační databáze podle E-R modelu z vašeho úkolu z minulé kapitoly.

6 Relační databáze – test (VY_32_INOVACE_286)

Fotbal

Chceme evidovat informace o fotbalových týmech a hráčích, kteří mohou hrát jen za jeden tým. Každý tým má jednoho trenéra, který už nemůže trénovat jiný tým. Každý tým může mít více sponzorů, kteří mohou financovat i jiné týmy. Sestavte strukturu relační databáze pro evidenci těchto skutečností (tabulky, klíče, základní atributy, relace naznačte spojnicemi atributů).

Úkoly

Firma je rozdělena na několik oddělení. Každý zaměstnanec pracuje v jednom oddělení. Dále chceme evidovat informace o plnění úkolů. Pro každý úkol je vždy určena z řad zaměstnanců jedna odpovědná osoba a dále se na něm může podílet libovolný počet zaměstnanců z různých oddělení. Sestavte relační model pro evidenci těchto skutečností (tabulky, klíče, základní atributy, relace naznačte spojnicemi atributů).

Poliklinika

Na poliklinice pracují lékaři a sestřičky a navštěvují ji pacienti. Každý lékař má svou jedinou odbornost (obvodní nebo různé specializace). Pro každou odbornost (včetně obvodních) je jmenován jeden vedoucí lékař. Každý lékař má jednu nebo více sestřiček, ale každá sestřička pracuje pouze u jednoho lékaře. Všichni pacienti mají svého jediného obvodního lékaře, ale mohou také navštěvovat různé specializované lékaře. Sestavte relační model pro evidenci těchto skutečností (tabulky, klíče, základní atributy, relace naznačte spojnicemi atributů).

Dodávky

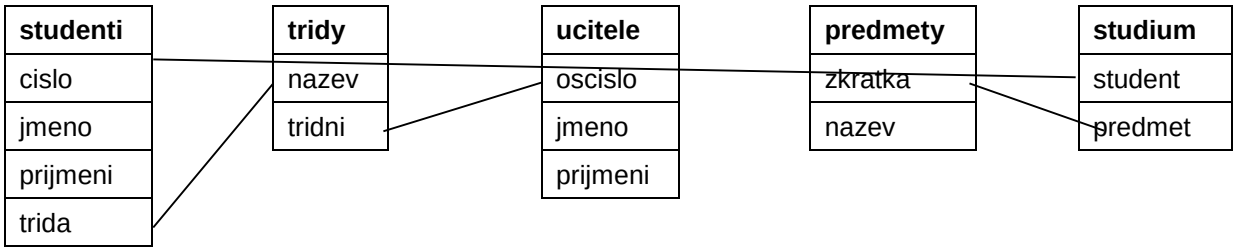
Firma nakupuje výrobky od různých dodavatelů. Jeden výrobek může dodávat více firem. Výrobky jsou rozdělené do kategorií (každý výrobek může patřit pouze do jedné kategorie). Dále je třeba evidovat jednotlivé dodávky, tj. informace o tom, které výrobky od které firmy byly dodány, v jakém počtu a v jaké ceně. Sestavte relační model pro evidenci těchto skutečností (tabulky, klíče, základní atributy, relace naznačte spojnicemi atributů).

Školící středisko

Školící středisko pořádá kurzy z jednotlivých předmětů. Z jednoho předmětu může být více kurzů. Každý kurz učí jeden lektor, navíc pro každý předmět je určen jeden z lektorů jako garant výuky. Dále se mají evidovat studenti navštěvující jednotlivé kurzy (student může navštěvovat více kurzů). Sestavte relační model pro evidenci těchto skutečností (tabulky, klíče, základní atributy, relace naznačte spojnicemi atributů).

7 Relační databáze – Access (VY_32_INOVACE_287)

Základy práce s relačními databázemi v Accessu si ukážeme na struktuře z minulé kapitoly. Kromě toho se stručně seznámíme s dalšími pojmy.



Spusťte program Access a vytvořte databázi s názvem skola (soubor skola.accdb). Nadefinujte strukturu všech tabulek. Data zatím nezadávejte. Datové typy atributů, které propojují tabulky, musí být stejné. Pokud je klíčový atribut typu Automatické číslo, odpovídající atribut v související tabulce musí být Dlouhé celé číslo. První tabulka se vytváří automaticky, každou další tabulku vytvořte na záložce Vytvoření, nejlépe bude zvolit Návrh tabulky.

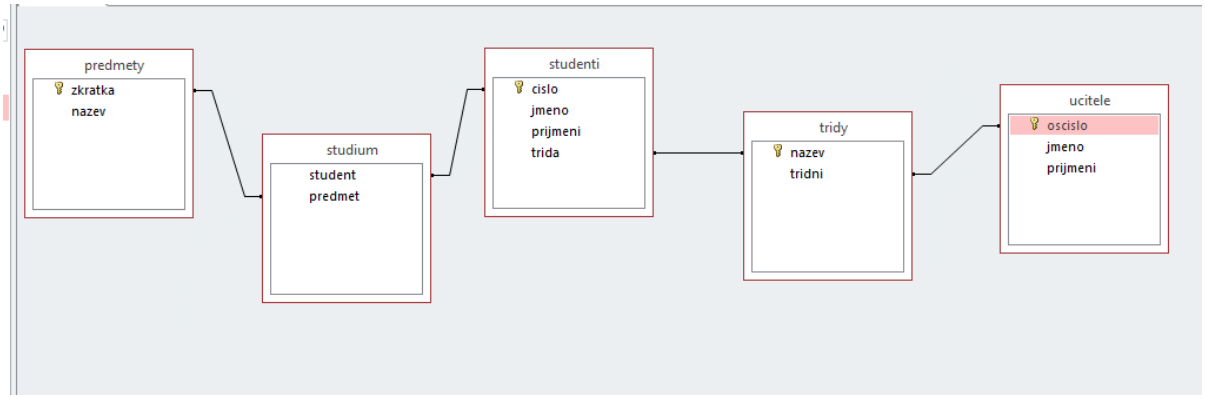
V každé tabulce nastavte klíč. Klíče, které si vytvoří Access (ID typu Automatické číslo) odstraňte. Klíč nastavíte volbou Primární klíč v horní nabídce.

V tabulkách lze zadat některá tzv. integritní omezení: omezení hodnot, vztahy mezi atributy, nutnost zadat, unikátnost a další. Databázový systém potom sám hlídá, zda zadávané hodnoty splňují dané podmínky. Pokud nesplňují, nedovolí hodnotu do databáze zapsat. Požadavky na integritu dat se zadávají v Návrhovém zobrazení v dolní části Vlastnosti pole. Kromě integritních omezení lze zadat i další vlastnosti pro zjednodušení přípravy kompletní databázové aplikace.

Další významnou vlastností polí jsou je indexování. Indexy umožňují rychle vyhledávat a uspořádávat záznamy. Uživatel neurčuje, který index se má použít. Jednoduchý index je tvořen pouze jedním atributem a chceme-li, aby byl atribut indexován, nastavíme ve Vlastnostech pole hodnotu Indexovat na Ano. Indexy mohou být tvořeny i více atributy v pořadí důležitosti. Tyto se zadávají volbou Indexy v horní nabídce. V jedné tabulce může nadefinováno i více indexů. Indexy volíme u takových atributů (nebo jejich skupin), u kterých předpokládáme, že se podle nich bude často třídit nebo vyhledávat. Klíče jsou indexem automaticky.

