



# Počítače a ochrana dat – Návody do cvičení

Pavel Šenovský



## Obsah

<b>OBSAH.....</b>	<b>3</b>
<b>ÚVOD.....</b>	<b>5</b>
<b>1 STRUČNÝ A JEDNODUCHÝ ÚVOD DO TEORIE DATABÁZÍ.....</b>	<b>5</b>
1.1 Typologie databází.....	5
1.2 Návrh databáze.....	7
1.3 Pravidla návrhu tabulek.....	10
<b>2 MS Access.....</b>	<b>13</b>
2.1 Tabulky.....	14
2.2 Relace.....	17
2.3 Formuláře.....	20
2.4 Dotazy.....	28
2.5 Sestavy.....	31
<b>3 OPEN OFFICE BASE.....</b>	<b>31</b>
3.1 Tabulky.....	32
3.2 Relace.....	33
3.3 Formuláře.....	34
3.4 Dotazy.....	35
3.5 Sestavy.....	36
<b>LITERATURA.....</b>	<b>36</b>



## Úvod

Vítám Vás při studiu učebních textů Počítače a ochrana dat – Návody do cvičení. Tento text je zaměřen na bezproblémové zvládnutí základů práce s databázemi a ulehčit Vám jako studentům začátky v této oblasti.

Studium tohoto textu nepředpokládá žádné specifické znalosti z oblasti výpočetní techniky, kromě základů práce s programy MS Office. Jednotlivé příklady, které zde budu rozebírat budou řešeny v MS Access a Open Office Base. Při volbě jiné databáze se jednotlivé kroky mohou do určité míry lišit – principiálně by ale měly fungovat analogicky.

Celý text je rozdělen do tří částí. První, teoretické, ve které bude vysvětlena základní teorie okolo databází, tvorby modelů jako základních předpokladů správného řešení problému v databázovém prostředí. Druhé zaměřené na praktickou realizaci databáze pomocí MS Access a konečně třetí, ve které budeme databázi realizovat pomocí Open Office Base.

## 1 Stručný a jednoduchý úvod do teorie databází

### 1.1 Typologie databází

Databáze jsou jednou z nejstarších typů aplikací. Jejich počátek je prakticky totožný s počátkem nasazování počítačů k uchovávání údajů a za tuto dobu prošly poměrně výrazným vývojem.

Jako první typ, který se rozšířil byly *databáze stromové*, ty byly později nahrazeny *databázemi síťovými*. Architekturu těchto systémů zde rozebírat nemá smysl, protože oba výše zmíněné typy databází jsou beznadějně zastaralé a už se mnoho let nepoužívají. Byly nahrazeny během osmdesátých let *databázemi relačními* a dosud se jedná o nejpoužívanější druh databází. Relačními databázemi se budeme také zabývat výhradně v tomto textu (s výjimkou této kapitoly).

Relační databáze jsou tvořeny entitami, které představují virtuální představitele reálných objektů a obsahují jednotlivé atributy – tedy údaje, které o objektu budeme shromažďovat. Samotný název napovídá jaký bude druhý zásadní prvek relačních databáze, tedy relace, které představují vztahy mezi jednotlivými entitami.

Relační databáze můžeme ještě dělit do dvou skupin:

- 1) *velké databáze* – jsou určeny pro provoz na serveru. Předpokládá se u nich schopnost obsluhovat velké množství požadavků zároveň a schopnost pracovat s velmi rozsáhlými databázemi. Příkladem takových databází může být systém Oracle, Informix, MS SQLServer, MySQL, Postgress a další.
- 2) *Osobní databáze* – určeny k běžnému používání na klasických stolních počítačích – tedy jeden nebo několik málo uživatelů. Obsahují nástroje, které uživatelům usnadňují navrhování formulářů sestav a podobně. Cenou za snadnost použití je nižší výkon databáze a do určité míry omezená funkčnost – zejména v oblasti škálovatelnosti řešení apod. Mezi osobní databáze bychom mohli zařadit třeba Paradox, Fox Pro, DBase nebo MS Access.

V tomto textu se budeme zabývat pouze osobní databází MS Access.

Relační databáze přesáhly dvacátý rok svého života a vzhledem k rychlému vývoji v oblasti informačních technologiích by bylo podivné, kdyby se zrovna v této oblasti vývoj zastavil. Jako budoucího následníka relačních databází byly dlouho považovány *databáze*

*objektové*. S postupem času se ukázalo, že objektové databáze relační databáze nenahradí a dokonce se prakticky vůbec nerozšíří.

Základním rozdílem je filozofie konstrukce databází. Zatímco relační databáze si vystačí s entitami a relacemi, databáze objektové pracují s objekty a vztahy mezi nimi. Jednotlivé objekty jsou podobně jako entity obrazy reálných objektů, údaje o objektu jsou uchovávány ve vlastnostech objektu, tedy podobně jako atributy u entit. Objekty na rozdíl od entit obsahují i metody. Metody popisují chování objektů. Vztahy mezi objekty jsou také složitější, protože umožňují definovat i vztahy generalizační a výčty.

Spojení vlastností a metod do objektů je velmi blízké způsobu vnímání objektů reálného světa člověkem. Formalizace objektů a vztahu mezi nimi, které je nutná pro implementaci ve zvoleném databázovém prostředí je ale mnohem náročnější. Právě tento fakt pravděpodobně způsobil, že objektové databáze mají na trhu zanedbatelný podíl.

Posledním typem databází, ok kterých se zde zmíním jsou *XML databáze*. XML je zkratka pro Extensive Markup Language, tedy obecný značkovací jazyk. Jedná se o standard, který byl přijat v roce 1999 World Wide Web konsorciem. Jeho základním rysem je, že umožňuje vytvářet vlastní značkovací jazyky pro nejrůznější typy dokumentů a ty potom elektronicky velmi jednoduše zpracovávat.

Co je XML si nejlépe demonstrujeme na nějakém příkladě. Následující příklad demonstruje označování dopisu. Jednotlivé prvky dokumentu jsou ohraničeny tagy, které přiřazují těmto prvkům význam z hlediska sémantiky dokumentu.

```
<dopis>
  <hlavička>
    <odesilatel>
      <jméno>Pavel</jméno>
      <příjmení>Šenovský</příjmení>
      <ulice>Neuvedená 123</ulice>
      <město>Ostrava</město>
      <psč>123 45</psč>
    </odesilatel>
  </hlavička>
  <tělo_dopisu>
  Vážený pane,
  reaguji na Váš .....
  </tělo_dopisu>
</dopis>
```

Výše uvedený dokument XML není z hlediska validity navržen úplně korektně, ale jako demonstrace principu práce s údaji plně postačuje. V čem je tedy XML z hlediska databází zajímavé – odpověď je jednoduchá, ve flexibilitě. Pomocí XML lze popsat bez větších problémů prakticky jakýkoliv dokument – to je něco, co pomocí klasických databází je velmi náročné. Představte si, jak se snažíte do jednotlivých tabulek rozčlenit třeba daňové přiznání.

XML databáze lze rozčlenit do dvou skupin:

- 1) specializované XML databáze
- 2) relační databáze s podporou XML – obsahují funkce pro práci s XML, samotné datové prvky jsou ukládány do klasických tabulek.

Závěrem k rozdělení databází lze říci, že v současné době jsou relační databáze na trhu dominantní. Do budoucna se dá očekávat, že vývoj půjde směrem k většímu využití hybridních databází – relačních s podporou XML.

## 1.2 Návrh databáze

Vraťme se k relačním databázím a zamysleme se nad postupem návrhu a implementace databáze ve zvoleném databázovém prostředí. Návrh probíhá v několika fázích:

- 1) konceptuální,
- 2) tvorba ERD diagramu,
- 3) tvorba relačního modelu.

### Konceptuální fáze

V konceptuální fázi dochází ke shromažďování všech dostupných informací relevantních z hlediska řešení problému. Na počátku této fáze si jsme jistí pouze tím, že problém chceme vyřešit prostřednictvím databázového prostředí.

Jako základ pro řešení mohou sloužit dostupné formuláře. Pomocí nich jsme obvykle schopni odhalit jaké informace jsou v současné době nějakým způsobem zpracovávány a musí být tedy zahrnuty i do našeho řešení. Dalším hodnotným zdrojem informací jsou lidé, kteří v praxi problém, kterým se zabýváme, řeší.

Konzultace jsou významné tím, že během nich můžeme získat přehled o logistice procesů v rámci řešené oblasti a také se můžeme dovědět, jestli stávající údaje jsou skutečně využívány, popř. by bylo výhodné zahrnout do řešení nějaké jiné.

Výsledkem této fáze jsou shromážděné formuláře, poznámky s konzultací a myšlenkový model řešení – první představa o řešení, kterou jsme si vytvořili během konzultací. Tyto představy je nutné formalizovat předtím, než se je pokusíme implementovat ve zvoleném databázovém prostředí.

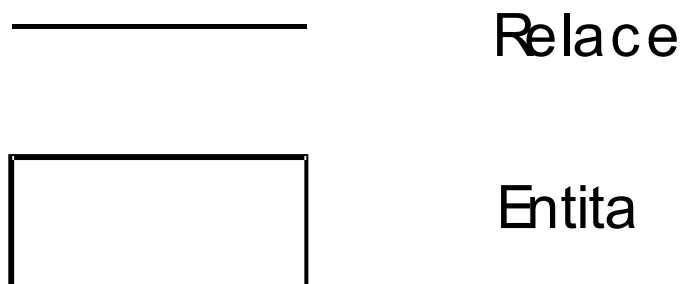
### Tvorba ERD digramu

ERD je zkratka pro Entity Relationship Diagram, tedy diagram entit a vztahů mezi nimi. V této fázi zpracováváme údaje shromážděné během konceptuální fáze. Procházíme poznámky, formuláře – z nich vybíráme podstatná jména, která jsou významná, z hlediska řešeného problému. Tato podstatná jména poslouží jako základ entit a atributů.

Jak jsme již napsali výše – entity jsou virtuální reprezentace objektů reálného světa. Ve skutečnosti je někdy z hlediska budoucí bezproblémové implementace zavádět uměle nové entity pro zjednodušení vztahů, které mezi nimi panují.

Tvorba těchto „umělých“ entit je pro začátečníka obtížná na pochopení, proto prosím věnujte zvýšenou pozornost následujícím stránkám a potom praktické ukázce vytváření tabulek a vztahů mezi nimi v rámci výkladu práce s MS Access. Pokud nepochopíte princip fungování zde představených konceptů, nemá cenu, abyste dále postupovali ve studiu této publikace – žádejte o konzultaci, popřípadě si najděte podrobnější studijní literaturu.

Nyní zpět k entitám, máme seznam podstatných jmen, z něj vyloučíme synonyma tak, aby každý objekt nebo atribut se v seznamu objevil pouze jednou.



Obr. 1: Konstruktory ERD diagramu

S použitím na obr. 1 uvedených konstruktorů sestavíme ERD digram, který zachycuje existenci vztahů mezi jednotlivými entitami. Na těchto vztazích nás zajímá nejen jejich prostá existence, ale i další vlastnosti, např. mezi kolika entitami daný vztah existuje. Binární vztahy (mezi dvěma entitami) je nejčastějším případem, ale existují i tzv. ternální vazby mezi třemi entitami, ty se ale vyskytují mnohem méně.

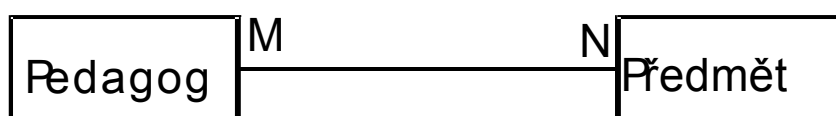
Zatímco entity jsou popisovány prostřednictvím podstatných jmen, vztahy mezi nimi popisujeme prostřednictvím sloves. ERD tak lze číst jako jednoduché věty popisující problém.

Kromě počtu entit které vztah spojuje sledujeme také tzv. kardinalitu vztahu – česky četnost vztahu. Z hlediska četnosti hodnotíme všechna zakončení vztahu. Demonstrujme si to na jednoduchém příkladu. Zkoumejme vztahy mezi entitami pedagog a předmět. Jedná se tedy o binární vztah. Při zkoumání kardinality se ptáme:

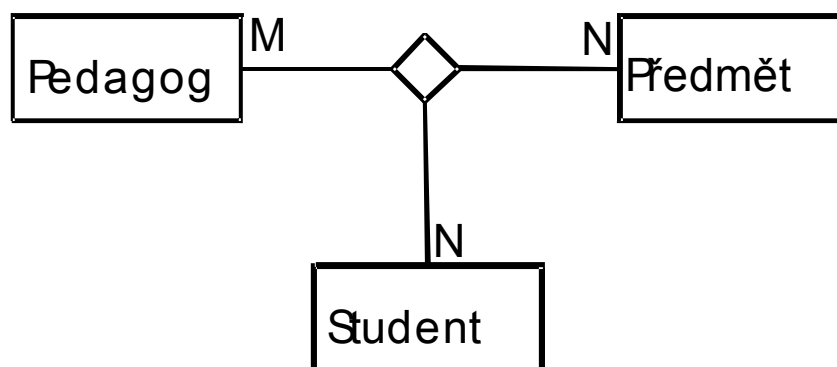
- a) kolik předmětů může učit jeden pedagog -> nevíme (N)
- b) kolik pedagogů může učit jeden předmět -> nevíme (M)

Jedná se o neurčitý vztah M:N. Přidejme do našich úvah ještě entitu student. Získáme ternální neurčitý vztah (viz. obr. 2).

### Binární vztah



### Ternální vztah



Obr. 2: Binární a ternální vztah (ERD diagram)



Ke vztahům je potřeba dodat, že ve finálním návrhu, který bude implementován v prostředí zvolené databáze, by se neměly vyskytovat neurčité a ternální vazby, ty totiž nejsou v databázím implementovatelné.

V počátečních fázích tvorby ERD tyto vazby používáme pro formalizaci našeho myšlenkového modelu, se kterým dále pracujeme. Před tím, než se vrhneme na řešení neurčitosti a ternálních vazeb se vrátíme k atributům, tedy specifikaci dat, která o entitě budeme shromažďovat.

Zkusme se zamyslet nad atributy entity Pedagog. Tato entita bude zcela jistě obsahovat atributy Jméno, Příjmení, katedra a takhle bychom mohli pokračovat dále. Všem zde zmíněným atributům schází jedna podstatná vlastnost – nejsou schopné jednoznačně identifikovat případ entity. Atributům, které mají tuto schopnost, říkáme kandidátské klíče. Z těchto klíčů vybereme jeden, který označíme primární klíč. Takovým číslem by mohlo být nějaké identifikační číslo z evidence zaměstnanců, rodné číslo, číslo pojištění apod.

Navrháme atributy všech tří entit:

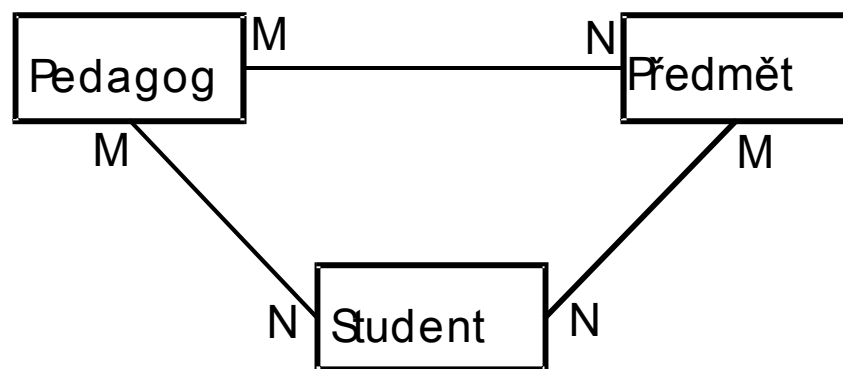
Pedagog: #IČPedagog, Jméno, Příjmení, Katedra

Student: #IČStudent, Jméno, Příjmení

Předmět: #IČPředmět, Jméno, Kredity

Pro lepší čitelnost jsou primární klíče označeny příznakem #. Pro každou entitu by měl být primární klíč stanoven. Primární klíč přitom může být jednoduchý, tedy tvořený jediným atributem, tak jak je to ukázáno v příkladech výše, nebo složený. Složený primární klíč je tvořen více atributy.

