VŠB - Technická univerzita Ostrava, Fakulta bezpečnostního inženýrství

doc. Ing. Pavel Šenovský, Ph.D.

Bezpečnostní informatika 1
Návody do cvičení

skripta
1. vydání

Ostrava, 2017

Bezpečnostní informatika 1 - Návody do cvičení
1. rozšířené vydání
tento text neprošel jazykovou úpravou

Vysoká škola báňská - Technická univerzita Ostrava, Fakulta bezpečnostního inženýrství

Obsah

Seznam obrázků

Seznam tabulek

Slovník

ERD Entity Realtionship Diagram.

GUI Graphical User Interface.

JRE Java Runtime Environment.

MS Microsoft.

OLE Object Linking and Embedding.

RD Realtionship Diagram.

SQL Structured Query Language.

VBA Visual Basic for Applications.

WWW World Wide Web.

XML Extensive Markup Language.

[title=Seznam zkratek]

Úvod

Vážený studente, dostává se Vám do rukou učební text předmětu *Bezpečnostní informatika I - Návody do cvičení*.

Mým cílem při psaní tohoto textu bylo, aby čtenář získal základní přehled v oblasti uživatelského užití databázových systémů. Tento text reaguje na potřeby praxe v oblasti extenzivní práce s daty, obvykle prostřednictvím informačních systémů. Systémy databázové obvykle slouží jako „backend“ takových systémů.

Zvládnutí databázových systémů tak přispívá k hlubšímu pochopení fungování informačních systémů a zároveň dává Vám - čtenářům do rukou mocný, obecně použitelný nástroj použitelný pro shromažďování a editaci dat, může však posloužit také pro účely filtrace dat a jako základ datových analýz, zejména pokud se použije společně s dalšími analytickými nástroji - ať už je to MS Excel nebo nástroje pokročilejší - R, SAS a nebo další.

Text je zaměřen na „desktopové databáze“, zejména pak MS Access. Výhodou je uživatelská přívětivost tohoto nástroje, která Vám usnadní výukový proces. Znalosti, které získáte v tomto nástroji pak můžete aplikovat při použití pokročilejších databází klien-server, jelikož se v obou případech jedná o relační databáze - a tedy systémy sdílející základní filozofii použití.

Studium tohoto textu nepředpokládá žádné předchozí znalosti z oblasti výpočetní techniky, kromě základů práce s programy MS Office (ačkoliv pokud nějaké znalosti v oblasti IT máte, rozhodně to nebude na škodu).

Celý text je orientován na praktické zvládnutí zvoleného databázového produktu, proto se v těchto skriptech setkáte s řadou příkladů. Příklady, uvedené ve skriptech, budou řešeny v produktech MS Access 2007, LibreOffice Base a databáze Kexi. Při volbě jiné databáze se jednotlivé kroky mohou do určité míry lišit – základní princip by ale měl zůstat stejný.

Celý text je rozdělen do čtyř částí. První, teoretické, ve které bude vysvětlena základní teorie okolo databází, tvorby Entity Realtionship Diagram (ERD) modelů jako základních předpokladů správného řešení problému v databázovém prostředí. Zbylé tři kapitoly jsou zaměřeny na praktickou realizaci databáze pomocí MS Access, LibreOffice Base a databáze Kexi.

Pro zpříjemnění čtení jsem se také rozhodl, zpracovat tento text formou vhodnou pro „distanční vzdělávání“, tak aby práce s ním byla co možná nejjednodušší. Z tohoto důvodu je text jednotlivých kapitol segmentován do bloků.

Každá kapitola začíná náhledem kapitoly, ve kterém se dozvíte, o čem budeme v kapitole mluvit a proč. V bodech se pokusím shrnout, co byste po prostudování kapitoly měli znát a kolik času by Vám studium mělo zabrat. Prosím mějte na paměti, že tento časový údaj je pouze orientační, nebuďte proto prosím smutní nebo naštvaní když ve skutečnosti budete kapitole věnovat o něco méně nebo více času.

Za kapitolou následuje shrnutí, ve kterém budou zdůrazněny informace, které byste si rozhodně měli zapamatovat. To, že jste správně pochopili probíranou látku, si budete moci ověřit pomocí kontrolních otázek, které by Vám měly poskytnout dostatečnou zpětnou vazbu k rozhodnutí, zdali jít dále nebo věnovat delší čas opakování.