Po nadefinování tabulek bychom ještě měli určit relace, tj. přes které atributy jsou tabulky propojeny. Schéma relace nadefinujeme volbou Relace na záložce Databázové nástroje. Ve

vstupní tabulce volíme a přidáváme tlačítkem Přidat tabulky (mohli bychom i dotazy), které chceme propojit. Zvolte postupně všechny tabulky a přidejte je na pracovní plochu a zavřete dialogové okno tlačítkem Zavřít. Relaci mezi tabulkami nastavíte přetažením příslušného atributu na související atribut v druhé tabulce. Dialogové okno, které žádá souhlas s vytvořením relace, vždy zkontrolujte a potvrďte. Relace jsou vyznačeny spojnicemi příslušných atributů. Taháním je vhodné umístit tabulky co nejpřehledněji. Výsledné relační schéma může vypadat takto:



Kliknutím můžete kteroukoli spojnici atributů označit. Lze ji potom smazat (Delete) nebo dvojitým kliknutím na spojnici můžete nastavit vlastnosti relace. Pro každou relaci lze nastavit pravidla už dříve zmiňované referenční integrity, ale to ponecháváme vašemu dalšímu studiu.

Relační schéma si uložte.

Přejděme k tabulkám. Každou z tabulek si otevřete a zadejte do ní několik záznamů. Problematické může být zadávání hodnot klíčů pro zajištění vztahů mezi tabulkami. Nemůžeme vždy po uživateli chtít, aby si pamatoval a vyplňoval klíče z ostatních tabulek. Často to bývají uměle vytvořená čísla generátorem automatických čísel. Takové klíčové hodnoty uživatelům ani nebudeme ukazovat. Jednu z možnosti zajištění komfortnějšího zápisu klíčů do tabulek si ukážeme v definici tabulky tridy, kde každé třídě přiřadíme třídního učitele. K tomu si nejdříve připravte několik záznamů v tabulce ucitele. Pak se v tabulce tridy přepněte do Návrhového zobrazení. Označte atribut tridni a v dolní části Vlastnosti pole se přepněte na záložku Vyhledávání. V poli Zobrazit ovládací prvek zvolte Pole se seznamem. Tento prvek se běžně nazývá Combo box. Nadefinujeme rozbalovací seznam, kdy uživatel kliknutím na šipku vybere příslušný prvek ze seznamu. Do databázové tabulky se zapíše klíč. K tomu nastavte:

- Typ zdroje řádků: tabulka nebo dotaz,
- Zdroj řádků: ucitele (ze které tabulky chceme seznam vytvořit),
- Vázaný sloupec: 1 (ve kterém sloupci tabulky je klíč, který chceme zapsat do naší tabulky),

- Počet sloupců: 3 (kolik sloupců se má v seznamu zobrazovat),
- Hlavičky sloupců: ne
- Šířky sloupců: 1;2;2 (v centimetrech, odděleno středníkem, pokud nastavíme 0, nebude se sloupec zobrazovat).

Ostatní hodnoty a možnosti si prostudujte podle zájmu sami. Podobná vyhledávací pole si vytvořte pro nastavení relací v ostatních tabulkách. Jsou i další možnosti přehledného zadávání klíčů, určitě by byl problém se zadáváním relací typu M:N, ale to už je na podrobnější studium.

Nezapomeňte ve všech tabulkách zapsat několik záznamů.

Nakonec ještě jeden pojem a tím je hodnota Null. V mnoha případech nebude v záznamu některá z hodnot vyplněna. Buď daný objekt tuto vlastnost nemá, nebo ji ještě nemáme zjištěnu a doplníme ji později. Klíče a základní informace (jméno a příjmení, název,...) by prázdné zůstat neměly. Null je obecné označení pro nezadanou hodnotu libovolného typu. Její význam je tedy nezadáno nebo neznámá hodnota. Určitě je zajímavé vyhodnocování aritmetických, relačních a logických operací s hodnotou Null. U operací s logickými hodnotami pravda a nepravda musíme počítat ještě se třetí pravdivostní hodnotu Null ve smyslu „nevím“.

Nadefinujte tabulky, vytvořte relační schéma a zadejte záznamy vaší úlohy z předchozích kapitol.

8 SQL – select (VY_32_INOVACE_288)

V databázových tabulkách je velmi mnoho dat. Často nás budou zajímat pouze některé informace, které by se v tabulkách hledaly velice složitě a mohlo by dojít k přehlédnutí. K výběru pouze požadovaných informací slouží dotazy (query).

Dotazy umí:

- vybrat atributy a určit pořadí jejich zobrazení (faktorizace),
- „vypočítat“ nové hodnoty,
- vybrat záznamy (filtrování),
- spojit informace z více tabulek,
- uspořádat informace,
- seskupit záznamy do skupin,
- vypočítat souhrnné hodnoty za jednotlivé skupiny nebo za celou tabulku.

Pro popis operací jsou definovány dotazovací jazyky. Nejznámějším z dotazovacích jazyků je jazyk SQL (Structured Query Language). Často se používá také jazyk QBE (Query By Example), ve kterém se dotazy pokládají na základě příkladu v tabulce. Ukážeme si základy jazyka SQL. Dotazy budeme psát v textové podobě. Tento jazyk má přesně definovanou syntaxi a sémantiku. Syntaxe a možnosti se mohou v různých verzích SQL lišit, budeme vycházet z pravidel definovaných v programu MS Access. Všechny výše uvedené operace lze realizovat pomocí příkazu SELECT. Dotazovací jazyky mají i příkazy pro definici a modifikaci struktury tabulek (DDL – Data Definition Language) a pro zadávání, mazání a změnu záznamů v tabulkách (DML – Data Manipulation Language). Obsahují tedy všechny prostředky pro práci s databázovou tabulkou. Vlastní data jsou umístěna na nějakém místě. Uživatel má k dispozici pouze nějaké softwarové rozhraní, ve kterém buď přímo zapisuje dotazy nebo prostřednictvím formulářů volí akce a edituje data a všechny jeho požadavky jsou převedeny na dotazy. Dotazy jsou zaslány na databázový server, který buď modifikuje databázi (DDL, DML) nebo v případě příkazu SELECT vybere a vrátí informace v požadované podobě.

Při práci s databází Access budeme mít vše (uživatelské rozhraní, server pro interpretaci dotazů i data) na jednom počítači a v jednom souboru. Pro zjednodušení a zefektivnění práce s daty je výhodnější síťová databáze (klient – server), kde na více počítačích (klienti, stanice) je umístěno uživatelské rozhraní pro práci s daty. Vlastní data jsou uložena na jednom místě v počítačové síti.

Uživatelé zasílají ze stanic dotazy, které jsou prostřednictvím databázového serveru vykonávány na datech. V případě dotazu SELECT jsou vybraná data vrácena na stanici. Přístup k databázi má tedy jediný program. Pokud jsou data uložena na více místech (více počítačů v síti nebo na internetu) a přistupuje se k nim přes jeden databázový server, jde o tzv. distribuovanou databázi.

Základem jazyka SQL je příkaz SELECT, který se skládá z následujících částí:

- SELECT - co se má vybrat – atributy,
- FROM – odkud se to má vybrat (tabulka nebo více tabulek, dotazy),
- WHERE – které záznamy se mají vybrat (podmínky),
- ORDER BY – jak mají být záznamy uspořádány (třídění),
- GROUP BY – do jakých skupin mají být záznamy seskupeny,
- HAVING – které skupiny se mají vybrat.

Výsledkem příkazu SELECT je vždy relační tabulka.