Nyní se vraťme k řešení vztahů M:N a ternálním vazbám. Začneme ternálními vazbami. Ty můžeme nahradit dvojicí binárních vztahů, které ovšem zůstávají binární. Řešení je demonstrováno na obr. 3.



Obr. 3: Řešení ternálních vztahů v rámci ERD

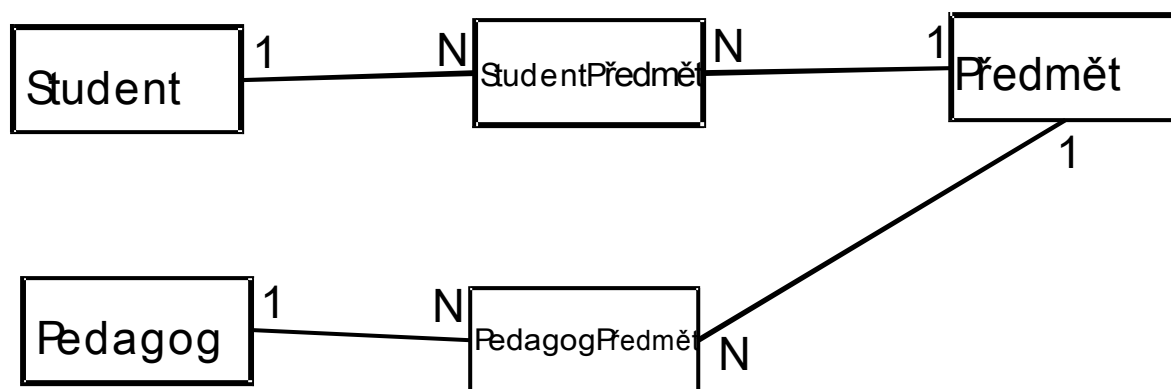
Řešením vztahů kardinality M:N je jejich nahrazení vytvořením umělých entit, které budou obsahovat jako složený primární klíč primární klíče obou původně propojovaných entit a dvojice vztahů kardinality 1:N, kterými se napojí nově vytvořená entita na entity původní.

V našem příkladě budeme potřebovat vytvořit dvě umělé entity:

StudentPředmět: #IČPředmět, #IČStudent

PedagogPředmět: #IČPedagog, #IČPředmět

Grafické řešení je zobrazeno na obr. 4.



Obr. 4: Grafické řešení vztahů kardinality M:N

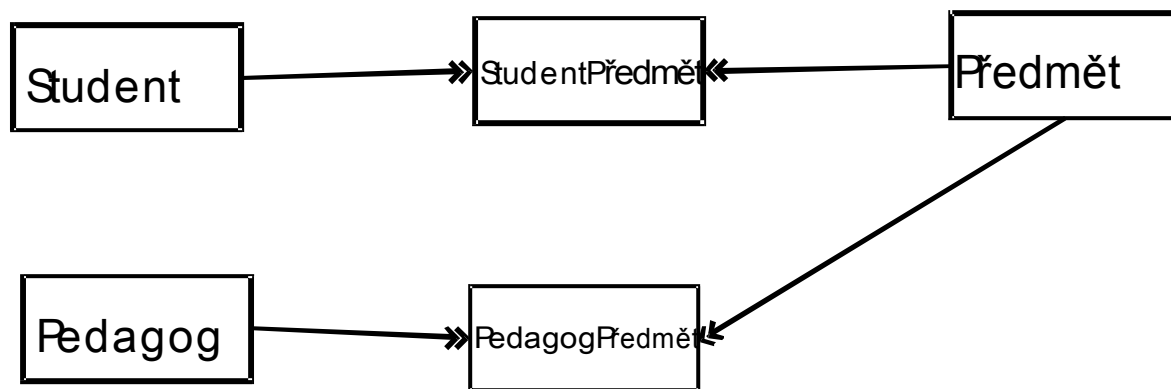
### Relační diagram

Relační diagramy jsou zaměřeny vyloženě na implementaci v ERD navrženého řešení do zvoleného databázového prostředí. Relační diagram již neumožňuje využití ternálních vazeb a i vztahy kardinality M:N by se v něm neměly vyskytovat. Na relační diagram lze tedy pohlížet jako na finální řešení ERD diagramu.

Aby bylo možné jednoduše rozlišit ERD a relační diagramy, tak se pro konstrukci relačních diagramů používá lehce změněná notace, která je však odvozená od notace použité pro tvorbu ERD diagramů.

Entity jsou tedy nahrazeny tabulkami, relace zůstávají, ale jsou značeny trochu jinak. Kardinalita není určena číselným nebo písmenným značením ale prostřednictvím šipek. Jednoduchá resp. chybějící šipka označuje 1, dvojitá šipka označuje N.

Finální návrh relačního diagramu odvozený z obr. 4 najdete na obr. 6.



Obr. 6: Finální návrh relačního diagramu

### 1.3 Pravidla návrhu tabulek

V kapitole 1.2 jsme se zabývali metodikou správného návrhu databáze jako propojení tabulek pomocí relací. V rámci výkladu jsme se zmínili o atributech entit a vysvětlili jsme si i pojem primárního klíče. V této kapitole se zaměříme na především na návrh tabulek a výklad pravidel, které s tímto problémem souvisejí.

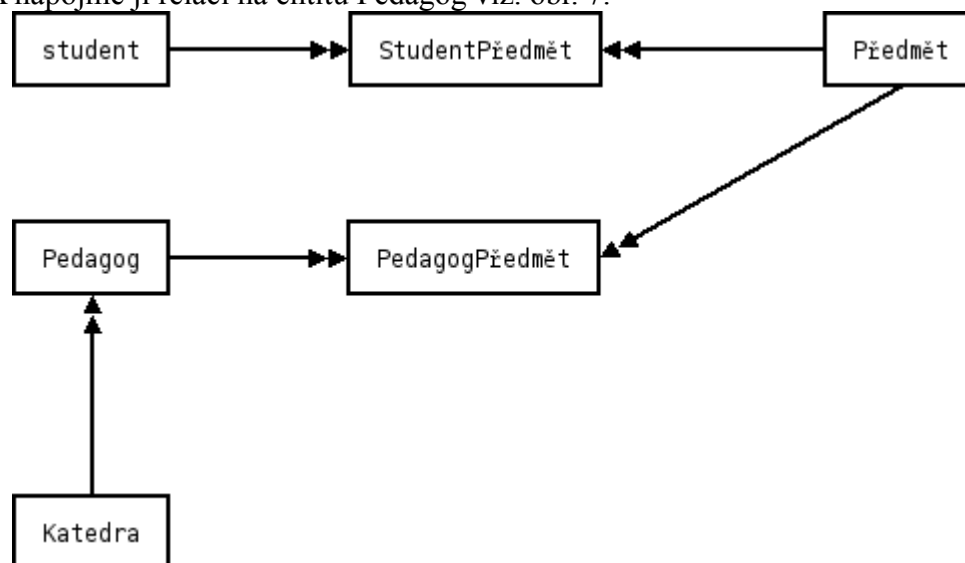
Základní funkcí databáze je aby údaje v ní obsažené byly uchovávány pokud možno pouze na jednom místě (v jedné tabulce). Prostřednictvím relací jsme schopni definovat propojení mezi tabulkami a získat tak data bez ohledu na to, kde jsou uložena.

Při propojování entit v ERD diagramu na obr. 4 byly jednotlivé entity propojeny prostřednictvím spojení primárního klíče jedné entity (Student) a části primárního klíče druhé entity (StudentPředmět). Propojované atributy však nemusí mít charakter primárního klíče na

obou stranách vazby. Rozpracujme dále uvažovaný příklad, obohatme jej o další entitu, kterou nazvěme třeba Katedra:

Katedra: #IČKatedry, Jméno

A napojme ji relací na entitu Pedagog viz. obr. 7.



Obr. 7: Zavedení entity Katedra do ERD.

Propojené atributy budou: atribut Katedra entity Pedagog a atribut IČKatedry entity Katedra. Všimněte si několika podstatných momentů tohoto spojení.

Za prvé oba atributy mají rozdílné jména. V rámci celé databáze můžeme mít tolik stejně pojmenovaných atributů kolik jen uznáme za vhodné s jedinou podmínkou, že stejně pojmenované atributy nemohou být v rámci jedné entity. Relace vytváří člověk manuálně na základě faktické totožnosti atributů nikoliv na základě stejného pojmenování.

Atribut Katedra entity přitom není součástí primárního klíče. V entitě Pedagog nám vystupuje jako tzv. cizí klíč, tedy primární klíč, popř. kandidátský klíč jiné tabulky. Fakt, že se jedná o klíč není náhodný, relace musí být schopna propojit přesně určené záznamy a to bez klíčových atributů prostě nejde.

Samotné atributy do entit nelze zařazovat zcela náhodně. Aby entita (a ve finále tabulka) byla v databázi konzistentní musí být v jedné z tzv. normálních forem. Normálních forem je celá řada. Jedená se o požadavky na vlastnosti tabulek se specifikací problémů, které mohou nastat při nesplnění těchto problémů.

Vzhledem k tomu, že normalizace bývá obvykle vysvětlována pomocí tzv. relačního kalkulu, jehož výklad přesahuje možnosti i účel této publikace omezíme se na vymezení základních problémových situací a metody jejich řešení a to do úrovně 3 normální formy, což je obvykle pro zachování konzistence databáze považováno za dostačující.

Prvním problémem, je problém opakujících se polí tabulky. Uvažujme případ, kdy požadujeme, aby v databázi byly u pedagogů vedeny telefonní čísla, na kterých jsou v případě potřeby k dosažení. Nejjednodušší implementace tohoto požadavku by byla asi následující:

Pedagog: #IČPedagog, Jméno, Příjmení, Katedra, **Tel1, Tel2, Tel3, ..., TelX**

Kde Tel1 je telefon do kanceláře, tel2 domů, k manželce, přítelkyni, k rodičům, druhé práce, mobil vlastní, mobil firemní ... Jedná se o opakující se pole. V takovýchto případech není jednoduché být jen určit kolik polí tabulky má být k tomuto účelu vyčleněno, protože se mohou objevit extrémní případy, kdy někdo nebude mít žádný telefon (poustevník) nebo bude

mít ještě víc telefonů než jsme čekali (telefonní fetišista). Dalším problémem je prohledávání takto postavené tabulky, protože musíme prohledat pole Tel1, Tel2, Tel3 až TelX, abychom zjistili, zda v něm náhodou není zavedené telefonní číslo a pokud ano musíme s ním něco udělat. **Z tohoto důvodu je nutné opakujícím se atributům vyhnout.**

Tento problém řešíme rozdělením takto postižené tabulky na dvě:

Pedagog: #IČPedagog, Jméno, Příjmení, Katedra

Telefon: #IČPedagog, #Telefon

Vznikne nám nová tabulka Telefon, do které soustředíme všechna telefonní čísla, která jsou pro daného pedagoga dostupná bez ohledu na to kolik jich je. Připomínám že tabulka Telefon má složený primární klíč, každý řádek je tedy jednoznačně identifikován prostřednictvím unikátní kombinace identifikačního čísla pedagoga a telefonního čísla.

Dalším problémem, který nám může přinést nepříjemnosti v integritě databáze je **problém tranzitivních závislostí**. Zatímco opakující se pole je názvem, který sám o sobě něco vypovídá, problém tranzitivních závislostí Vám pravděpodobně nic neřekne, takže co rozumíme pojmem tranzitivní závislost: *je to závislost, kdy jedno nebo více polí tabulky závisí na jiném poli než je celý primární klíč.*

Tato definice je z hlediska pochopení lepší než samotný název, ale pro jistotu si tranzitivní závislost rovnou demonstrujeme na nějakém příkladě. Použijme opět tabulku Pedagog, byť trochu upravené.

Pedagog: #IČPedagog, Jméno, Příjmení, Katedra, JménoKatedry

V tabulce pedagog nám oproti původní verzi přibylo pole JménoKatedry. V v čem je tedy problém. Řekli jsme si co jsou to tranzitivní závislosti, pokusme se identifikovat na čem závisí jednotlivá pole.

Tab. 1: Závislosti v tabulce Pedagog

#IČPedagog	Jméno, Příjmení, Katedra, JménoKatedry
Katedra	JménoKatedry

V čem je tedy problém? Buď máme v rámci databáze už nějakou tabulku, ve které je tento údaj veden, což je náš případ, a potom se jedná o redundantní data, která je nutné zcela zbytečně aktualizovat. V případě, že jediné místo, kde tyto údaje vedeme je právě tabulka s tranzitivní závislostí, potom ale musíme čelit opravdovým problémům z hlediska integrity databáze.

Představme si situaci, kdy v rámci organizačních změn byla zavedena nová katedra. Tuto katedru potřebujeme zavést do informačního systému, který využíváme (data z IS jsou uložena v naší databázi). Zjistíme, že zavedení nové katedry není možné dokud do ní nebude přidělen alespoň jeden pedagog. Nebo opačně, v rámci organizačních změn se pedagogové přesunují na jiné katedry a uvažovaná katedra tak fakticky zaniká převodem posledního pedagoga, ačkoliv z organizačního hlediska může ještě nějakou dobu „dožívat“.

Oba případy jsou nebezpečné a pokud to jen trochu jde je potřeba se jim vyhnout. Problém tranzitivních závislostí opět řešíme prostřednictvím rozdělení tabulky na dvě.

Pedagog: #IČPedagog, Jméno, Příjmení, Katedra

Katedra: #ČKatedry, JménoKatedry

Výše uvedená pravidla a problémy, které jejich nedodržování s sebou může nést je nutné brát v úvahu ačkoliv se nejedná o dogma, které je nutné dodržet za každou cenu. V některých případech může být výhodné jejich porušení.

## 2 MS Access

MS Access je standardní součástí MS Office ve verzi Profesional nebo vyšší. Datové formáty jednotlivých verzí spolu narozdíl od ostatních programů rodiny Office nejsou kompatibilní a je nutné je před použitím v jiné verzi Access konvertovat.