Pro zjednodušení orientace v textu jsem zavedl systém ikon:

Přeji Vám, abyste čas, který strávíte s tímto textem, byl co možná nejpříjemnější a abyste jej nepovažovali za zcela ztracený. Databáze jsou totiž sice z počátku obtížnější na zvládnutí, přesto jejich plnohodnotné zvládnutí rozhodně za to stojí, protože Vám takto nabyté znalosti mohou pomoci efektivněji řešit problémy, se kterými je možné se v praxi setkat.

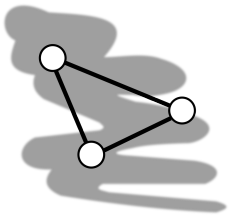
Průvodce studiem
Slouží pro seznámení studentů s látkou, která bude v kapitole probírána.

Čas nutný ke studiu
Představuje odhad doby, který budete potřebovat k prostudování celé kapitoly. Jedná se pouze o orientační odhad, neznepokojujte se proto, pokud Vám studium bude trvat o něco déle nebo budete hotovi rychleji.

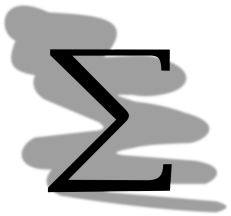
Vysvětlení, definice, poznámka
U této ikony najdete vysvětlující text, poznámku k probíranému tématu, která problém uvede do širších souvislostí, popřípadě důležitou definice.

Kontrolní otázky
Na závěr každé kapitoly je zařazeno několik otázek, které prověří, zda jste problematice kapitoly dostatečně porozuměli. Pokud nebudete vědět odpověď na některou otázku, je to signál pro Vás, abyste se ke kapitole vrátili.

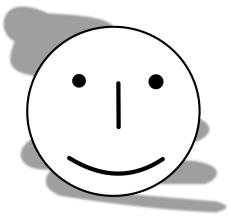
Příklad
Příklady obsahují praktické demonstrace diskutovaného problému.



Návaznosti
V tomto segmentu budou zmíněny další návaznosti probíraného tématu na další témata tohoto předmětu, ale také dalších předmětů.



Shrnutí
Obsahuje základní myšlenky kapitoly, kterým by měl být věnována zvláštní pozornost během studia.



Přestávka
Po obtížné části textu, nebo prostě občas jenom tak je nutné si udělat krátkou přestávku, načerpat síly k novému studiu.

Kapitola 1

Stručný a jednoduchý úvod do teorie databází

Náhled kapitoly
V této kapitole se seznámíme s nezbytnými základy databázové teorie a položíme si tak solidní základy pro návrh databáze ve zvoleném databázovém systému

Po přečtení této kapitoly budete umět

1. navrhnout strukturu databáze,
2. a formálně ji vyjádřit pomocí ERD diagramu a také
3. se vyvarovat častých chyb při návrhu databáze

Čas pro studium
Tuto kapitolu je možné prostudovat během 15 – 20-ti minut, ale dejte si načas a dejte si skutečně záležet na správném pochopení myšlenek představených v této kapitole. Je to nutná, bohužel však nikoliv postačující podmínka pro zvládnutí správného návrhu databáze.

1.1 Typologie databází

Databáze jsou jedním z nejstarších typů aplikací. Jejich počátek je prakticky totožný s počátkem nasazování počítačů k uchovávání údajů a za desítky let provozu také prošly poměrně výrazným vývojem.

Prvním typem databází, který se rozšířil, byly *databáze stromové*, ty byly později nahrazeny *databázemi síťovými*. Architekturu těchto systémů zde rozebírat nemá smysl, protože oba výše zmíněné typy databázi jsou beznadějně zastaralé a už se mnoho let nepoužívají. Byly nahrazeny během osmdesátých let databázemi relačními a tento typ databází setrval, až do dnešních doby, v pozici nejpoužívanějších typ databázových systémů. *Relačními databázemi* se proto budeme také zabývat v tomto textu (s výjimkou této kapitoly) a to výhradně.