V tabulce zam jsou informace o zaměstnancích firmy.

zam

OC	oddeleni	jmeno	prijmeni	narozen	stav	jesef	plat
42	100	Jan	Novák	12.12.1965	S	x	14500
25	200	Petr	Novotný	1.11.1968	S	x	12000
33	200	Marcela	Veselá	5.5.1964	V		8000
34	100	Janička	Neveselá	15.5.1969	R		10000
55	500	Eleonora	Tachecí	25.12.1965	M	x	10000
1	300	Petr	Horák	5.5.1967	S	x	11000
2	200	Jana	Novotná	12.1.1959	V		9000
54	300	Jana	Špačková	15.2.1954	M		9000

Atribut OC je osobní číslo zaměstnance. Oddělení jsou zapsána celými čísly. Stav je zakódován pomocí písmen (Krátký text délky 1) s významy: S – svobodný/svobodná, V – vdovec/vdova, R – rozvedený/rozvedená, M – ženatý/vdaná. V atributu jesef je zapsána pravdivostní hodnota true (označeno x) v případě, že je dotyčný pracovník šéfem oddělení. Jinak je tam hodnota false (neoznačeno). Další atributy jsou jasné.

Napište dotazy SQL pro výběr informací z tabulky zam.

Seznam všech zaměstnanců (jméno a příjmení).

```
SELECT jmeno, prijmeni FROM zam
```

V části SELECT popisujeme seznam atributů (výběr a pořadí), které chceme zobrazit. Prvky seznamu se oddělují čárkou. V části FROM určíme, ze které tabulky se mají informace vybírat. Obě tyto části jsou povinné, musíme je vždy uvést. Ostatní části jsou volitelné.

Výsledkem je tabulka, která bude mít dva sloupce (jmeno a prijmeni). Bude obsahovat všechny záznamy, záznamy nebudou nijak seřazené (budou zobrazeny tak, jak jsou uloženy v tabulce).

Seznam všech zaměstnanců (nejdříve příjmení a potom jméno).

```
SELECT prijmeni, jmeno FROM zam
```

Stejná tabulka jako v předchozím příkladu, pouze budou prohozené sloupce.

Seznam všech zaměstnanců (všechny atributy).

```
SELECT * FROM zam
```

Jestliže nám na výběru atributů nezáleží, nahradíme jejich seznam hvězdičkou. Ve výsledné tabulce se zobrazí všechny atributy v pořadí definované v tabulce.

V dalších dotazech budeme vybírat všechny atributy (pokud nebude řečeno jinak).

Seznam všech zaměstnanců seřazený podle platu od nejmenšího po největší (vzestupně).

```
SELECT * FROM zam ORDER BY plat
```

V části ORDER BY uvádíme atribut, podle jehož hodnot má být výsledná tabulka seřazena.

Seznam všech zaměstnanců seřazený podle platu od největšího po nejmenší (sestupně).

```
SELECT * FROM zam ORDER BY plat desc
```

Třídění může být vzestupné (ascending - od nejmenší po největší hodnotu nebo od A do Z) nebo sestupné (descending - od největší po nejmenší nebo Z do A). Pokud chceme řadit podle atributu sestupně, napíšeme za něj zkratku desc. Vzestupné třídění je automatické (nemusíme nic psát). Sestupné třídění můžeme v části ORDER BY zapsat u kteréhokoli atributu v seznamu.

Seznam všech zaměstnanců (jméno a příjmení) seřazený abecedně.

```
SELECT jmeno, prijmeni FROM zam ORDER BY prijmeni, jmeno
```

Příjmení více zaměstnanců může být shodné. Proto definujeme druhé kritérium, které rozhodne o pořadí v případě rovnosti hodnot prvního atributu. Kritérií může být více, zapisujeme je v seznamu (oddělené čárkami) v pořadí důležitosti.

Seznam všech zaměstnanců seřazený od nejmladšího po nejstaršího.

```
SELECT * FROM zam ORDER BY narozen desc
```

Na hodnoty typu Datum/čas můžeme z hlediska třídění pohlížet jako na čísla. Jako první má být nejmladší zaměstnanec, tj. ten, jehož datu narození je největší. Musíme tedy třídit sestupně.

Seznam všech zaměstnanců seřazený podle oddělení a v rámci oddělení abecedně.

```
SELECT * FROM zam ORDER BY oddeleni, prijmeni, jmeno
```

Dá se předpokládat, že v každém oddělení bude více zaměstnanců. Proto je vhodné zvolit další kritérium třídění. Výsledná tabulka tedy zobrazí zaměstnance z jednoho oddělení pohromadě a budou seřazení abecedně.

Seznam zaměstnanců (jméno, příjmení, plat), kteří mají plat větší než 10000.

```
SELECT jmeno, prijmeni, plat FROM zam WHERE plat>10000
```

V dalších příkladech se budeme zabývat filtrováním dat (tedy výběrem záznamů). V části WHERE vždy zapíšeme podmínku. Výsledná tabulka bude obsahovat pouze ty záznamy, které podmínku splňují. Pokud bychom požadovali uspořádání tabulky, můžeme na konec přidat část ORDER BY.

Seznam zaměstnanců z oddělení 100.

```
SELECT * FROM zam WHERE oddeleni=100
```

Pro porovnávání můžeme použít relační operátory =, <>, >, >=, <, <=.

Seznam zaměstnanců, kteří se jmenují Josef.

```
SELECT * FROM zam WHERE jmeno="Josef"
```

Textové hodnoty píšeme do uvozovek.

Seznam zaměstnanců z oddělení 200, kteří mají plat větší než 10000.

```
SELECT * FROM zam WHERE oddeleni=200 and plat>10000
```

Mají platit dvě podmínky zároveň, použijeme tedy logický operátor and. Připomeňme si, že podmínka je pravdivá, když jsou pravdivé obě části.

Seznam zaměstnanců z oddělení 100 nebo 200.

```
SELECT * FROM zam WHERE oddeleni=100 or oddeleni=200
```

Má platit jedna z podmínek, použijeme tedy operaci or. Ta je pravdivá, když je pravdivá alespoň jedna z relací.

Příjmení všech, kteří jsou svobodní nebo rozvedení a mají plat menší než 10000.

```
SELECT prijmeni FROM zam WHERE (stav="S" or stav="R") and plat<10000
```

Pokud kombinujeme and a or, jsou závorky důležité.

Seznam všech šéfů oddělení.

```
SELECT * FROM zam WHERE jesef
```

Jesef je atribut typu Ano/ne (pravdivostní hodnota). Mohli bychom tedy podmínku napsat jesef = true. Tato podmínka je pravdivá (má hodnotu true), když hodnota jesef je rovna true. Není pravdivá (má hodnotu false), když hodnota jesef je false. Má tedy stejnou hodnotu jako hodnota atributu jesef a porovnání s hodnotu true je možné v podmínce vynechat. Ušetřili jsme tím operaci a tím zrychlili vyhodnocování dotazu.

Seznam zaměstnanců, kteří nejsou šéfové oddělení a mají plat větší než 10000.

```
SELECT * FROM zam WHERE not jesef and plat>10000
```

Podmínku, která rozhoduje o tom, zda daný zaměstnanec není šéfem žádného oddělení, jsme mohli napsat jesef = false. Výsledná hodnota této podmínky je přesně opačná než hodnota atributu jesef. Proto ji stačí negovat operátorem not. Not je operátor negace a obrací pravdivostní hodnotu.