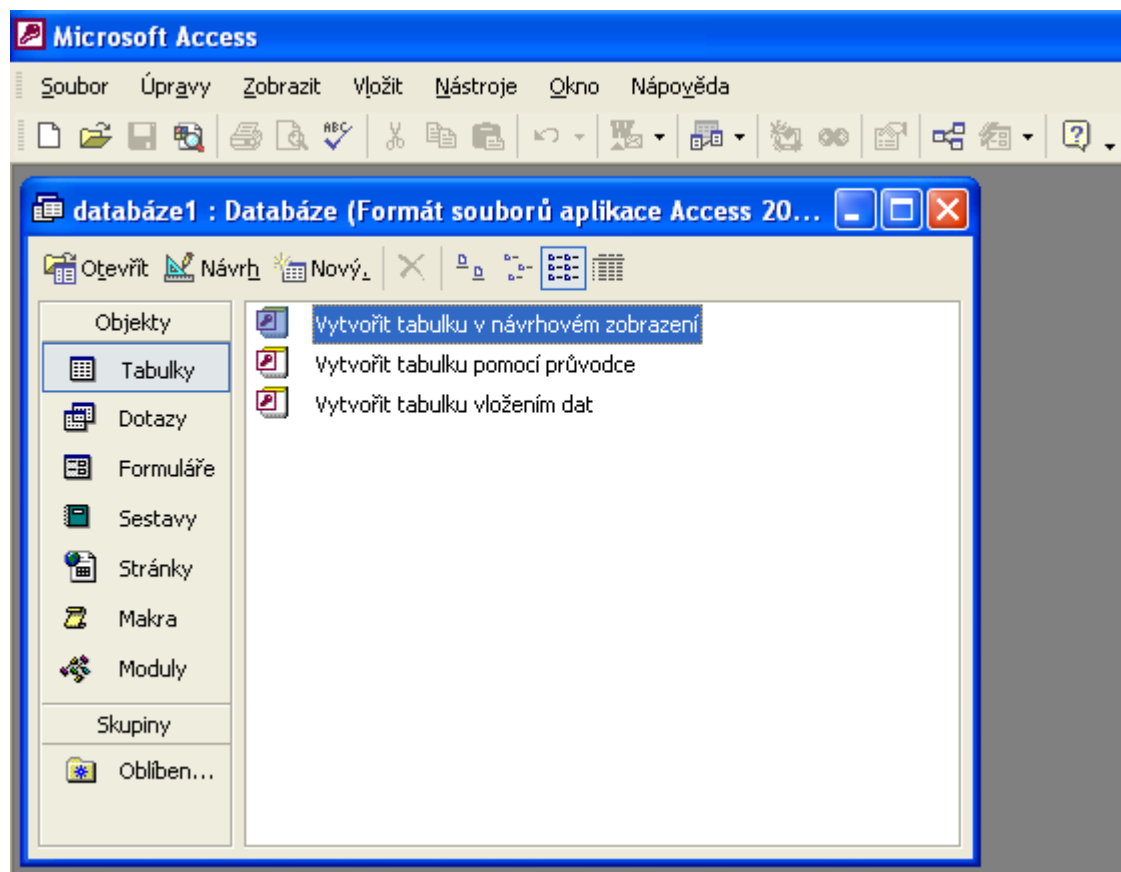
Kromě výše zmíněné distribuce MS Access je možné využít MS Access Runtime, které se dodává jako součást MS Office Developer Tools, tedy sady nástrojů pro vývoj aplikací v MS Office. Jedná se o odlehčenou verzi MS Access, která může být distribuována spolu s databázemi v MS Access vytvořenými a v tom případě nejste omezeni nutností přítomnosti MS Access na cílovém počítači.

Grafické uživatelské rozhraní je odvozeno od GUI MS Office a adaptace uživatele je tedy pohodlná a rychlá.

Spusťme tedy konečně MS Access a pusťme se do práce.

Vytvoříme novou prázdnou databázi. Databáze se implicitně ukládají do složky dokumenty uživatele přihlášeného k počítači pod názvem databázeX, kde X je pořadové číslo databáze ve složce.

Po vytvoření databáze získáme nový pracovní prostor, ve kterém můžeme vytvářet různé datové objekty, a také nástroje pro práci s nimi (viz obr. 8).















Obr. 8: Základní rozhraní prázdné databáze MS Access

Na obr. 8 máme základní menu, část jeho funkcí je analogická s funkcemi ostatních produktů rodiny MS Office, ty ostatní vysvětlíme během výkladu později. Pod menu máme panel nástrojů aplikace, ten se mění v závislosti na typu objektu, se kterým pracujeme.

Význam jednotlivých položek panelu nástrojů naleznete v tabulce 2.

Tab. 2: Význam ikon základního panelu nástrojů MS Access

	nová databáze (původní databáze bude uzavřena)
	otevřít existující databázi
	uložit databázi (nyní šedé, protože není co ukládat)
	vyhledat
	tisk a náhled, databáze v současné době neobsahuje žádný objekt, který by bylo možné tisknout
	kontrola pravopisu
	vyjmout, kopírovat, vložit
	o krok zpět
	kód, MS Script Editor – slouží pro editaci maker v jazyku Visual Basic for Applications
	vlastnosti objektu
	spustit nástroj pro návrh relací
	automaticky vytvořit formulář, sestavu ...

Uvnitř základního rozhraní MS Access vidíme dceřiné okno s námi vytvořenou databází1. To má vlastní panel nástrojů, který obsahuje nástroje pro otevření vybraného objektu, editaci vybraného objektu a vytváření nových objektů.

Objekty databáze jsou děleny a shromažďovány podle typu. Typy mohou být následující: tabulka, dotaz, formulář a sestava. Access zvládá i jiné typy objektů, jejich kompletní seznam ostatně můžete vidět na obr. 8 v okně databáze1 v levém sloupci, ale jejich výklad přesahuje možnosti této publikace.

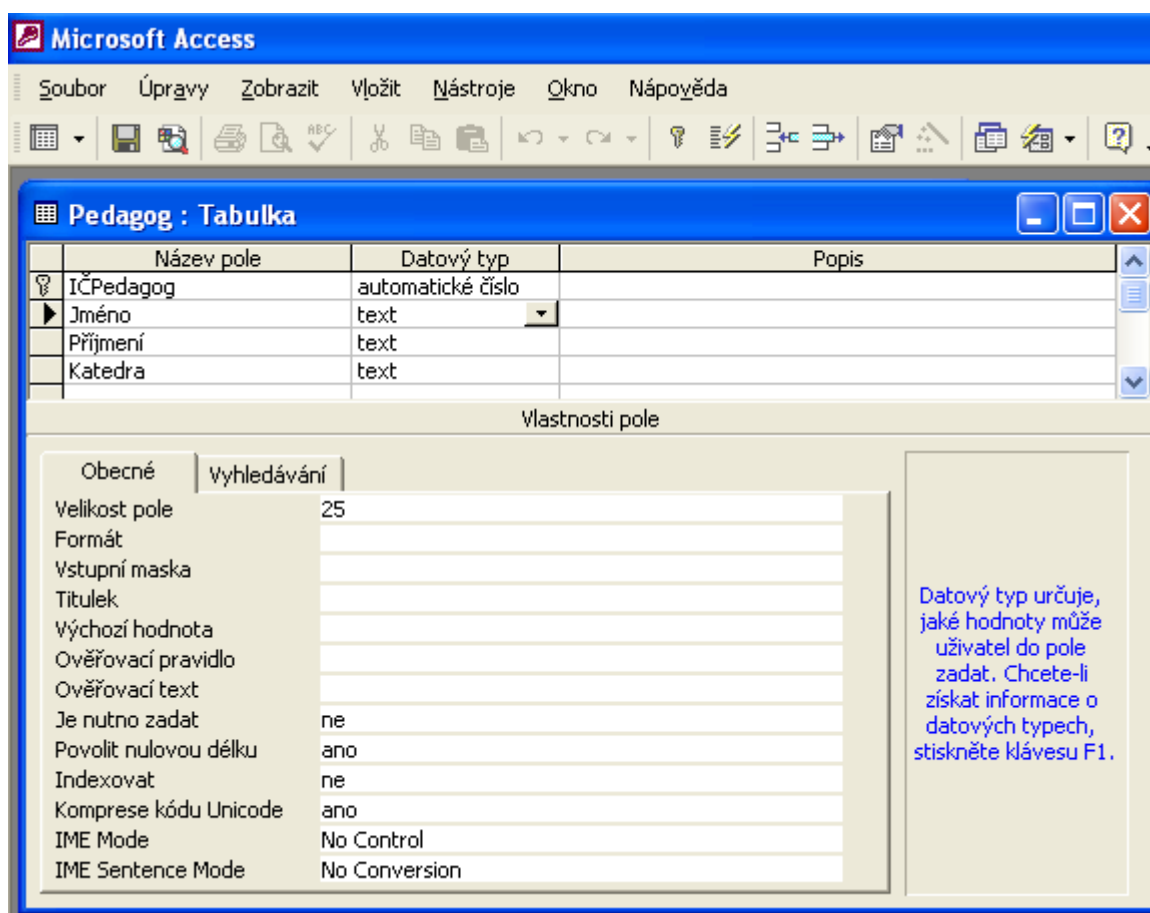
## 2.1 Tabulky

Vytváření tabulek je možné jedním ze tří způsobů: Vytvořit tabulku v návrhovém zobrazení – to je také způsob, který bych Vám doporučil, protože poskytuje maximum funkčnosti.

Druhým způsobem je pro vytvoření tabulky použít průvodce. Průvodce tvorbou tabulky je pro obvyklé účely poměrně málo využitelný. Je tomu tak pro to, že poskytuje uživateli základní typy tabulek a polí v nich obsažených, o kterých Microsoft předpokládá, že je nejspíš v databázi použijete, podle mých zkušeností tomu tak ovšem není.

Další možností je vytvořit tabulku vložením dat. V tomto případě zadáte několik řádků tabulky a na základě analýzy jejich obsahu, se nadefinuje struktura databáze. Vzhledem k automatickému pojmenování polí a neschopnosti v tomto režimu nadefinovat primární klíč, je tento způsob použitelný na opravdu jednoduché tabulky, nebo předpokládá, že je dodefinujete v návrhovém zobrazení.





Vraťme se tedy k návrhovému zobrazení. Základní rozhraní s předvyplněnou tabulkou Pedagog je zobrazena na obr. 9.



Obr. 9: Tabulka Pedagog v návrhovém zobrazení

Hlavní panel nástrojů se změnil. Význam nových ikon je patrný z tab. 3.

Tab. 3: Význam ikon hlavního panelu nástrojů – návrh tabulky

 Návrhové zobrazení	(popis se týká pouze „rozbalené“ ikony úplně vlevo) jedná se o sadu nástrojů, které slouží pro přepínání režimů práce tabulky, zejména mezi režimem návrhu, ve kterém měníme strukturu tabulky a režimem zobrazení datového listu, ve kterém vyplňujeme údaje.
	primární klíč – tento nástroj se použije pro označení primárního klíče
	indexy – slouží pro definování indexů. Indexy zrychlují vyhledávání
	vložit řádky, odstranit řádky v definici polí

Pro definici tabulek Microsoft připravil samostatné okno. V jeho horní části definujeme jednotlivá pole tabulky a jejich datový typ, ve spodní části je možné doplnit některá „jemnější“ nastavení datových polí. Definice jména a datového typu je povinná, vyplnění položky popis naopak povinné není, slouží pro zlepšení orientace v datové bázi zejména pokud se k její definici uživatel vrátí po delší době nebo její správu po někom převzal.

Pro pojmenovávání položek by mělo platit, že jména budou dokumentovat obsah této položky, tedy co bude uživatel posléze do této položky vyplňovat. Pro pojmenovávání přitom

můžeme použít i interpunkci, háčky a čárky Accessu jako takovému nevádí. Pokud však očekáváte, že v budoucnu bude databáze převedena do nějakého „většího“ databázového systému, které již problémy s interpunkcí ve jménech položek mít mohou, v tom případě je potřeba: *psat jména položek bez háčku a čárky*.

Pro uchování dat máme k dispozici celou řadu různých datových formátů. Při vytváření nového datového pole se implicitně navolí datový typ text o maximální délce 50 znaků. Do tohoto typu pole můžeme vkládat alfanumerické znaky do maximální délky 255 znaků, maximální délku je přitom možné zmenšit a zefektivnit tak práci databáze.

Pokud 255 znaků není dostatečný počet, máme k dispozici datový typ memo, který umožňuje vložit až 65 535 znaků, což je pro většinu aplikací dostatečné množství. Kromě množství dat, která se do pole typu memo vejdu je hlavním rozdílem oproti textovému poli nemožnost indexovat pole memo. V případě, že budeme v tomto poli vyhledávat tak toto vyhledávání se může pěkně protáhnout, z tohoto důvodu memo používáme pouze tam kde je to nutné, jinak preferujeme textová pole.

Pro zadávání číselných dat máme k dispozici datový typ číslo. V rámci tohoto datového typu v položce velikost pole můžeme nadefinovat požadovanou velikost pole od bajtu až po desetinné číslo s dvojitou přesností.

Datové typy datum a čas a měna slouží k zadávání data a času a měny. Oproti číslům resp. textu umožňují definovat formát který pro tuto položku vyžadujeme.

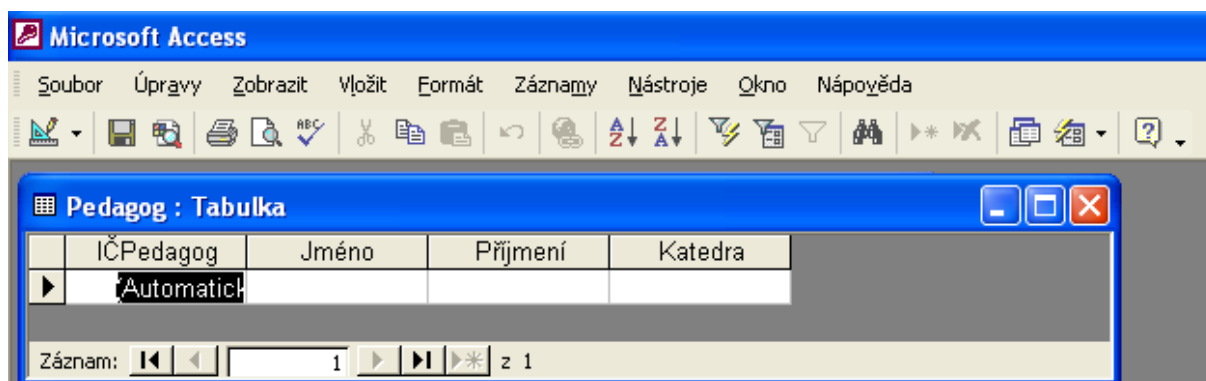
Zajímavým typem je automatické číslo. Jedná se o speciální datový typ, který nevyplňuje uživatel ale počítač. Slouží k vygenerování unikátního čísla v rámci dané položky tabulky, z tohoto důvodu je často využíván pro tvorbu privátního klíče. Při propojování tabulek pomocí relací je nutné mít na paměti datový typ – ten musí být stejný na obou koncích relace, to ovšem neznamená, že na obou koncích relace může být automatické číslo! Dvě položky automatické číslo znamenají že propojené položky mezi sebou nikdy nenajdou souvislost, nebo ji najdou ale bude chybná. Automatické číslo můžeme použít v tabulce, ve které má relace kardinalitu 1. V tabulce druhé dáváme do souvislosti číslo o stejné délce (implicitně dlouhé celé číslo).

Ano/Ne datový typ odpovídá číslu o délce 1 bit – tedy logická hodnota TRUE-FALSE.

Objekt OLE umožňuje vkládat libovolné objekty (obvykle přes schránku). Maximální délka je 1GB. Jedná se o jediný datový typ, který umožňuje uchovávat obrázky.

Vyplňme nyní položky tak, aby odpovídaly entitě Pedagog, se kterou jsme pracovali v kapitole 1. Jedinou věc, kterou jsme si zatím neukázali je tvorba privátního klíče. Privátní klíč se tvoří tak, že označíme řádek nebo řádky, které představují všechny pole primárního klíče (postupujeme, jako bychom označovali řádky v Excel) a klikneme na ikonu klíče, čímž Accessu řekneme, že označeným položkám se má přiřadit statut primárního klíče.

Uložíme a přepneme se do režimu zadávání dat viz obr. 10.



Obr. 10: Zadávání dat do tabulky



Zadávání dat se podobá práci s MS Excel samozřejmě s výjimkou nemožnosti zadávat vzorce a ovlivňovat formát. Pro editaci údajů je třeba zmínit několik věcí. Ukládání změn na řádku se provádí buď automaticky po přechodu na jiný řádek, nebo ručně kliknutím na ikonu diskety, popřípadě přes menu.