Z tabulky studenti vyberte požadované informace pomocí dotazů SQL.

studenti

jmeno	prijmeni	narozen	bydliste	prumer	trida	ukoly	zaloha	vydaje	klasifikace
Jiří	Novák	1.1.1980	Hlučín	1,58	1.A	X	200	150	2
František	Dvořák	2.5.1980	Ludgeřovice	2,14	2.A	X	130	150	1
Alena	Moravcová	5.7.1984	Hlučín	1,24	2.A		200	200	
Marie	Adamcová	1.8.1983	Hať	1	4.B		100	90	
Milan	Ondruš	8.5.1984	Hlučín	2	2.B	X	120	150	1
Václav	Obrusník	5.5.1984	Kozmice	3	3.B	X	150	150	3

S podobnou tabulkou už jsme pracovali v předchozích kapitolách, takže by významy atributů a jejich typy měly být jasné.

Vytvořte:

Seznam studentů (jméno a příjmení), kteří mají známku 1.

Seznam studentů, kteří bydlí v Hlučíně, seřazený podle věku od nejstaršího po nejmladšího.

Seznam studentů, seřazené podle tříd a v rámci tříd podle abecedy.

Seznam studentů, kteří jsou z druhého ročníku a mají známku horší než 2.

9 SQL – select (VY_32_INOVACE_289)

V této části budeme pracovat s tabulkami z předchozí kapitoly. Procvičíme a rozšíříme. Strukturu tabulky si připomeneme pouze výčtem atributů.

Pro tabulku zam napište dotazy, které poskytnou:

zam
oddeleni
jmeno
prijmeni
narozen
stav
jesef
plat

Seznam všech svobodných šéfů oddělení seřazený od nejstaršího po nejmladšího, zajímají nás jejich platy.

```
SELECT jmeno, prijmeni, plat FROM zam WHERE stav="S" and jesef ORDER BY
narozen
```

Všichni šéfové s platem menším než 12000.

```
SELECT * FROM zam WHERE jesef and plat<12000
```

Všichni nešéfové s platem od 9000 do 13000 (včetně krajních hodnot).

```
SELECT * FROM zam WHERE not jesef and plat>=9000 and plat<=13000
```

Všichni Petři z oddělení 100 a 200.

```
SELECT * FROM Zam WHERE (Jméno="Petr") AND (Oddělení=100 OR Oddělení=200)
```

Všechny údaje o těch zaměstnancích, kteří nejsou z oddělení 500, mají plat větší než 15000 a jsou rozvedení.

```
SELECT * FROM zam WHERE not (oddeleni=500) and plat>15000 and stav="R"
SELECT * FROM zam WHERE oddeleni<>500 and plat>15000 and stav="R"
```

Oba dotazy jsou možné, druhý šetří operaci negace.

Seznam zaměstnanců a údaj o jejich dani z příjmu (15 % z platu) seřazení abecedně abecedně.

```
SELECT jmeno, prijmeni, 0.15*dan AS dan FROM zam ORDER BY prijmeni, jmeno
```

Výsledná tabulka bude mít tři sloupce, První se bude jmenovat jmeno, druhy prijmeni, třetí dan. Hodnoty třetího sloupce nejsou nikde v databázových tabulkách přímo obsaženy, dotaz je spočítá jako nové hodnoty a vrátí je v tabulce. Do databáze je nikam nezapisuje. Pro výpočty můžeme použít operátory pro sčítání (+), odčítání (-), násobení (*) a dělení (/). Za slovem AS definujeme název sloupce. Pokud bychom uvedli pouze 0.15*plat, sloupec by dostal nějaké systémové označení, které by bylo nesrozumitelné. Takto si můžeme pojmenovat libovolný sloupec. Pokud

by se nám například nelíbilo, že názvy sloupců neobsahují háčky, čárky a velká písmena, mohli bychom to vyřešit dotazem

```
SELECT jmeno AS Jméno, prijmeni AS Příjmení, 0.15*dan AS Daň FROM zam ORDER
BY prijmeni, jmeno
```

Seznam zaměstnanců spolu s vypočítaným atributem daň, kteří mají daň větší než 1000.

```
SELECT jmeno, prijmeni, 0.15*dan AS dan FROM zam WHERE 0.15*plat>1000
```

Jednosloupcový seznam zaměstnanců, kde v každém řádku bude jméno a příjmení.

```
SELECT jmeno & " " & prijmeni AS zaměstnanec FROM zam
```

Místo aritmetických operátorů jsme použili operátor pro spojování řetězců &. Tato operace spojí dva řetězce v uvedeném pořadí a vytvoří z nich jediný. Kdybychom spojili pouze jmeno a prijmeni (jmeno & prijmeni), chyběla by uvnitř mezera. Proto musíme spojit 3 řetězce. Dva jako hodnoty atributů z tabulky a mezi nimi řetězec, který obsahuje jediný znak – mezeru.

Seznam zaměstnanců, u kterých chybí křestní jméno.

```
SELECT * FROM zam WHERE jmeno IS NULL
```

Pokud pole v záznamu nemá zadánu hodnotu, nazývá se tato prázdná hodnota označením NULL. V testu na hodnotu NULL se nepoužívá relační operátor =, ale operátor IS. Na hodnotu NULL můžeme testovat atribut libovolného typu.

Seznam jen těch zaměstnanců, kteří mají zapsáno datum narození.

```
SELECT * FROM zam WHERE not (narozen IS NULL)
```

Z tabulky studenti vyberte požadované informace pomocí dotazů SQL.

studenti
jmeno
prijmeni
narozen
bydliste
prumer
trida
ulohy
zaloha
vydaje
klasifikace

Seznam studentů (všechny atributy), kteří nemají splněné úkoly.

```
SELECT * FROM studenti WHERE not ukoly
```

Seznam jedničkářů z Ostravy nebo z Opavy.

```
SELECT * FROM studenti WHERE klasifikace = 1 and (bydliste="Opava" or
bydliste="Ostrava")
```

Studenti, jejichž průměr je nad 1,5, ale není horší než 2,00.

```
SELECT * FROM studenti WHERE prumer>1.5 and prumer<=2
```

Jméno a příjmení studentů a údaj o tom, kolik Kč se má studentovi vrátit.

```
SELECT jmeno, prijmeni, zaloha – vydaje AS vratit FROM studenti
```

Seznam neklasifikovaných studentů.

```
SELECT * FROM zam WHERE klasifikace IS NULL
```

Seznam studentů, kde v prvním sloupci je informace ve tvaru: „Jan Novák (1.A)“ a ve druhém sloupci průměrná známka.

```
SELECT jmeno & " " & prijmeni & " (" & trida & ")" AS student, prumer FROM
studenti
```

10 SQL – select – cvičení (VY_32_INOVACE_290)

Tabulka hráčů fotbalového týmu má strukturu:

hraci	
RC	rodné číslo
jmeno	jméno hráče
prijmeni	příjmení hráče
tym	název týmu, za který hraje
post	post, na kterém hraje (brankář, obránce, záložník, útočník)
narozen	datum narození
zapasy	počet sehraných zápasů
goly	počet vstřelených gólů
repre	zda hraje za reprezentaci (Ano/ne)

Sestavte dotazy SQL, které poskytnou tabulky:

Seznam hráčů (jméno a příjmení hráče, tým, post), kteří dali více než 5 gólů a nehrají za reprezentaci, seřazený podle týmů a v rámci týmu abecedně.

Seznam hráčů (všechny údaje), kteří hrají za Baník nebo Spartu a sehráli více než 20 zápasů, seřazené sestupně podle počtu zápasů.

Seznam reprezentačních útočníků (jméno a příjmení, počet gólů na zápas) seřazený od nejstaršího po nejmladšího.