Při výmazech se řádky označí jako smazané, nedojde ale k jejich fyzickému výmazu. Přesto obnovení dat je prostřednictvím jiné funkce než zpět v MS Access nemožné, takže dávejte pozor jaké změny provádíte, Vaše práce by mohla přijít na zmar.

Specifické nástroje pro zadávání dat do tabulky jsou vysvětleny v tab. 4.

Tab. 4: Panel nástrojů – zadávání dat do tabulky

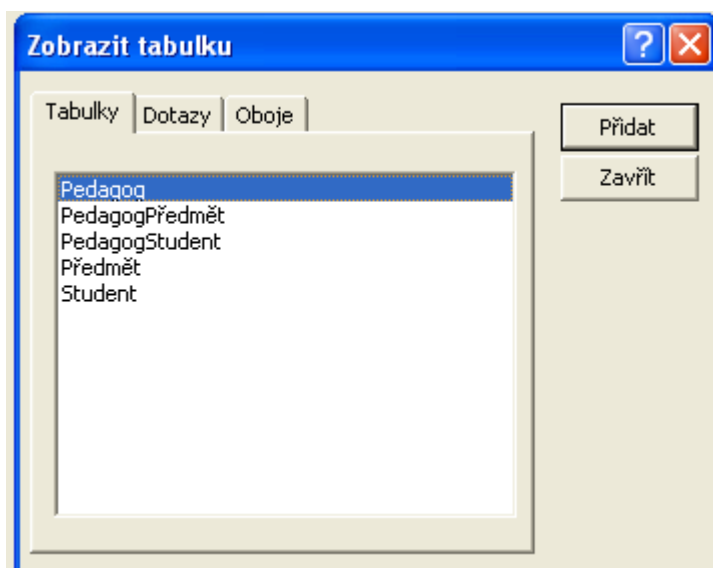
	seřadit vzestupně nebo sestupně podle označeného sloupce tabulky
	filtrovat data podle různých kritérií
	vyhledat
	nový záznam, zrušit záznam

Opuštěme nyní tuto tabulku a doplníme ostatní tabulky, tak abychom se dostali do stavu podobnému obr. 4 (samozřejmě bez definovaných relací).

## 2.2 Relace

Máme nadefinovány tabulky, nyní se pokusíme definovat relace. Před započítím práce se ujistěte, že máte zavřené tabulky v návrhovém režimu – tento režim práce je výhradní a neumožní vytvořit relace mezi takto otevřenými tabulkami. V průběhu práce se Vám může stát, že se relaci o požadované kardinalitě nepodaří vytvořit. Nejčastější příčinou bývá nesoulad datových typů polí propojovaných relací. V případě, že si nebudete vědět totálně rady, zkuste prozkoumat referenční databázi, která je k této publikaci příkládána – obsahuje kompletně vyřešený příklad, který v této publikaci společně řešíme.

Do okna pro zavádění relací se dostaneme z hlavního okna databáze (toho ze kterého můžeme otevírat různé tabulky, formuláře ...) kliknutím na tlačítko relace (viz. tab. 2). Zobrazí se dialogové okno pro specifikaci tabulek, které chceme pomocí relací propojit, viz. obr. 11.



Obr. 11: Výběr tabulek pro zavádění relací




Dialogové okno by mělo obsahovat všechny tabulky, které jsme v databázi vytvořili. Vybírat je můžeme po jednom a nebo najednou. Vybrané tabulky zavedeme do relačního diagramu pomocí tlačítka přidat.

Jednu tabulku je možné do diagramu přidat libovolně-krát. To je výhodné zejména u databází, které mají velké množství tabulek a relace by se táhly přes celou obrazovku, což by snížilo čitelnost celého digramu. V tomto případě je efektivní přidat kopii tabulky někde v blízkosti cílové tabulky a nadefinovat relaci v ní.

Všech pět tabulek rozmístíme v digramu dle našeho gusta a s ohledem na čitelnost výsledného diagramu. Přesun a změna velikosti se provádí jako u standardních oken Windows – chytne za titulek a táhnu, chytne za okraj a změním velikost. V případě, že v diagramu přebývají nějaké tabulky, nechťenou tabulku prostě označím a stisknu klávesu delete.

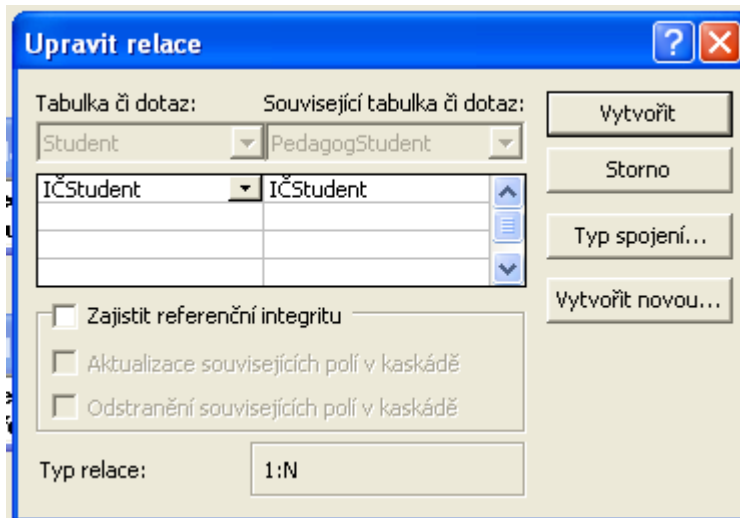
Podívejme se na panel nástrojů a možnosti, které nám poskytuje (viz. tab. 5).

Tab. 5: Panel nástrojů relace

	přidat tabulky do diagramu
	nástroje pro filtrování relací
	odstranit tabulku z diagramu

Všimněte si že primární klíče v tabulkách jsou znázorněny tučně. Nyní jak vytvořit relaci – postup je prostý – chytete pole ve zdrojové tabulce a táhnete na pole v cílové tabulce. Poté co pustíte levé tlačítko myši se objeví dialogové okno pro definici vlastností relace (viz. obr. 12).

Zkusme postupovat společně. Uchopte pole IČStudent z tabulky Student a přetáhněte jej na pole IČStudent z tabulky PedagogStudent.



Obr. 12: Definice vlastností relace

V případě že uchopíme špatné pole nebo ho upustíme na špatném poli můžeme v tomto dialogovém okně z tabulky vybrat pole správné. Všimněte si typu relace 1:N (na obr. 12 dole). Typ relace je rozpoznáván automaticky. Pokud tedy typ relace 1:N nemáte – někde ve struktuře databáze jste udělali chybu. Zkuste ji najít a odstranit dle pokynů na začátku této kapitoly.

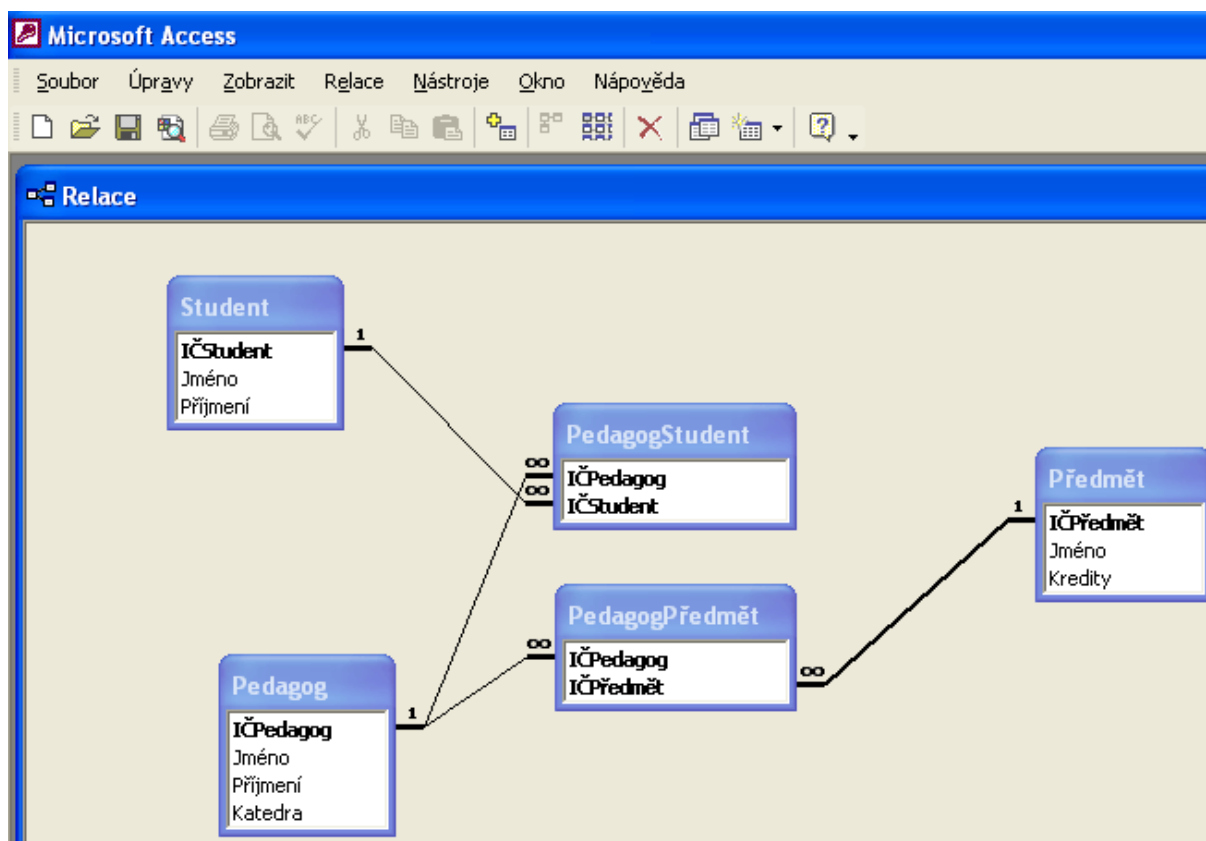
V dialogovém okně máme také zaškrťovací pole Zajistit referenční integritu, které implicitně není zaškrtnuto. Co tato volba přesně znamená. V úvodní kapitole jsme sestavovali ERD model, také jsme zmínili pravidla pro správnou tvorbu databází – z nichž se jedno bych

rád v tomto kontextu zopakoval – údaje jsou vedeny pouze na jednom místě a k ostatním údajům jsou vztaženy prostřednictvím relací.

Zajištění referenční integrity se týká případů, kdy manipulujeme s primárním klíčem nebo záznamem jako celkem. Uvažujme příklad: mějme databázi podobnou té naší. V tabulce Student ale pro tento případ uvažujme jako primární klíč rodné číslo. Při zadávání údajů se studijní referentka spletla a napsala jiné rodné číslo. Po dvou letech studia si při rutinní kontrole referentka chyby všimla a opravila ji. Rodné číslo studenta se ale také objevuje v tabulce StudentPředmět, která obsahuje všechny zapsané předměty studenta. Přepsáním rodného čísla v tabulce student se najednou studentovi „ztratily“ všechny jeho studijní výsledky. A ještě hůř, objevil se další student (nový), který čistě náhodou měl stejné rodné číslo jako to původní chybné prvního studenta, hned po zápisu se ukázalo že řadu předmětů již „absolvoval“.

Aby bylo možné se těmto chybám vyvarovat je nutné aktualizovat související pole v dalších tabulkách. MS Access tyto úkoly zvládá automaticky, pokud se mu řekneme, aby to automaticky dělal, tedy zaškrtneme políčko zajistit referenční integritu a potom aktualizace souvisejících polí v kaskádě a odstranění souvisejících polí v kaskádě.

Až budeme spokojeni s nastavením, klikneme na tlačítko vytvořit. Podobným způsobem nadefinujeme i ostatní relace. Výsledek by měl odpovídat obrázku 13.



Obr. 13: Relační diagram databáze

Malé upozornění na závěr této kapitoly. Vytvořením relací jsme trochu omezili své možnosti úpravy datové báze, tedy struktury tabulek, především předefinování primárních a cizích klíčů. Pokud zjistíte potřebu tyto údaje měnit, je nutné nejprve odstranit všechny relace z tabulky, kterou hodláte editovat (její strukturu), provést požadované změny a potom znovu nadefinovat relace.

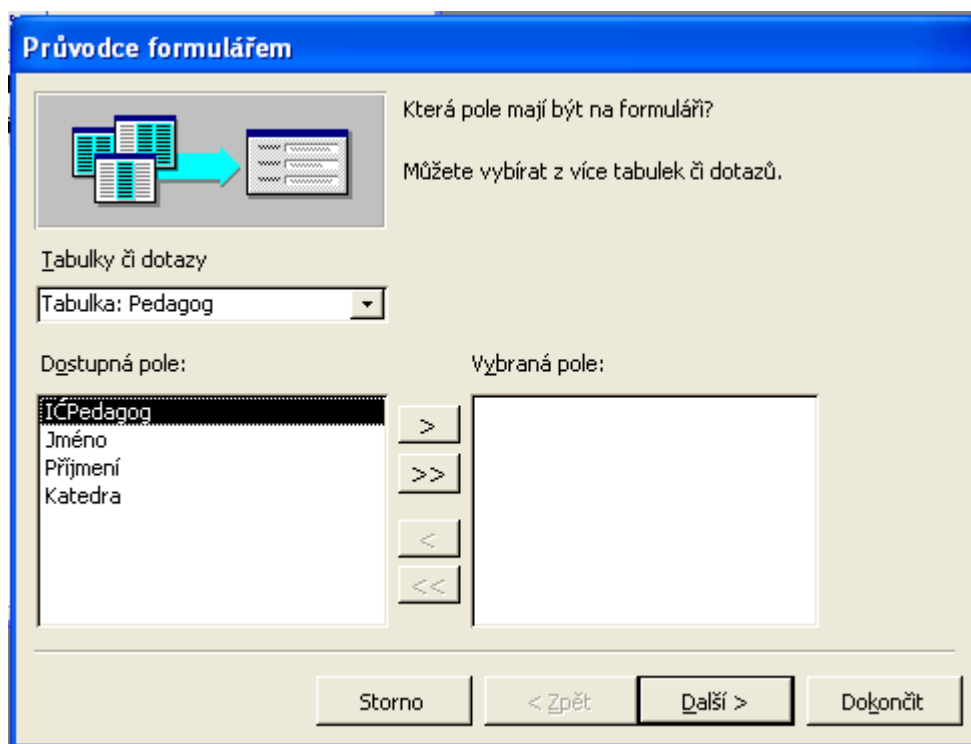
## 2.3 Formuláře

Formuláře slouží pro vytváření jednoduchého grafického uživatelského rozhraní pro editaci dat. Pro demonstraci možností formulářů budeme potřebovat mít něco vyplněného v tabulkách Pedagog, Předmět a Student. Pokuste se doplnit alespoň dva nebo tři záznamy do těchto tří tabulek. K naplnění použijte režim zobrazení datového listu. Data jako takové mohou být libovolná.

Formulář můžeme navrhovat buď na „zelené louce“, tedy v režimu návrhového zobrazení, nebo pomocí průvodce. Průvodce je na rozdíl od tvorby tabulky použitelnější a lze jej s úspěchem použít jako základ nad kterým doděláte specifickou funkčnost vyžadovanou Vaší databází.

Začneme s vytvářením formulářů prostřednictvím průvodce.

Na první obrazovce průvodce definujeme dotazy nebo tabulky na základě kterých hodláme zkonstruovat formulář. Nic jiného než tabulky v naší databázi nemáme nadefinováno, takže nic jiného než tabulky zatím nemůžeme použít. V rámci této obrazovky definujeme jednotlivé položky, které chceme zobrazit na formuláři.

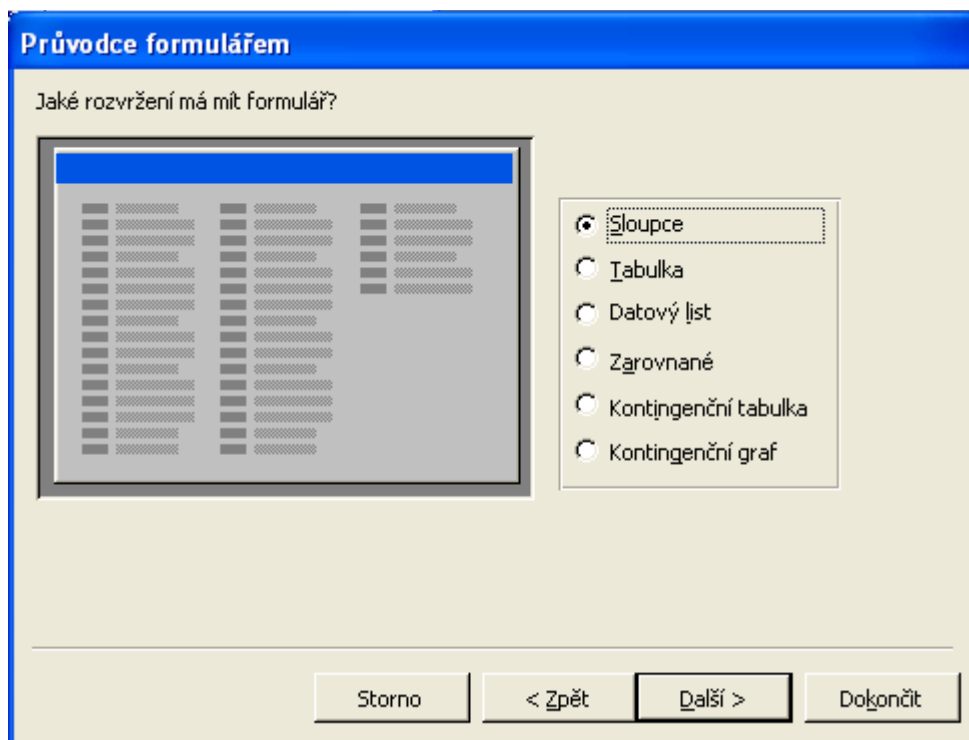


Obr. 14: Průvodce tvorbou formuláře – volba polí

Pole vybíráme tak, že nejprve zvolíme tabulku, ze které hodláme použít položky a tyto položky přesuneme tlačítkem > (po jednom) >> (všechny) z dostupných polí do vybraných polí.

V případě, že je nutné použít obsah více tabulek na formuláři, budeme pokračovat výběrem další tabulky a přidáním jejích polí do vybraných polí. Teprve až jsme spokojeni s výběrem polí pro formulář klikneme na další.

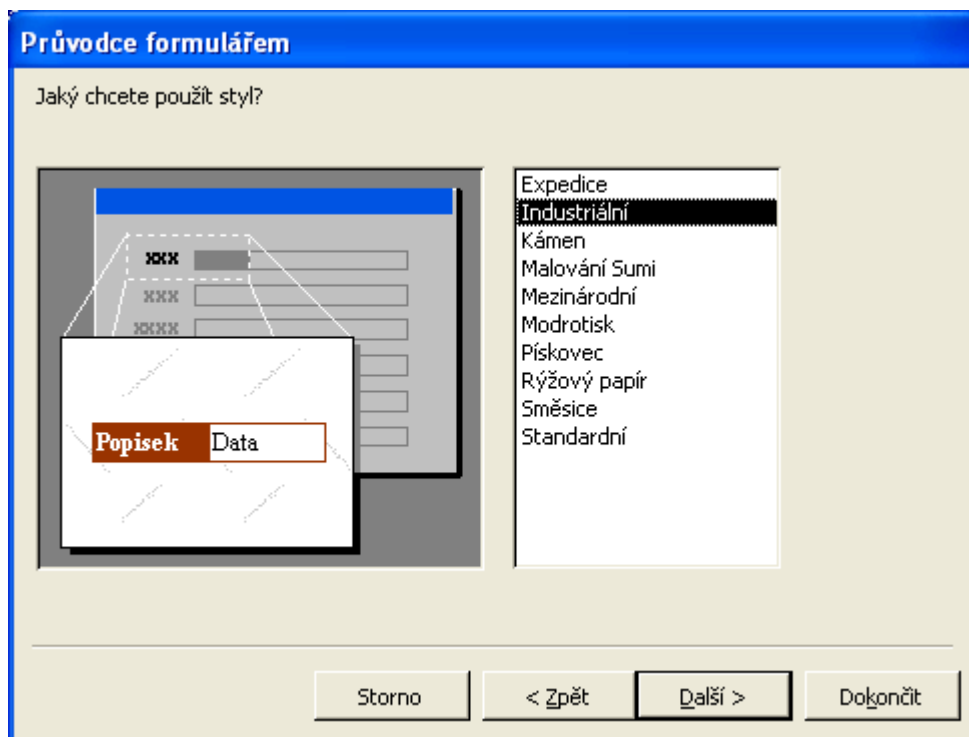
Naším cílem bude vytvoření formuláře pedagoga, vybereme tedy všechna pole tabulky pedagog a klikneme na další.



Obr. 15: Výběr rozvržení formuláře

Pokud vynecháme výklad kontingenční tabulky a grafu, které mají specifické použití zbývají nám na výběr čtyři typy rozvržení. Ty můžeme rozdělit do dvou skupin podle toho, kolik záznamů zobrazí na jedné obrazovce. Sloupce a zarovnané zobrazí pouze 1 záznam, zatímco tabulka a datový list jich zobrazí více.

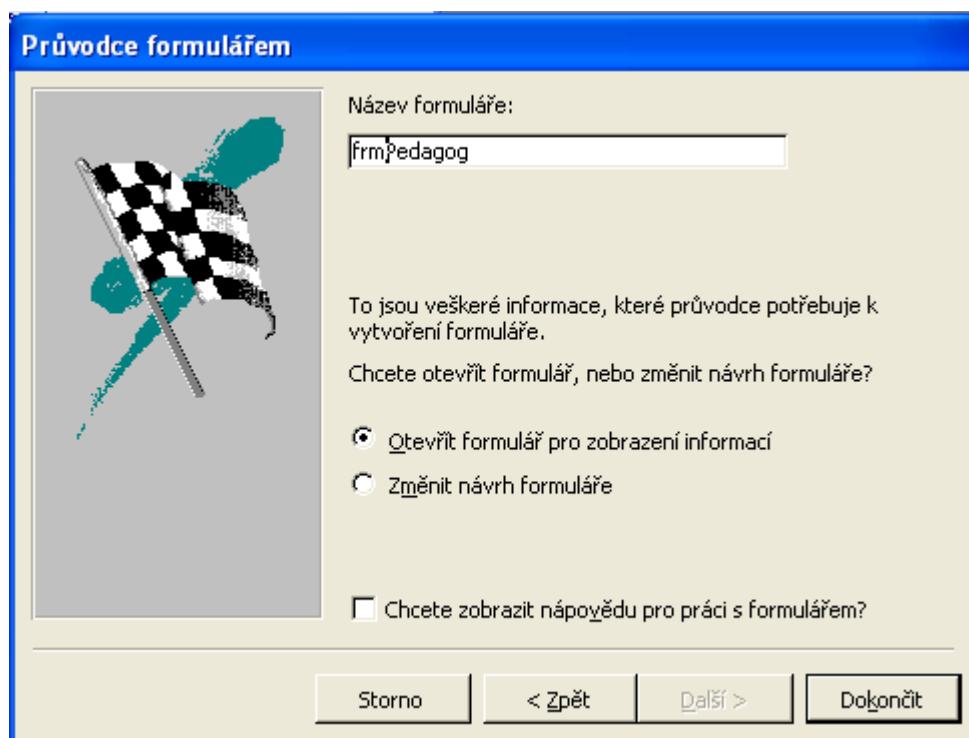
Pro naši aplikaci bude stačit na obrazovce pouze jeden záznam, vyberme tedy třeba zarovnané a klikněme na další.



Obr. 16: Definice vizuálního stylu formuláře

V dalším kroku definujeme vizuální styl formuláře. Access má vizuálních stylů zabudovaných poměrně dost. Při volbě stylu bychom měli být konzistentní – tedy použít pokud možno jeden styl pro všechny formuláře, které budeme vytvářet.

Vyberte styl, který je Vám blízký a klikněte na další.



Obr. 17: Pojmenování formuláře

Čeká nás poslední krok tvorby formuláře – jeho pojmenování a rozhodnutí, jak zobrazit výsledek našeho úsilí. Implicitně se formulář pojmenovává stejně jako tabulka nebo dotaz, na základě kterého byl vytvořen. V našem případě tedy Pedagog. Při větším množství objektů a práci s nimi prostřednictvím maker může dojít ke zmatení, pokud objekty různých druhů zůstanou pojmenovány stejně.

Abychom odlišili jednotlivé typy objektů můžeme použít systém prefixů, které nám přímo v názvu určí o jaký typ objektu se jedná. Já jsem v příkladu použil prefix frm pro formulář, tedy výsledek frmPedagog. Systém prefixů MS Access nevyžaduje a jména na přítomnost prefixů ani nijak nekontroluje, jedná se pouze o jednoduchý způsob, jak zvýšit čitelnost databáze jako celku.

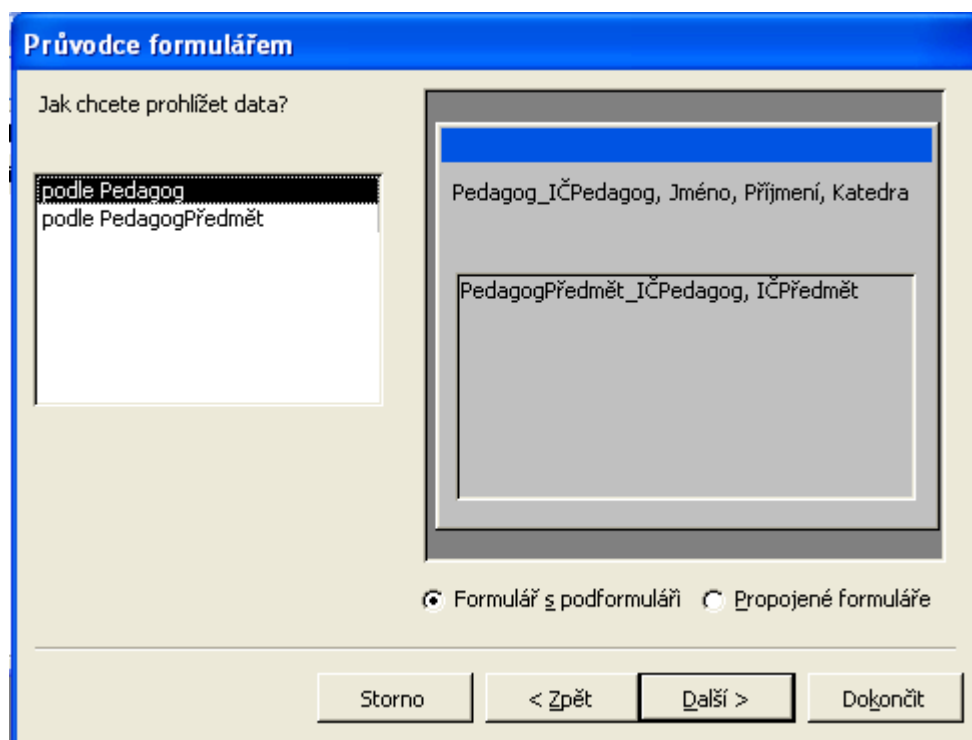
Výsledný formulář můžeme zobrazit buď v režimu návrhu nebo režimu pro zobrazení informací, podle toho, jestli chceme okamžitě provádět změny vzhledu nebo ne. My si necháme formulář zobrazit v režimu pro zobrazení informací. Naš výsledek by měl být podobný obr. 18.



Obr. 18: Výsledek tvorby formuláře prostřednictvím průvodce

Až se dostatečně pokocháte svým výtvoře uzavřete formulář a zkusíme něco trochu složitějšího, pokusíme se vytvořit formulář pedagogů, kterým bychom chtěli v rámci formuláře přiřazovat předměty, které učí.

Spustíme znovu průvodce, znovu vybereme všechny pole tabulky Pedagog navíc však vybereme i všechny pole tabulky PedagogPředmět. Všimněte si, že průvodce tvorbou formuláře se oproti předchozímu příkladu trochu změnil. Nyní máme k dispozici dialog pro volbu podle čeho hodláme prohlížet data.



Obr. 19: Definice podle čeho prohlížet data

Data bychom měli prohlížet podle tabulky, která má na svém konci relace kardinalitu 1. V našem příkladě tedy tabulka Pedagog. Z náhledu je patrné, že pedagog se na výsledném formuláři objeví jednou a jeho předměty se objeví v podformuláři. Klikneme na další.

Tentokrát nemáme možnost volit rozvržení hlavního formuláře a podformuláře, ale pouze podformuláře, omezil se nám také výběr na datový list a tabulku, podformulář totiž z logiky věci by na jednom formuláři měl zobrazit více záznamů. Zvolíme tedy rozvržení tabulka (na ni se na rozdíl od datového listu vztahuje vizuální styl) a klikneme na další.







Po definici stylu pojmenujeme formulář i podformulář a zobrazíme výsledek v režimu pro zobrazování informací. Výsledek by měl odpovídat obr. 20.

Obr. 20: Formulář s podformulářem vytvořené prostřednictvím průvodce

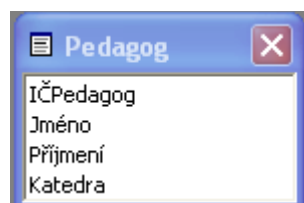
Výsledný formulář má jednu zásadní nevýhodu – vyžaduje od uživatelů, aby si zapamatovali identifikační číslo předmětu což má k optimálnímu řešení dost daleko, je tedy jasné, že se bez úprav formuláře tentokrát neobejdeme.

Otevřeme si tedy formulář v režimu pro návrh a nejprve se podíváme na možnosti, které nám vývojové prostředí přináší. Panel nástrojů obsahuje několik ikon, se kterými jsme se zatím nesečkali, jejich popis naleznete v tabulce 6.