Seznam hráčů ve tvaru: „František Veselý: Slavia Praha – útočník“.

V tabulce evidence knih v knihovně odpovězte pomocí dotazu SQL na uvedené dotazy.

knihy	
jautor	jméno autora knihy
pautor	příjmení autora knihy
nazev	
cena	
rok	rok vydání
nakup	datum nakupu
prez	zda kniha nesmí být půjčena domů (je prezenční)
pujceno	datum zapůjčení knihy
ctenar	jméno čtenáře, který má knihu půjčenu

Které knihy jsou momentálně v knihovně?

Nepůjčili jsme náhodou někomu prezenční knihu?

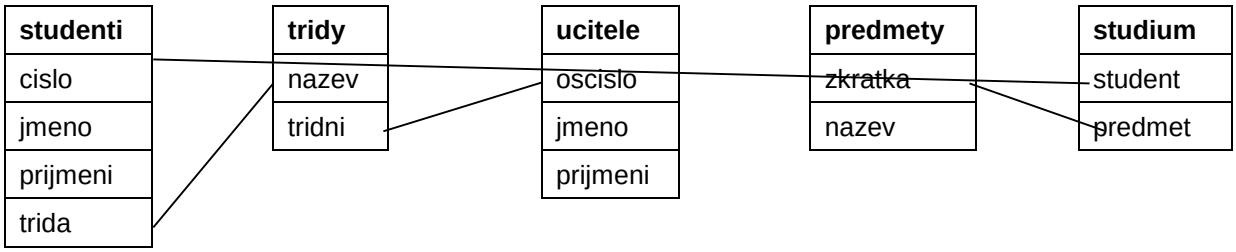
Kterým čtenářům pošleme upomínku, že už mají knihu půjčenou déle než měsíc? (Počítejte k vašemu aktuálnímu datu, zjistěte si formát zápisu data v dotazech SQL).

Nepůjčujeme mnoho drahých knih a neměli bychom jich více dát do prezenčních?

11 SQL - spojování tabulek (VY_32_INOVACE_291)

SELECT umí také spojit informace z více tabulek.

V kapitole 6 jsme došli ke struktuře databáze pro evidenci výuky ve škole.



Napište dotazy které poskytnou:

Seznam tříd a třídních učitelů

```
SELECT nazev, tridni FROM tridy
```

Tento dotaz pouze do jedné tabulky vypíše seznam tříd, ale u třídních jejich osobní čísla, což není dobré. Musíme názvy tříd spojit se jménem a příjmením učitele z tabulky ucitele.

```
SELECT nazev, jmeno, prijmeni FROM tridy, ucitele
```

Toto řešení je chybné. Pokud v části FROM uvedeme seznam dvou tabulek oddělený čárkou, vrátí nám databázový server kartézský součin tabulek, tzn. spojí každou třídu se všemi učiteli. Musíme specifikovat, které záznamy se z kartézského součinu mají vybrat.

```
SELECT nazev, jmeno, prijmeni FROM tridy, ucitele WHERE tridni = oscislo
```

Tento dotaz by už vrátil požadovanou tabulku. Není ale příliš efektivní a přehledný. Část WHERE bychom si měli nechat na případné další podmínky, které vyžadujeme. Spojení se dá definovat přímo v části FROM pomocí operátoru spojení INNER JOIN a popisem atributů, přes které je navázána relace.

```
SELECT nazev, jmeno, prijmeni FROM tridy INNER JOIN ucitele ON tridni =
oscislo
```

Dejte pozor, aby názvy atributů za slovem ON korespondovaly s pořadím tabulek v definici spojení. Tento dotaz je už v pořádku. Vrátí požadovaný seznam. Nebudou v něm ale třídy, které momentálně nemají třídního učitele (atribut tridni není definovaný – má hodnotu NULL). Toto můžeme napravit použitím levého (LEFT) nebo pravého (RIGHT) spojení.

Seznam tříd a třídních učitelů včetně tříd bez třídního učitele

```
SELECT nazev, jmeno, prijmeni FROM tridy LEFT JOIN ucitele ON tridni =
oscislo
```

Tento dotaz vrátí seznam všech tříd. Třídy, které mají definovanou hodnotu tridni, budou mít ve druhém a třetím sloupci jméno a příjmení třídního učitele. Třídy, kde hodnota tridni chybí, budou mít v těchto sloupcích prázdná pole.

Kromě levého spojení (LEFT JOIN) můžeme použít také pravé spojení (RIGHT JOIN).

```
SELECT nazev, jmeno, prijmeni FROM tridy RIGHT JOIN ucitele ON tridni =
oscislo
```

Tentokrát bychom dostali seznam všech učitelů a ti, kteří jsou třídními by měli v prvním sloupci název třídy. Ostatní by měli první sloupec prázdný.

Jestli použít levé nebo pravé spojení záleží pouze na tom, v jakém pořadí napíšeme tabulky v části WHERE.

Seznam studentů s názvy tříd

```
SELECT jmeno, prijmeni, nazev FROM studenti INNER JOIN tridy ON trida = nazev
```

Možná zbytečné spojení, protože název třídy je přímo klíčem, takže je zapsán v tabulce studentů. Využijme tento dotaz k porovnání jednotlivých spojení. Ve výsledku by nebyli studenti, kteří nejsou momentálně přiřazeni do žádné třídy ani třídy, ve kterých není žádný student.

```
SELECT jmeno, prijmeni, nazev FROM studenti LEFT JOIN tridy ON trida = nazev
```

V této tabulce už budou všichni studenti. Ti, kteří ještě nejsou zařazeni do žádné třídy, budou mít ve třetím sloupci (nazev) prázdné pole.

```
SELECT jmeno, prijmeni, nazev FROM studenti RIGHT JOIN tridy ON trida = nazev
```

Tento dotaz vrátí i třídy bez studentů.

Seznam studentů a jejich třídních učitelů

```
SELECT studenti.jmeno, studenti.prijmeni, ucitele.jmeno, ucitele.prijmeni
FROM (studenti INNER JOIN tridy ON trida = nazev) INNER JOIN ucitele ON
tridni = oscislo
```

V tomto příkladu řešíme dva problémy. Názvy atributů jsou v tabulkách stejné. Musíme rozlišit, ze které tabulky jsou. K tomu se používá tečkový zápis v podobě název tabulky, tečka a název atributu. Pokud nám nezáleží na konkrétních atributech a chceme vypsat všechny, píšeme:

```
SELECT studenti.*, ucitele.* FROM (studenti INNER JOIN tridy ON trida =
nazev) INNER JOIN ucitele ON tridni = oscislo
```

Druhý problém je, že tabulky nejsou spojené přímo, ale prostřednictvím tabulky tridy. Musíme tedy postupně spojit tabulky tři. Takto by šlo propojit i více tabulek.