Tab. 6: Panel nástrojů návrhového režimu formulářů

	seznam polí
	sada nástrojů
	automatický formát, slouží ke změně vizuálního stylu formuláře nebo prvku formuláře, podle toho co je vybráno
	kód, otevře vývojové prostředí pro návrh maker
	vlastnosti
	sestavit makro, výraz, kód

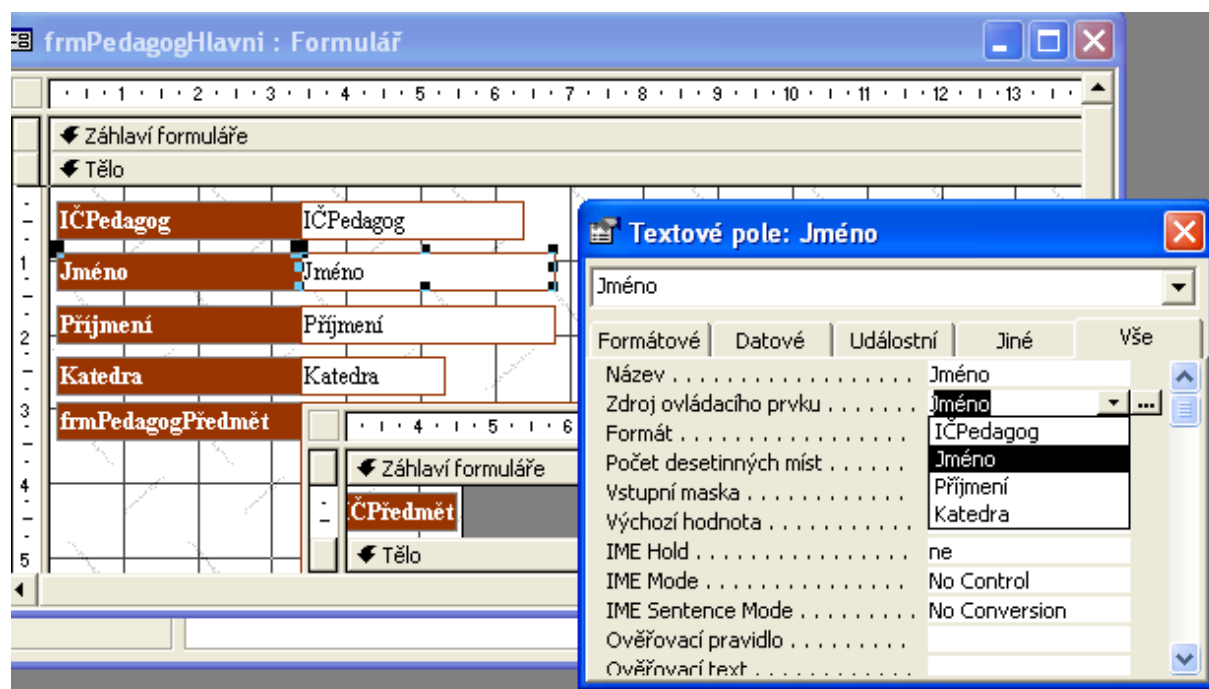
Prostřednictvím ikony seznamu polí se zobrazí/skryje seznam dostupných polí aktivního formuláře. Všimněte si jak se seznam polí mění při označování prvků hlavního formuláře a podformuláře. Prostřednictvím seznamu polí můžeme jednoduše vytvořit nový prvek na formuláři a to prostě tak, že uchopíme pole, které na formuláři hodláme vytvořit a přetáhneme je na formulář na místo, kde ho hodláme vytvořit.



Obr. 21: Seznam polí



Podíváme se jakým způsobem jsou jednotlivé prvky formuláře navázány na tabulky, tedy jak Access ví, které údaje má kam ukládat. Klikněme na prvek formuláře jméno a zobrazme si jeho vlastnosti.








Obr. 22: Vlastnosti textového pole jméno

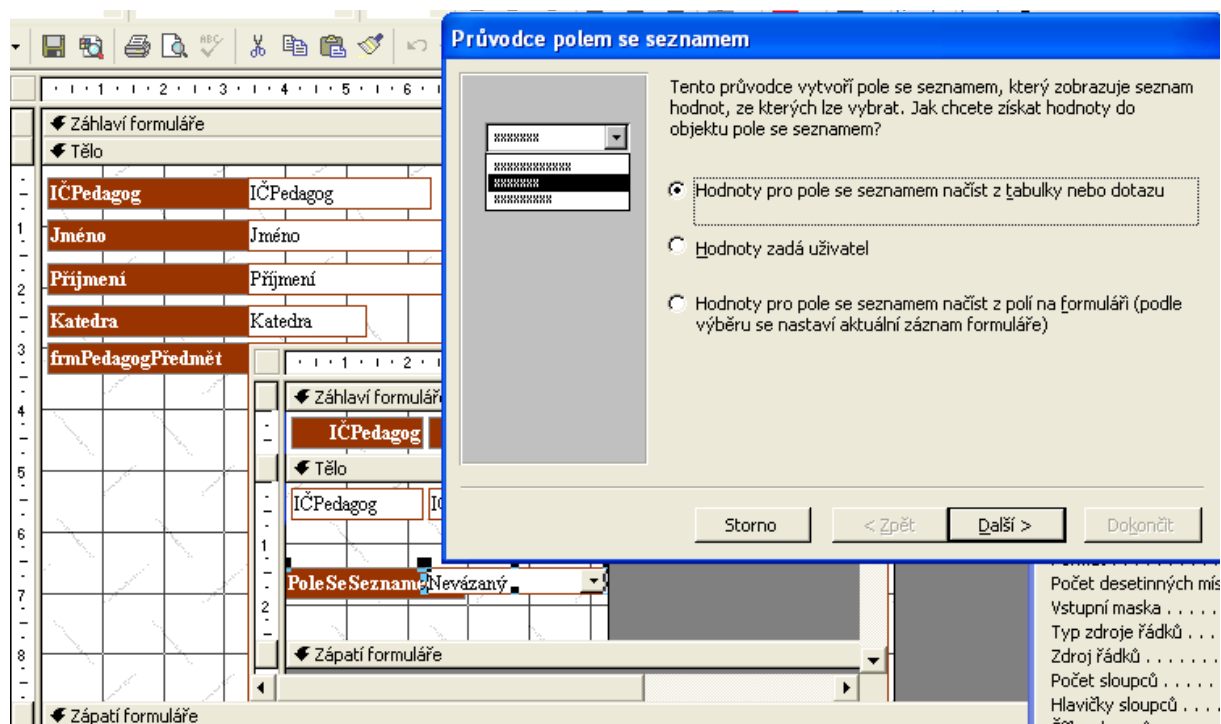
Ve vlastnostech na kartě datové nebo vše máme možnost nastavit zdroj ovládacího prvku, tedy to kýžené navázání prvku na pole tabulky. Prostřednictvím tohoto nastavení máme možnost svázat s polem tabulky i prvky, které byly vytvořeny čistě v návrhovém režimu.

Nyní se vrhneme na tvorbu pole se seznamem – tento prvek nalezneme v sadě nástrojů. Některé prvky sady nástrojů jsou popsány v tabulce 7.

Tab. 7: Panel nástrojů

	vybrat objekty
	průvodci ovládacími prvky, při „zaškrtnutí“ tohoto nástroje se po vytvoření některých prvků na formuláři dojde ke spuštění průvodce nastavením vlastností tohoto prvku (ujistěte se, že tento nástroj je zaškrtnutý)
	popisek. Jedná se o statický text, který obvykle nebývá navázán na tabulky a zůstává tak v rámci formuláře neměnný
	textové pole, ve skutečnosti jde o dvojici prvků textové pole a k němu příslušející popisek, tento prvek je nutné svázat s položkou tabulky
	pole se seznamem. Spojuje funkčnost textového pole seznamu, umožňuje tak uživateli jak vyplnit hodnotu, nebo ji vybrat ze seznamu

Nejprve si vytvoříme prostor pro práci. Ten vytvoříme tak, že uchopíme horní okraj zápatí formuláře (šedý pruh s nápisem zápatí formuláře) a táhneme směrem dolů. Potom zvolíme na panelu nástrojů pole se seznamem a klikneme do uvolněného prostoru. Pozor pracujeme v podformuláři! Pokud vytvoříte pole se seznamem na hlavním formuláři nebudete jej schopni navázat na požadované pole tabulky!

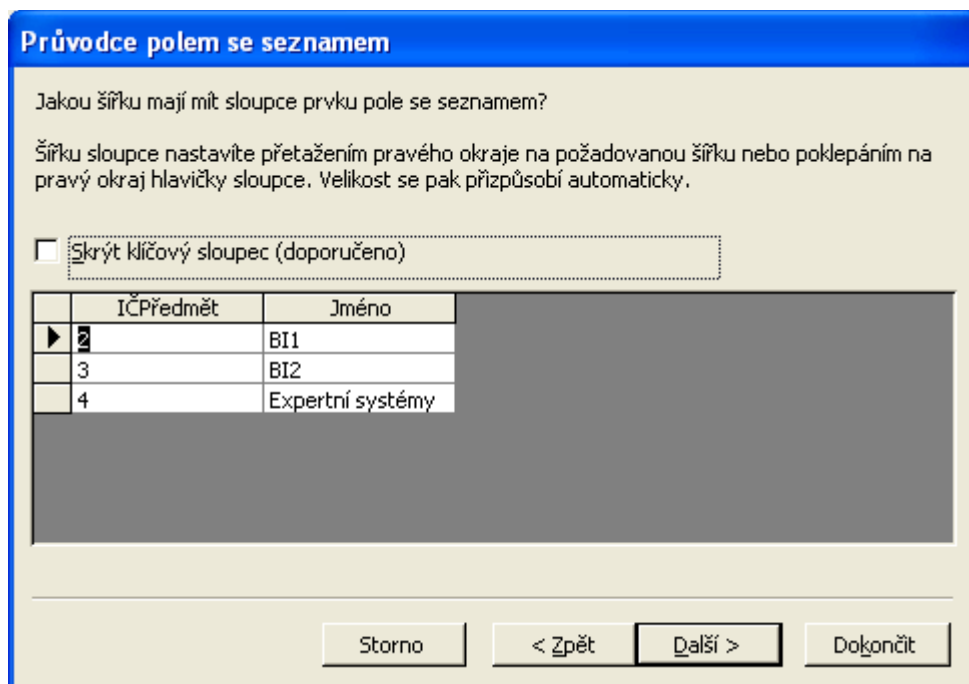


Obr. 23: Průvodce pole se seznamem

Máme tři možnosti jak pole se seznamem naplnit údaji, ze kterých má mít uživatel možnost výběru. Hodnoty můžeme načíst z existující tabulky – to je náš případ. Máme tabulku předmět, která obsahuje údaje – jméno předmětu, ze kterého chceme vybírat. Další výhodou této volby je aktuálnost, v případě aktualizace tabulky Předmět bude automaticky aktualizováno i pole se seznamem (při dalším načtení formuláře).

Druhou možností je, že hodnoty zadá uživatel manuálně a nakonec máme možnost načíst hodnoty z existujících polí formuláře. Z hlediska flexibility bych doporučoval řešit napojením na tabulku.

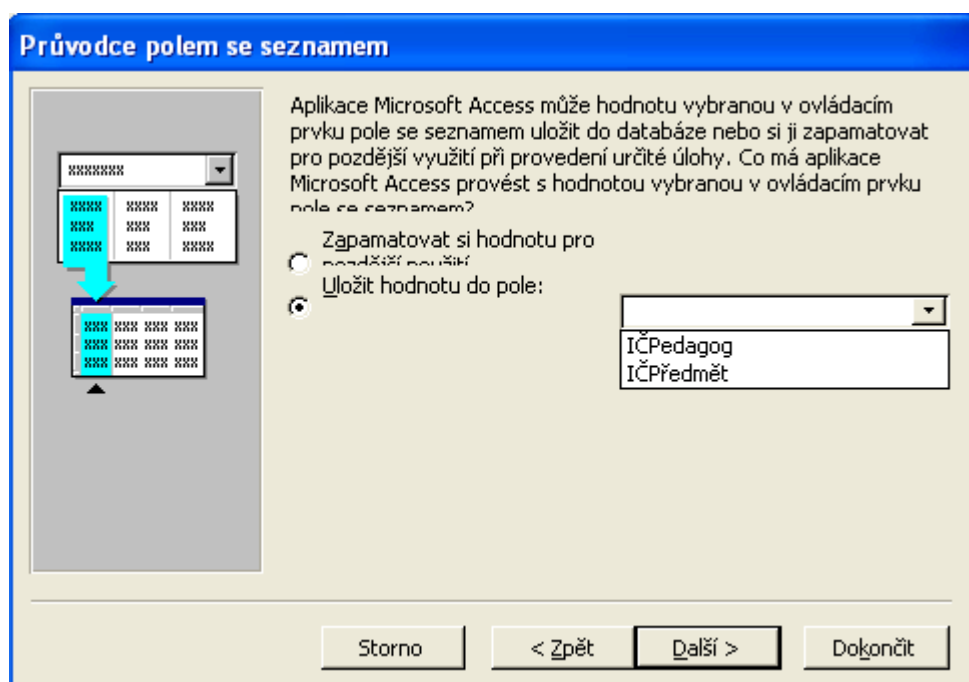
V dalším kroku vybereme tabulku Předmět a klikneme na další. Objeví se naše staré známé okno pro výběr polí, která chceme použít. Logicky musíme vybrat hodnotu, která je obsažena i v tabulce PedagogPředmět, jinak by tabulku Předmět nebylo jak zavést do formuláře. Dále vybereme Jméno, to je totiž ta položka, na základě které se bude uživatel rozhodovat. Položku kredity ponechám na zvážení, můžeme ale nemusíme ji tam zavést.



Obr. 24: Skrýt či neskrýt klíčový sloupec

Skrýt či neskrýt klíčový sloupec, toť otázka o kterou v tomto kroku průvodce běží. Co to pro nás znamená. Ve vybraných polích se nachází IČPředmět, jedná se o primární klíč tabulky předmět. Toto pole má charakter automatického čísla, obsah tohoto pole tedy nemá nějaký praktický význam, který mohl přispět k rozhodování uživatele o výběru. Můžeme tedy zaškrtnout pole skrýt klíčový sloupec.

Po zaškrtnutí se při výběru nebude zobrazovat IČPředmět, ačkoliv bude nadále v rámci pole se seznamem vedeno – musí být, bez něj nejsme schopni napojit tento prvek do formuláře.



Obr. 25: Co udělat s vyplněnou hodnotou

Vyplněnou hodnotu pole se seznamem je možné si zapamatovat pro pozdější užití. Tím pozdějším užitím se ovšem myslí navázání logiky zpracovávající tento údaj prostřednictvím nějakého makra navázaného na nějaký ovládací prvek. Alternativou této možnosti je uložit hodnotu do některého z polí. V našem případě zvolíme druhou možnost a napojíme pole se seznamem na pole IČPředmět.

Dokončíme průvodce a v návrhovém režimu pak pouze doopravíme vzhled, tedy především odstraníme nyní již nadbytečné textové pole IČPředmět, to je nahrazeno nově vytvořeným polem se seznamem. Dále bychom mohli zmenšit pracovní plochu tak aby se nám vzhled podformuláře začal líbit. Výsledek naší práce by se měl blížit výsledku zobrazenému na obr. 26.

Obr. 26: Upravený podformulář s polem se seznamem

Zkuste si trochu pohrát s návrhem formuláře, změnit vzhled, experimentovat s vytvářením prvků, které jsme zde neprobali.

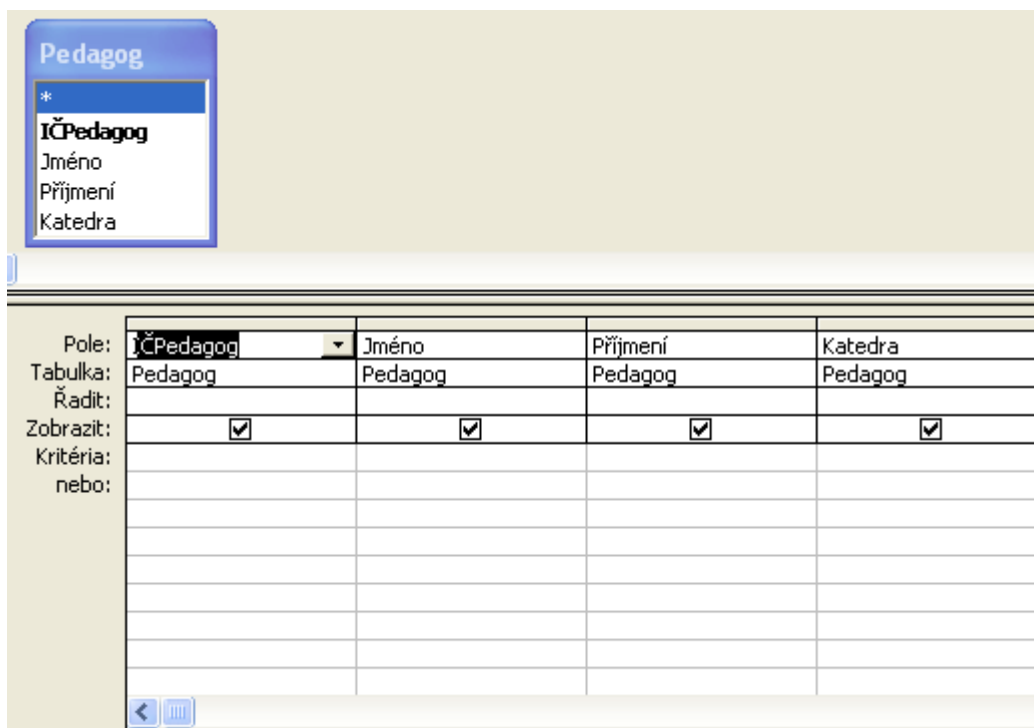
## 2.4 Dotazy

Předtím než se pustíme do vytváření dotazů bychom si mohli v krátkosti říci, co to dotazy jsou a k čemu je možné je využít. Dotazy, jak jejich název napovídá, umožňují vytvořit výběr dat z databáze i s možností nadefinovat si omezující podmínky.