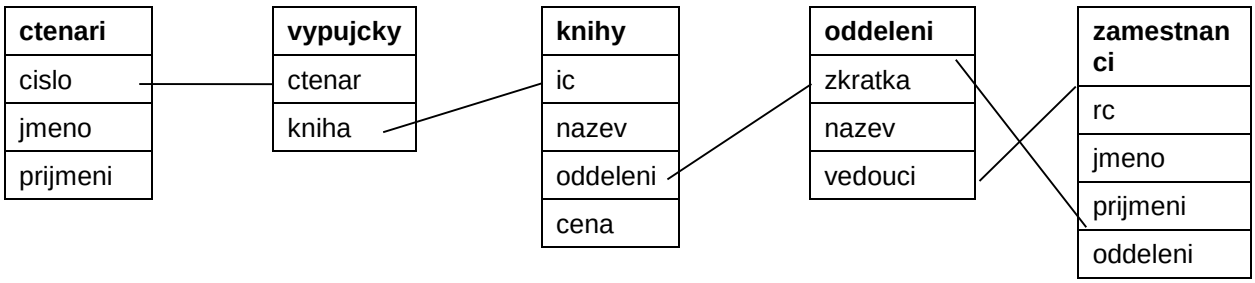
Seznam studentů s názvy předmětů, které studují.

```
SELECT jmeno, prijmeni, nazev FROM (studenti INNER JOIN studium ON cislo =
student) INNER JOIN predmety ON predmet = zkratka
```

Ve všech dotazech jsme mohli přidat WHERE a ORDER BY.

12 SQL – cvičení (VY_32_INOVACE_292)

V databázi knihovna pište dotazy.



Seznam knih, které nemají přiřazené inventární číslo (ic).

Seznam vedoucích jednotlivých oddělení.

Seznam všech momentálně vypůjčených knih, u každé vždy název a v závorce zkratka oddělení.

Seznam knih a názvů oddělení, do kterých jsou zařazeny, včetně knih, kterým ještě oddělení nebylo přiřazeno, seřazený podle názvů oddělení a v rámci oddělení podle ceny od nejdražší po nejlevnější.

Seznam všech knih s cenou nad 500 Kč seřazený abecedně a u knih, které jsou vypůjčené i jméno a příjmení čtenáře.

Seznam vypůjčených knih z oddělení „hudební“.

13 SQL – Access (VY_32_INOVACE_293)

Podle tabulky v kapitole 8 založte v Accessu databázi firma, v ní vytvořte tabulku zam a naplňte ji daty. Připravte si dotazy z této kapitoly.

Na záložce Vytvoření zvolte Návrh dotazu.V dialogovém okně přidejte tabulku studenti a dejte Zavřít. Dostaneme se do prostředí pro práci s dotazy.

S dotazy můžeme pracovat ve třech režimech zobrazení:

- Datový list – zobrazí výsledek dotazu,
- Zobrazení SQL – v něm můžeme přímo psát SQL,
- Návrhové zobrazení – v tomto prostředí se právě nacházíme, jde vlastně o prostředí pro dotazovací jazyk QBE, ale tím se nebudeme zabývat.

Přepněte se do Zobrazení SQL. Okno pro editaci dotazu obsahuje připravený text, který doplňte podle prvního příkladu z kapitoly 8. Na konce dotazů dávejte středník. Na výslednou tabulku se můžete podívat v zobrazení datového listu. Vyzkoušejte všechny příklady dotazů. Dotaz vždy vymažte a napište nový. Mohli bychom si je také uložit pro pozdější použití.

Takto si můžete vyzkoušet dotazy ze všech kapitol, určitě zkuste spojování tabulek. Tabulky si vždy nadefinujte a naplňte vhodnými daty. Zájemci se mohou podívat i do návrhového zobrazení a zkusit jazyk QBE. Jednotlivé způsoby zobrazení jsou navzájem propojené, takže to, co zadáte v SQL uvidíte v QBE a naopak.

14 SQL - skupiny, agregace (VY_32_INOVACE_294)

Agregací se rozumí shlukování. Ze záznamů v tabulkách lze vytvořit skupiny podle hodnot nějakého atributu. Za každou skupinu můžeme počítat souhrnné údaje pomocí agregačních funkcí. Jsou to počet (COUNT), součet (SUM), aritmetický průměr (AVG), minimum a maximum (MIN, MAX).

Jestliže použijeme některé agregační funkce na celou tabulku, jedná se o agregaci bez klíče. Souhrnné hodnoty se počítají ze všech řádků. Agregace s klíčem počítá souhrny pro jednotlivé skupiny.

Skupiny a souhrnné hodnoty se definují v dotazu SELCT, do kterého přidáme nové části GROUP BY a HAVING.

Základní možnosti si ukážeme na tabulce evidence počítačů.

pocitace	
ic	
nazev	
dodavatel	
cena	
rok	rok nákupu
umisteni	číslo místnosti
osoba	rodné číslo odpovědné osoby

```
SELECT COUNT(ic), AVG(cena), SUM(cena), MAX(cena) FROM pocitace
```

Dotaz vrátí jednořádkovou tabulku se čtyřmi sloupci, ve kterých budou postupně informace: počet počítačů, průměrná cena jednoho počítače, celková hodnota všech počítačů, cena nejdražšího počítače. Nevytvořili jsme žádné skupiny, jedná se o agregaci bez klíče. Souhrnné hodnoty se budou počítat ze všech záznamů v tabulce. Bylo by dobré přejmenovat hodnoty pomocí AS.

```
SELECT umisteni, COUNT(ic), SUM(cena) FROM pocitace GROUP BY umisteni
```

Toto je agregace s jednoduchým klíčem. Vytvoří se skupiny, kdy v každé skupině budou vždy počítače z jedné místnosti (budou mít stejnou hodnotu atributu umisteni). Za každou skupinu (tedy místnost) se určí počet počítačů a součet cen. Výsledkem bude tabulka se třemi sloupci: číslo místnosti, počet počítačů v místnosti a celková cena. V každém řádku budou informace o jedné místnosti.

```
SELECT umisteni, AVG(cena) FROM pocitace GROUP BY umisteni WHERE rok>2010
```

Dotaz vrátí průměrné ceny počítačů v jednotlivých místnostech, ale pouze těch počítačů, které jsou zakoupené po roce 2010.

```
SELECT umisteni, COUNT(ic) FROM pocitace GROUP BY umisteni HAVING COUNT(ic)>3
```

Dotaz vrátí počty počítačů v těch místnostech, ve kterých jsou více než 3 počítače.

Pro tabulku studentů zkuste napsat požadované dotazy.

studenti	
rc	rodné číslo
jmeno	
prijmeni	
trida	
skupina	
znamka	
zaloha	
vyska	

Počet studentů, průměr známek, celková vybraná částka, největší a nejmenší výška studenta.

Počty studentů v jednotlivých třídách a průměrná známka v každé třídě.

Totéž, co v předchozím příkladě, ale pouze třídy s průměrem horším než 2.00.

Počty studentů v jednotlivých třídách, kteří mají alespoň dvojku.

Počty studentů s jedničkou, dvojkou, ...

15 SQL - DML, DDL (VY_32_INOVACE_295)

Příkaz SELECT pouze vybírá a vrací informace z databázových tabulek. Uložené informace nijak nemění. Jazyk SQL obsahuje i příkazy, které umožňují měnit strukturu databáze (DDL – Ddata Definition Language) a manipulovat s databázovými záznamy (DML – Data Manipulation Language). Základními příkazy jsou:

- CREATE TABLE – vytvoření tabulky,
- ALTER TABLE – změna struktury tabulky,
- INSERT INTO – přidání záznamu do tabulky,
- DELET FROM – odstranění záznamu z tabulky,
- UPDATE – změna dat v tabulce.