Dotazy používáme k získání dat, která pak můžeme použít v nějaké další aplikaci (MS Excel apod.), popřípadě můžeme na základě dotazu vytvořit formulář nebo sestavu.

Dotazy vytváříme opět podobně jako formuláře. Návrhový režim je však podstatně jiný. Zkusíme vytvořit dva dotazy, které budou odvozeny od pohledu na data, který jsme realizovali prostřednictvím formulářů.

V prvním dotazu provedeme výběr z tabulky Pedagog. Průvodce je velmi podobný průvodci tvorbou formuláře, takže jeho správné nastavení by Vám nemělo činit potíže. Jako prefix dotazů obvykle volím písmeno q (z angl. query). Návrhové zobrazení a režim pro zobrazování dat jsou znázorněny na obrázcích 27 a 28.



Obr. 27: Dotaz návrhové zobrazení

	IČPedagog	Jméno	Příjmení	Katedra
▶		1 Pavel	Šenovský	030
		2 Oldřich	Kodym	545
		3 Petr	Neznámý	123
*	omatické číslo)			

Obr. 28: Dotaz – režim zobrazování údajů

Horní část okna návrhového režimu funguje podobným způsobem jako okno návrhu relací a to včetně ikon pro doplňování tabulek. Nejprve musíme mít vybrány všechny tabulky, které chceme v dotazu použít. Jednotlivá pole tabulky, která chceme v rámci dotazu získat pak přetahujeme z tabulek do spodní části obrazovky.

V dolní polovině okna máme seznam vybraných polí s určením tabulky, ze které pocházejí. Pro každé pole je možné nastavit řazení záznamů a to podle abecedy vzestupně nebo sestupně popřípadě neřadit vůbec (předvoleno). Pokud řadíme podle více položek, vyšší prioritu řazení mají položky vlevo.

Tedy pokud nechám řadit vzestupně pole Jméno a Příjmení, záznamy se seřadí nejprve podle jména a tam kde jméno bude stejné se provede řazení podle příjmení.

Pořadí jednotlivých polí je možné měnit prostě tak, že označíte celý sloupec pole a přetáhnete jej směrem doprava nebo doleva, jak si situace vyžaduje.

Zaškrťovací pole zobrazit ovlivňuje zda se dané pole objeví ve výsledku dotazu. Mohou existovat taková pole, která jsou důležitá pro výběr záznamů, ale nepožadujeme jejich přítomnost ve výsledku.

Kritéria slouží pro definici omezení, podle kterých se vyberou záznamy z propojených tabulek. Formování kritérií závisí na typu pole, ke kterému formulujeme podmínku.

#### Textová pole

- Like „Š\*“ vybere všechny záznamy, které v daném poli budou začínat na Š
- Like „\*š\*“ vybere všechny záznamy, které v daném poli budou mít kdekoli š
- Like „\*š“ analogicky vybere všechny záznamy, kde š bude na konci

Not Like ... vybere takové záznamy, které nesplňují danou podmínku (opak výše uvedených příkladů)

Všimněte si prosím, že na velikosti písma v dotazu nezáleží.

Číselná pole

> 1, < 50, = 15 tedy použití klasických podmínek

jedinou záležitostí je operátor nerovnosti, za který Access považuje <>, tedy <>2 vybere záznamy větší nebo menší čísla 2.

Podmínky, které mají platit zároveň píšeme do jednoho řádku a pro jejich spojení používáme operátor and. <>2 můžeme tedy přesat jako >2 and <2. Podobně je možné postupovat i u manipulaci s podmínkami pro textová pole.

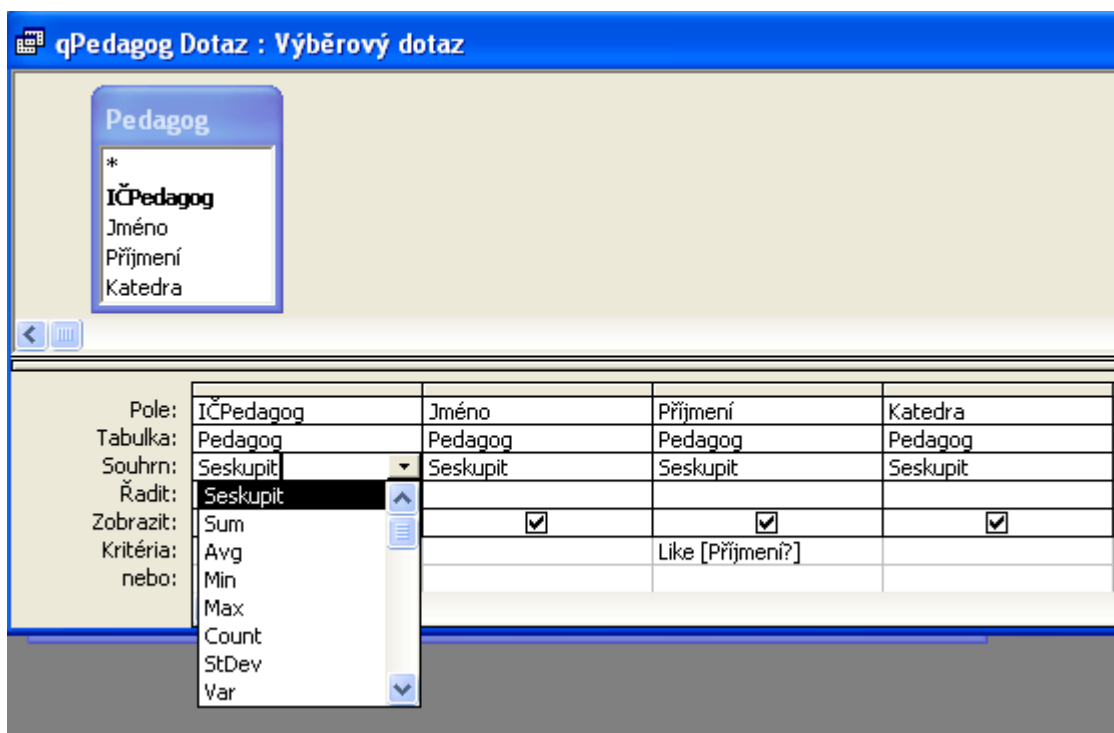
Dotazy, které jsme zde popsali se souhrnně nazývají výběrové. Výběrové dotazy však nejsou jediné, které můžeme prostřednictvím MS Access vytvářet. Dalším typem jsou tzv. parametrické dotazy. Parametrické dotazy jsou speciální třídou výběrových dotazů, u kterých si nejsme při vytváření jisti podmínkou.

Například již předem můžeme vědět, že budeme chtít vybírat pedagogy podle příjmení, ale podmínku budeme chtít stanovit až při spuštění dotazu. V tom případě formulujeme podmínku jinak:

Like [Zadejte příjmení:]

V hranatých závorkách se nachází text výzvy, která se zobrazí při spuštění dotazu.

Dalším typem dotazu jsou tzv. sumární dotazy. Výsledkem sumárních dotazů je vždy pouze jeden řádek. Nežjišťujeme totiž záznamy ale charakteristiky záznamů jako je minimum, maximum, suma, standardní odchylka apod. To že chceme vytvářet sumární dotaz sdělíme Accessu tak, že klikneme na ikonu sumy v panelu nástrojů. V okně návrhového režimu získáme k dispozici další řádek nazvaný souhrn viz. obr. 29.



Obr. 29: Sumární dotaz

## 2.5 Sestavy

Sestavy jsou vytvářeny obvykle na základě dotazů, ačkoliv je možné je sestavit i nad tabulkami. Proces tvorby a manipulace s prvky sestavy se velmi podobá tvorbě formuláře. Účel sestavy je ale jiný, jedná se o přípravu dat pro vytištění nebo převod do jiného programu pro další zpracování (MS Word).

V rámci sestav uživatel nemá možnost editovat data. Veškerá data se pouze „nalijí“ do předem připravené šablony. Při pojmenovávání sestav můžeme použít prefix rpt (z ang. report).

V současné době byste již měli mít dostatek znalostí, abyste s pomocí průvodce vytvořili sestavu sami. Výsledek by mohl být třeba takovýto:

### rptPedagog

<i>Pedagog_IČPedagog</i>	1			
<i>Pedagog_Jméno</i>	Pavel			
<i>Příjmení</i>	Šenovský			
<i>Katedra</i>	030			
<i>Předmět</i>	<i>Jméno předmět</i>	<i>IČPedagog předmět</i>	<i>IČPředmět</i>	<i>Kredity</i>
BI1		1	2	2 2
BI2		1	3	3 3
Expertní systémy		1	4	4 3

Obr. 30: Sestava z tabulek Pedagog, PedagogPředmět a Předmět

Výslednou sestavu by samozřejmě bylo možné dále upravovat.

## 3 Open Office Base

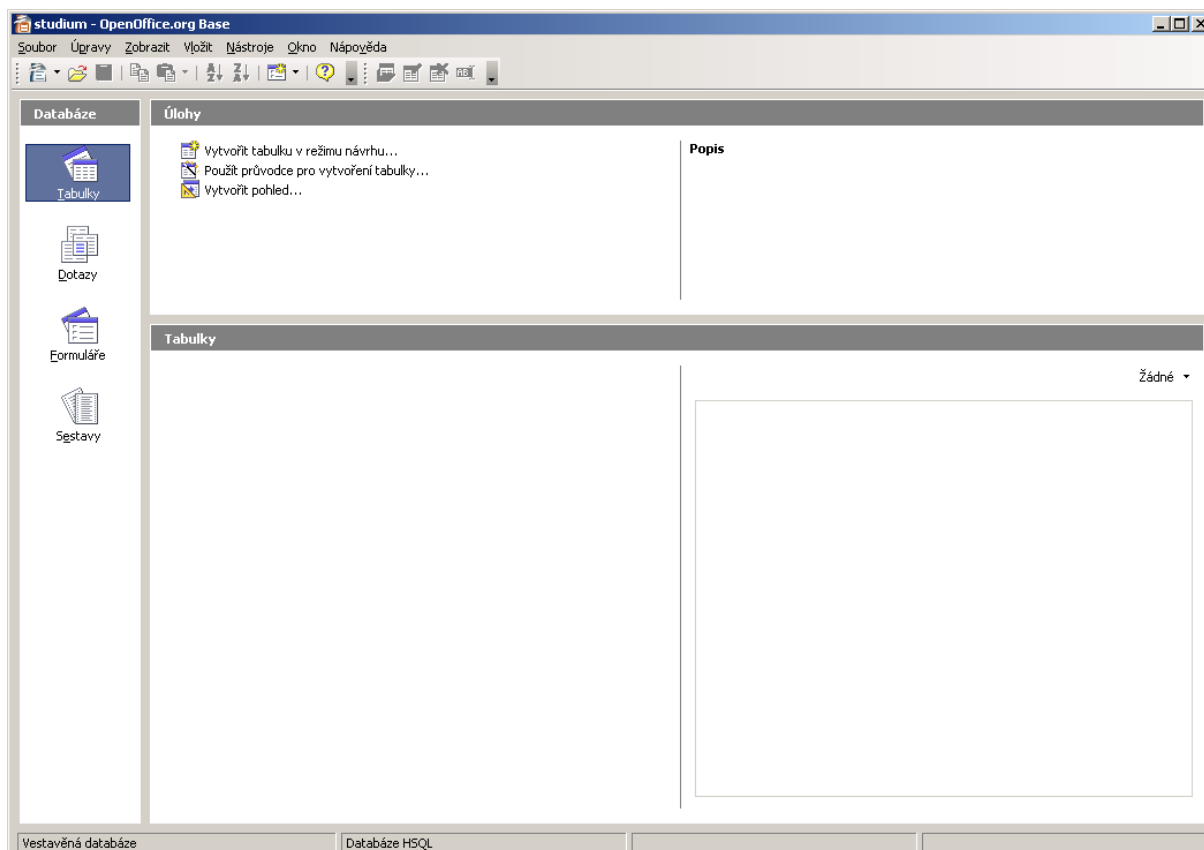
Open Office Base (OO Base) je součástí balíku Open Office od verze 2.0. Tato databáze, stejně jako celý balík kancelářských aplikací Open Office je zajímavý, že je volně dostupný k bezplatnému použití [1]. Svoji koncepcí se inspiroje pravděpodobně nejvíce z MS Access, nicméně v oblasti funkčnosti za tímto databázovým prostředkem výrazně zaostává.

Pro základní seznámení s databázemi však tento produkt plně vyhovuje. Do budoucna se dá také předpokládat vývoj v oblasti dostupných funkcí. Dost už obecného pokecu o databázi, zkusme vytvořit nějakou databázi prakticky.

První věc, kterou musíme udělat je vymezit prostor (soubor), do kterého budeme databázi ukládat. Podobně jako u MS Access se všechny objekty (tabulky, formuláře, apod.) ukládají do jednoho souboru.

Základní rozhraní databáze je zobrazeno na obr 31. Pro plnou práci s databází musí také být nainstalován Java Runtime Environment [2]. V dalším textu se zaměřím při popisu práce zejména na vysvětlení rozdílů od MS Access. Abyste se tedy „chytali“ je nutné si

předem alespoň pročíst kapitolu dvě, která je MS Access věnována. Nebojte se také vrátit k této kapitole i když Vám nebude něco jasného.



Obr. 31: Základní rozhraní Open Office Base

V databázi pracujeme se čtyřmi typy objektů:

- tabulkami
- formuláři
- dotazy
- sestavami

K těmto objektům (kromě tabulek) je možno přidávat novou funkčnost pomocí vestavěného skriptovacího jazyka Python nebo Basic.

Oproti MS Access se také poměrně výrazněji liší filosofie práce s jednotlivými objekty. Zatímco v MS Access je možné jednoduše se přepínat mezi režimy návrhu a režimem editace dat, OO Base striktně rozlišuje mezi oběma režimy a uživatel musí prostřednictvím volby při spuštění daného objektu deklarovat v kterém režimu chce pracovat. Pro přepnutí se mezi režimy musí potom objekt uzavřít a opětovně ho otevřít v požadovaném režimu.

Tato nepříjemná vlastnost systému je způsobena tím, že formuláře a sestavy nejsou řešeny v samotné databázi, ale jsou realizovány v Open Office Writer (textový editor).