Základní možnosti jazyka DDL a DML si ukážeme na jednoduchých příkladech.

```
CREATE TABLE studenti ( cislo INTEGER, jmeno CHAR(15),  prijmeni CHAR(20));
```

Vytvoří tabulku studenti s atributy: cislo (celé číslo), jmeno (řetězec 15 znaků), prijmeni (řetězec 20 znaků).

```
ALTER TABLE studenti ADD COLUMN narozen DATE;
```

Do tabulky studenti přidá další atribut narozen typu datum.

```
ALTER TABLE studenti ADD COLUMN ukoly LOGICAL;
```

Přidá atribut ukoly zápis pravdivostních hodnot.

```
ALTER TABLE studenti ADD COLUMN prumer REAL;
ALTER TABLE studenti ADD COLUMN znamka INTEGER;
```

Přidali jsme číselné atributy prumer a znamka.

```
CREATE UNIQUE INDEX stc ON studenti (cislo);
```

Vytvořili jsme klíč v tabulce studenti. Klíčem je atribut cislo a je uložen v indexu uloženém pod názvem stc.

```
DROP INDEX stc;
DROP TABLE studenti;
```

Tabulku jsme odstranili z databáze. Nejprve se musí odstranit index. Vytvoříme tabulku znovu, nastavíme v ní další vlastnosti.

```
CREATE TABLE studenti ( cislo integer NOT NULL UNIQUE, jmeno CHAR(15) NOT
NULL, prijmeni CHAR(20) NOT NULL, narozen DATE, znamka INTEGER CHECK znamka
BETWEEN 1 AND 5, prumer REAL, pohlavi CHAR(1) CHECK pohlavi IN („M“, “Z“),
ukoly LOGICAL, zaloha REAL);
```

Klíč byl tentokrát vytvořen přímo v definici tabulky, nemusíme pro něj vytvářet zvláštní index. Je to atribut cislo, který musí být zadán (NOT NULL) a jeho hodnoty musí být unikátní (UNIQUE). Integritní omezení lze nastavit i u dalších atributů (NOT NULL, CHECK).

```
ALTER TABLE studenti DROP COLUMN zaloha;
```

Odstranili jsme atribut zaloha z tabulky.

```
INSERT INTO studenti (cislo, jmeno, prijmeni, znamka, ukoly, prumer, narozen,
pohlavi) VALUES (5, ”František”, ”Dvořák”, 3, NO, NULL, „11.04.1996“, ”M”);
INSERT INTO studenti (cislo, jmeno, prijmeni, znamka, ukoly, prumer, narozen,
pohlavi) VALUES (3, ”Anna”, ”Veselá”, 2, YES, 2.30, „12.06.1997“, ”Z”);
INSERT INTO studenti (cislo, jmeno, prijmeni, znamka, ukoly, prumer, narozen,
pohlavi) VALUES (9, ”Jindřich”, ”Pospíšil”, NULL, YES, 1.25, NULL, ”M”);
```

Přidali jsme tři záznamy do tabulky studenti. Pořadí atributů je libovolné. Některé můžeme i vynechat.

```
INSERT INTO studenti (cislo, jmeno, prijmeni, znamka) VALUES (3, ”Josef”,
”Hlubek”, 5);
INSERT INTO studenti (cislo, prijmeni, znamka) VALUES (7, ”Novotný”, 4);
INSERT INTO studenti (cislo, jmeno, prijmeni, znamka, pohlavi) VALUES (12,
”Stanislava”, ”Hrubá”, 8, ”X”,);
```

Ani jeden z těchto příkazů nepůjde provést, SQL server je odmítne. První dotaz narazí na unikátnost klíčů. Druhý příkaz nedefinuje křestní jméno, které má nastaveno integritní omezení NOT NULL. Také třetí příkaz porušuje nastavená pravidla.

```
DELETE FROM studenti;
```

Odstraní všechny záznamy z tabulky studenti. Tabulka zůstane zachována, ale bude prázdná. I tak je to dost nebezpečný příkaz.

```
DELETE FROM studenti WHERE cislo=7;
```

Odstraní pouze studenta s číslem zadaným v podmínce. Zruší se pouze jeden záznam, neboť jde o unikátní klíč.

```
DELETE FROM studenti WHERE prumer>3;
```

Odstraní všechny studenty, kteří mají průměr horší než 3.

```
UPDATE studenti SET znamka=2, ukoly=YES WHERE cislo=5;
```

V tabulce studenti v záznamu s hodnotou klíče 5 (František Dvořák), nastaví hodnoty příslušných atributů.

```
UPDATE studenti SET znamka=NULL;
```

Zruší veškerou klasifikaci.

16 Excel – třídění (VY_32_INOVACE_296)

Práci s databázovými tabulkami umožňuje i tabulkový kalkulátor Excel. Poskytuje základní databázové operace:

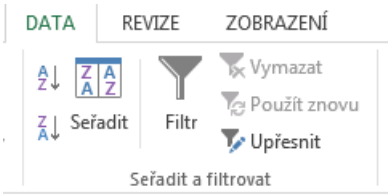
- třídění – uspořádání záznamů v tabulce,
- filtrování – výběr záznamů z tabulky,
- souhrny – seskupování záznamů a výpočet souhrnných hodnot,
- kontingenční tabulky – přehledné zobrazení agregovaných informací.

Kromě kontingenčních tabulek jsme všechny operace zapisovali v dotazech SQL.

Pro třídění musíme vždy zadat kritérium, tj. sloupec s atributem, podle jehož hodnot se má třídit. Pokud budou hodnoty ve sloupci stejné, musíme přidat další kritéria. U každého kritéria se nastavuje, zda se má jednat o vzestupné nebo sestupné třídění.

Otevřete si soubor studenti.xlsx. Pokud chceme vykonávat databázové operace s tabulkami, musíme vždy nastavit aktivní buňku někde v tabulce.

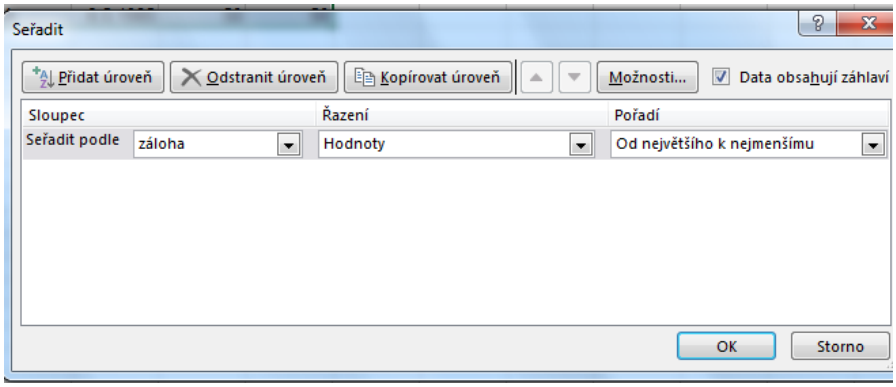
Nejrychlejší je třídění podle jednoho kritéria. Přepneme se do sloupce, podle kterého chceme třídit a na záložce DATA vybereme vzestupné nebo sestupné třídění.



Záznamy se v tabulce uspořádají podle hodnot vybraného kritéria. Dejte pozor, abyste neměli v tabulce označenou souvislou obdélníkovou oblast s více hodnotami. Uspořádaly by se pouze hodnoty v této oblasti.

Uspořádejte tabulku studentů:
Vzestupně podle výše vybrané zálohy.
Sestupně podle výše výdajů.
Vzestupně podle příjmení.
Od nejmladšího po nejstaršího studenta.

Vícekriteriální třídění nastavíme volbou Seřadit.



V dialogovém okně zvolíme vždy atribut a vzestupné nebo sestupné třídění. Další kritéria přidáme tlačítkem Přidat úroveň.

Uspořádejte tabulku studentů:

Abecedně.

Podle klasifikace a pro každou známku abecedně.