### 3.1 Tabulky

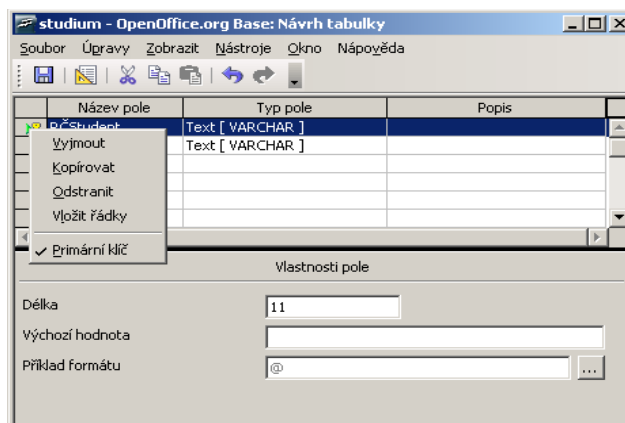
Jsou realizovány podobným způsobem jako v MS Access. Jediným podstatným rozdílem je realizace datových typů, které jsou odvozeny z SQL standardu a jsou tedy do určité míry bohatší než implementace v MS Access. Z různých datových typů lze vypíchnout např. typ image nebo binary, které umožňují pohodlné ukládání binárních dat.



U rozdílů bych zde také rád zmínil rozdílnou implementaci automatického čísla. OO Base automatické číslo nepovažuje za samostatný datový typ, ale za běžný celočíselný typ (integer), který má vlastnost automatická hodnota nastaven na ANO.

Bohužel v současné době pokulhávají formuláře a sestavy, ve kterých bychom mohli chtít tyto datové formáty využít.

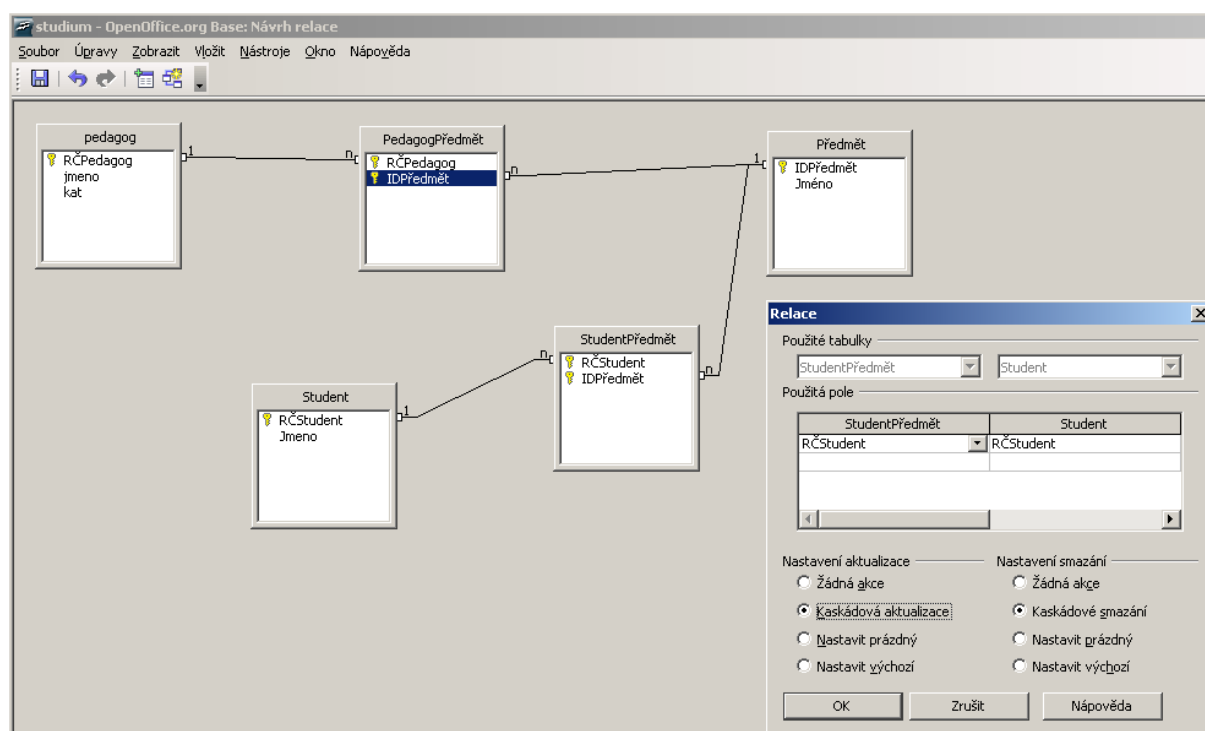
Rozdílná je taktéž práce s primárními klíči. Primární klíč do tabulky přidáme z kontextového menu označených polí tabulky (viz. Obr. 32). Při změně primárního klíče dbejte na to, aby jste nejprve odebrali stávající primární klíč a teprve potom definovali nový primární klíč. Současná verze totiž v některých případech prosté redefinice primárního klíče vyhazovala chyby.



Obr. 32: Definice tabulky Student

### 3.2 Relace

Opět, práce je velmi podobná MS Access. OO Base umožňuje editovat relace prostřednictvím volby menu Nástroje -> Vztahy ... (viz. Obr. 33).



Obr. 33: Definice relací mezi tabulkami

OO Base pro záznam relace používá jinou notaci 1:n. Taková notace, je však třeba podotknout, je bližší „informaticky korektní“ implementaci ERD. Z hlediska grafického ztvárnění také OO Base nedělá rozdíly mezi relací normální a relací, pro kterou je definována kaskadní aktualizace nebo výmaz.

Definice relace spočívá v nastavení souvisejících polí ve spojovaných tabulkách. Na rozdíl od MS Access však až na výjimky OO Base sama tyto pole nedefinuje – je nutné definici provést ručně.

### 3.3 Formuláře

Implementace formulářů v OO Base se výrazně liší od implementace formulářů v MS Access, protože OO Base k jejich implementaci využívá služeb OO Writer, využívá se přitom schopnosti OO Writer pracovat prostřednictvím jednotného rozhraní s různými datovými zdroji. OO Base je pak pouze jedním z těchto zdrojů.

Předně, v OO Base je možné navázat na jeden formulář pouze maximálně dvě tabulky. Výsledkem práce je vždy formulář (nikoliv formulář a podformulář jako u MS Access). OO Base to zajišťuje možností implementace gridů pro zobrazování dat z připojených tabulek.

Formulář jako takový je vytvářen v osmi krocích

1. výběr polí (hlavní tabulka) umožňuje uživateli vybrat pole z hlavní tabulky
2. nastavit podformulář – nastavíme podřízenou tabulku máme-li takovou. Tabulku, na základě budeme „podformulář“ vytvářet přitom můžeme vybrat podle relace, která na něj vede z tabulky hlavní (tady se vyplácí pečlivý design báze dat)
3. přidat pole podformuláře – nastaví pole podformuláře z podřízené tabulky pokud byla vybrána
4. zobrazit spojená pole – umožní uživateli nastavit podle kterých položek je možné tabulky spojit. Tento krok je možné vynechat (resp. se vynechá) pokud byly správně nadefinovány relace.
5. Uspořádat ovládací prvky - stará se o rozložení prvků v „hlavním formuláři“ a „podformuláři“. Pro hlavní formulář doporučuji jakékoliv rozložení kromě „jako tabulku“, pro podformulář doporučuji zejména „jako tabulku“. Funkční je nicméně jakákoliv kombinace rozložení.
6. Nastavit zadávání dat se stará o nastavení práv, která bude mít uživatel k práci s formulářem. Je např. možné zakázat úpravu dat, přidávání údajů nebo výmaz
7. použít styly – poskytně jednoduše aplikovatelný vzhled formuláře, na funkčnost celého formuláře
8. Nastavit název – nastaví název pod kterým bude formulář uložen v databázi.

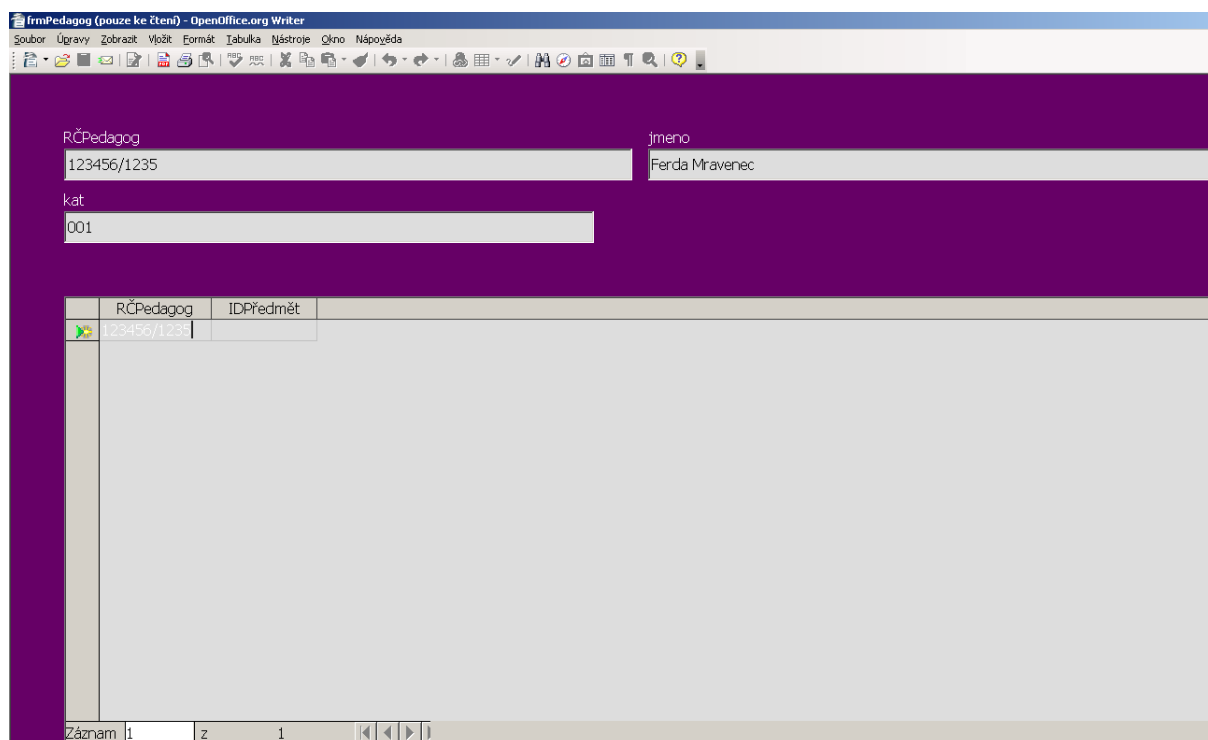
Výsledkem procesu návrhu je formulář jako třeba na obr. 34.

Práce s formuláři v režimu editace dat má určitá specifika. Např. grid podformuláře je iniciován pro nový záznam až po té, co přejdete na jiný záznam a vrátíte se zpět.

Podobná specifika má i režim návrhu. Dvojitým kliknutím na kterémkoliv datovém prvku získáte k dispozici jeho vlastnosti. Při zobrazování vlastností si dejte pozor, co ve skutečnosti prohlídíte. Textová pole se totiž implicitně vybírají i s popisky k nim připojenými a okno vlastností pak zobrazuje pouze společné vlastnosti. Jednotlivý prvek vyberete držením klávesy CTRL a kliknutím na prvek.

Ze všech vlastností jsou asi nejdůležitější vlastnosti obsažené na záložce data. Zde je totiž definováno připojení na tabulky. Ve vlastnostech polí je možné nadefinovat pouze připojení k jiné položce z tabulky, v případě, že je nutné předefinovat samotnou tabulku, potom je třeba zobrazit vlastnosti formuláře jako celku. To provedeme tak, že pravým

tlačítkem klikneme na kterémkoliv poli formuláře a z kontextového menu vybereme *Formulář ...*



Obr. 34: Formulář

Dodatečná pole (nebo celý nový formulář) můžeme definovat pomocí nástrojů obsažených na panelu nástrojů *Návrh formuláře*. Ten je možné zobrazit přes menu *Zobrazit -> Panely nástrojů -> návrh formuláře*.

Při manuálním vytváření formuláře postupujeme tak, že nejprve nadefinujeme vlastnosti formuláře jako celku (především datové) a potom datově navazujeme jednotlivá pole formuláře.

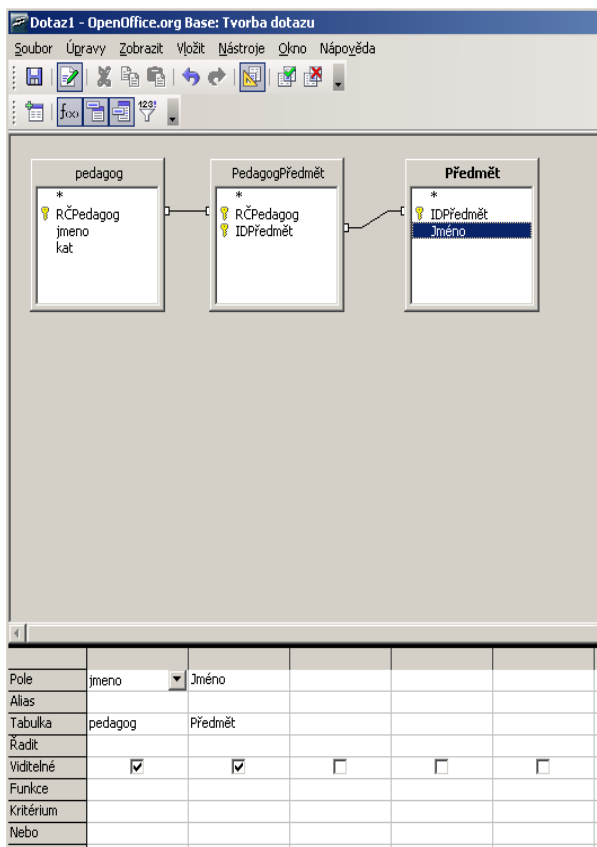
### 3.4 Dotazy

Dotazy slouží k propojování tabulek za účelem získání údajů z nich v tabulkové formě. Alternativně lze dotaz využít jako základ pro tvorbu výstupních sestav. Příklad tvorby dotazu je na obr. 35.

Jednotlivé položky dotazu můžeme definovat prostým přetažením polí z tabulek směrem dolů do volných polí dotazu. Pro položky stejného jména, byť z různých tabulek, je potřeba nadefinovat tzv. alias neboli alternativní jméno. OO Base by totiž nebylo schopno samo o sobě rozhodnout, kterou položku ve skutečnosti odkazujeme, nadefinováním aliasu tuto nejistotu odstraníme.

Řazení a sestavování kritérií funguje analogicky k podobným funkcím MS Access s výjimkou parametrických dotazů, které OO Base nepodporuje. Zároveň se částečně liší také způsob vytváření tzv. agregačních dotazů. OO Base považuje každý dotaz za normální až do chvíle, než je u něj definováno agregační funkce. Od té chvíle se s dotazem pracuje jako s agregačním dotazem.

Všechny položky agregačního dotazu musí být opatřeny funkcemi. Textové položky seskupujeme, na položky číselné můžeme aplikovat výpočetní funkce.



Obr. 35: Tvorba dotazu

### 3.5 Sestavy

Sestavy je možno vytvářet pouze na základě tabulky nebo dotazu (všimněte si jednotného čísla). Pokud je v sestavě potřeba využít více tabulek, je nutné vytvořit dotaz, který je propojuje.

Příklad sestavy je znázorněn na obr. 36.

**Title:**  
**Author: Pavel Šenovský**  
**Date: 5/1/06**

---

<i>RČPedagog</i>	<i>jmeno</i>	<i>kat</i>
123456/123	Ferda Mravenec	00
5		1

Obr. 36: Příklad sestavy

## Literatura

- [1] *Domáci stránky Open Office*. Dostupné z WWW <URL: <http://www.openoffice.org>> [cit. 20.4.2006]
- [2] *Domáci stránky Javy*. Dostupné z WWW <URL: <http://www.java.com/>> [cit. 20.4.2006]