Podle tříd a v rámci každé třídy od nejhoršího po nejlepší průměrný prospěch.

Další způsob třídění uvidíte v následující kapitole o filtrování.

17 Excel – filtrování (VY_32_INOVACE_297)

Filtry umožňují vybírat z tabulky pouze záznamy splňující určité podmínky. Budeme opět pracovat s tabulkou studenti. Pro filtrování je v Excelu nástroj Filtr na záložce DATA. Po jeho zapnutí se v záhlaví každého sloupce objeví šipky pro výběr a nastavení požadované operace. Opětovným stisknutím tlačítka Filtr v nabídce šipky zmizí.

Po rozbalení seznamu se jako první možnost nabízí vzestupné nebo sestupné třídění. Pokud chceme třídit podle více kritérií, můžeme použít nástroj z minulé kapitoly nebo můžeme postupně třídit podle jednotlivých kritérií, ale musíme postupovat od nejméně významného kritéria po nejvýznamnější.

Seřaďte záznamy v tabulce:
Podle zálohy od největší po nejmenší.
Podle věku od nejstaršího po nejmladšího.
Podle tříd a v rámci tříd abecedně.
Nyní už k vlastnímu filtrování. Ve spodní části rozbaleného seznamu nabídek vidíte seznam všech hodnot (v našem případě tříd), které se v tabulce v daného sloupci vyskytují. Kterékoli z nich můžete zaškrtnout. Ve výsledné vyfiltrované tabulce pak budou pouze řádky se zvolenými hodnotami.

Zobrazte pouze studenty z prvních ročníků.
Filtr lze tlačítkem Vymazat zrušit. Ve vyfiltrované tabulce je možné definovat další filtry. Pokud nechcete zobrazovat všechny atributy, nemažte celé sloupce, abychom nepřišli o data. Měli byste uměl sloupec skrýt a zase ho obnovit.

Filtry umožňují i jiné zadání podmínek než pouze výběr ze seznamu. U mnoha číselných hodnot by to ani nebylo rozumné. Ve Filtrech čísel můžeme nastavit, jaké podmínky mají hodnoty splňovat (rovnosti, nerovnosti, intervaly). Zajímavá je volba Prvních 10, kde můžeme nastavit kolik položek nebo procent z počtu položek tabulky chceme zobrazit a jestli to mají být první nebo poslední hodnoty. Podívejte se také na nastavení Vlastního filtru. Prostudujte také volby u textových sloupců a kalendářních měsíců.

Filtrujte:

Všechny Petry nebo Pavly.

Studenty 1.A, kteří jsou klasifikování 5 nebo 4.

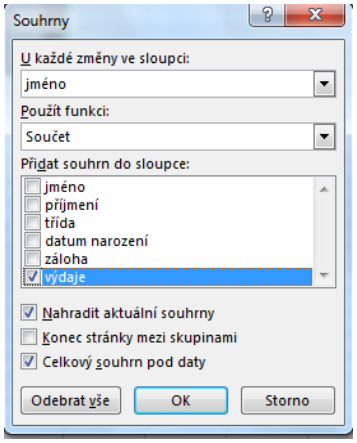
Studenty s jedničkou, kteří mají průměr horší než 2.

Studenty, kteří mají průměr horší než 1,5 ale nejsou horší než 2.

Pět studentů s nejhorším průměrem.

18 Excel – souhrny (VY_32_INOVACE_298)

Souhrny umožňují vytvářet skupiny a počítat za ně agregační funkce. Opět budeme pracovat s tabulkou studenti.xlsx. Nejdříve tabulku seřaďte podle tříd. V záložce DATA zvolte Souhrn.



Nejdříve nastavíme, podle hodnot kterého atributu se mají vytvářet skupiny. U každé změny ve sloupci vyberte atribut třída. Tabulka musí být předem seřazena podle tohoto atributu. Konec skupiny se vytvoří vždy před záznamem kde došlo ke změně hodnoty. U nesetříděné tabulky by to vedlo k chaosu.

Můžeme použít funkci součet, počet, průměr, maximum nebo minimum. Excel umožňuje i další funkce. Vyberte funkci Součet.

Dále zaškrtneme sloupce, ve kterých se má funkce počítat. Zvolme záloha a výdaje, jinde to asi nemá smysl.

Ve výsledné tabulce bude kromě souhrnů za jednotlivé skupiny i celková hodnota ze všech záznamů z tabulky. Můžeme zvolit, zda má být celkový souhrn pod daty nebo nad daty.

Po potvrzení tlačítkem OK se zobrazí požadované skupiny a souhrny. Jednotlivé záznamy i skupiny může skrýt nebo zobrazit pomocí tlačítek v levé části.

Skupiny a souhrny můžeme zrušit volbou Odebrat vše.

Určete počty žáků s jednotlivými známkami.
Určete průměrné známky v jednotlivých třídách.

19 Excel - kontingenční tabulky (VY_32_INOVACE_299)

Kontingenční tabulky přehledně zobrazují souhrnné informace z tabulek s možností volby požadovaných informací. Ukážeme si je na jednoduchém příkladu opět v tabulce studenti. Na záložce Vložení zvolte Kontingenční tabulka. V dialogovém okně potvrďte nebo nastavte oblast dat a umístění kontingenční tabulky.

V pravé části okna vyberte pole, která chcete mít v sestavě a přetáhněte pole do oblasti takto.

- FILTRY – třída,
- ŘÁDKY – bydliště,
- SLOUPCE – klasifikace,
- HODNOTY – klasifikace, rozbalte položku a v Nastavení polí hodnot zvolte Počet.

Ve výsledné kontingenční tabulce můžete volit třídy, bydliště a klasifikaci podobně jako u filtrování.

Zkuste rušit, přidávat a měnit jednotlivá pole a oblasti a studovat možnosti kontingenčních tabulek. Můžete zkusit vytvořit i kontingenční graf.

20 Excel - souhrnné cvičení (VY_32_INOVACE_300)

Kopírujte soubor firma.xlsx.

Během práce žádné vstupní informace nemažte.

Vytvořte tabulku souhrnů podle předlohy v souboru firma1.pdf, uložte pod názvem firma1.xlsx.

Vytvořte tabulku souhrnů podle předlohy v souboru firma2.pdf, uložte pod názvem firma2.xlsx.

V souboru firma.xlsx vytvořte nový list Výsledky.

Další úkoly vždy vyřešte, zkopírujte výsledek (bez nadpisu Seznam pracovníků, ale se záhlavím tabulky) na list Výsledky a zdrojovou tabulku hned vraťte do původní podoby.

Na listu Výsledky napište vždy do sloupce A číslo příkladu a hned od následujícího řádku nakopírujte vždy výslednou vyfiltrovanou tabulku, před dalším příkladem vynechejte jeden řádek.

Nic neformátujte.

Úkoly:

1. Všichni pracovníci z Hlučína seřazení abecedně

2. Seznam pracovníků podle jednotlivých oddělení seřazený v rámci každého oddělení podle platu od od největšího po nejmenší.

3. Všichni zaměstnanci z oddělení vedení nebo IT s platem menším než 20 000.

4. Pět zaměstnanců s nejnižším platem.

Uložte pod původním názvem.

Literatura

HERNANDEZ, Michael J., VIESCAS, John L.. *Myslíme v jazyku SQL – tvorba dotazů.* 1. vyd. Praha: Grada 2004. ISBN 80-247-0899-X.

POKORNÝ, Jaroslav. *Databázová abeceda*. 1. vyd. Veletiny: SCIENCE, 1998. ISBN 80-86083-02-0.