Příčinu úspěchu relačních databází je možné vysledovat ke způsobu, jakým pracují s daty. Data jsou totiž shromažďována v tabulkách, které ovšem můžeme vzájemně propojovat pomocí spojení, které nazýváme relacemi. Odtud název relační databáze. Relační databáze můžeme dělit do dvou skupin:

1. *velké (serverové) databáze* – jsou určeny pro provoz na serveru. Předpokládá se u nich schopnost obsluhovat velké množství požadavků zároveň z různých míst a od různých uživatelů a schopnost pracovat s velmi rozsáhlými objemy dat. Příkladem takových databází může být systém Oracle, Informix, MS SQLServer, MySQL, Postgress a další.

2. *Osobní (stolní) databáze* – jsou určeny k běžnému používání na klasických osobních počítačích – užívá je tedy obvykle pouze jeden uživatel. Na rozdíl od serverových databázových systémů osobní databáze obsahují nástroje, které uživatelům usnadňují navrhování formulářů sestav a podobně – jsou tedy uživatelsky přívětivé. Cenou za snadnost použití je však nižší výkon databáze a do určité míry také omezená funkčnost – zejména v oblasti škálovatelnosti řešení apod. Mezi osobní databáze bychom mohli zařadit třeba Paradox, Fox Pro, DBase nebo MS Access.

My se v předmětu Bezpečnostní informatika a v těchto skriptech budeme věnovat pouze databázím osobním a to konkrétně systémy MS Access, LibreOffice Base a Kexi.

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 mezi databázemi relačními a objektovými je filozofie konstrukce databází. Zatímco relační databáze si vystačí s tabulkami a relacemi, databáze objektové pracují s objekty a vztahy mezi nimi.

Zatímco tabulku jsme obvykle schopni si nějakým způsobem představit – odvodit ji z nějakého formuláře s objekty je to poněkud složitější. Jednotlivé objekty vycházejí ze svých předobrazů v reálném světě. Podobně jako ony mají nějaké vlastnosti a nějakým způsobem se chovají. Chování u objektů zprostředkovávají metody objektu. Objektové databáze tedy s objekty pracují jako s určitými šablonami. Pokud tyto šablony vyplníme konkrétními údaji, získáme tzv. *instanci objektu*.

I v objektových databázích je vyžadováno udržování určitých vztahů mezi objekty. Tyto vztahy však mohou být podstatně složitější než vztahy u relačních databází. Objektové databáze proto zvládají vztahy agregační (objekt se skládá z několika dalších objektů), dědičnost (podřízené objekty přejímají vlastnosti a metody svých nadřízených objektů) atd.

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 přitom mnohem náročnější. Právě tento fakt pravděpodobně způsobil, že objektové databáze mají na trhu pouze zanedbatelný podíl.

Posledním typem databází, o kterých se zde zmíním, jsou *XML databáze*. Extensive Markup Language (XML) [7] označuje obecný značkovací jazyk, použitelný pro binárně nezávislý popis dat libovolného druhu. Standard XML, 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.

Dopis v XML formátu

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

```
<dopis>
  <adresat>
    <jmeno>Jan</jmeno> <prijmeni>Nepomu</prijmeni>
    <adresa>
        Ulice 15
        Ostrava
        702 00
    </adresa>
  </adresat>
  <vec>Konference ABC2015</vec>
  <teloDopisu> text dopisu </telo_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 je pomocí klasických relačních databází 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.

1.2.1 Konceptuální fáze

V konceptuální fázi dochází ke shromažďování všech dostupných informací relevantních z hlediska řešení problému a to bez ohledu na formu těchto informací. Forma totiž bude více méně závislá na typu problému, který hodláme databází řešit. Na počátku této fáze si jsme tedy jistí pouze tím, že problém chceme vyřešit prostřednictvím zvoleného databázového prostředí. K řešení pak přistupujeme bez nějakých osobních zájmů, představ, s „hlavou otevřenou“.

Jako základ informací pro řešení mohou sloužit třeba formuláře zpracovávané elektronickou nebo papírovou formou. Pomocí nich jsme obvykle schopni odhalit, jaké informace jsou v současné době 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, které nově tvořený systém bude podporovat. Zároveň lze tímto způsobem získat také osobní zkušenost stávajících uživatelů z pohledu - jak si představují funkci nového řešení. Konzultacemi lze také rozlišit, které údaje jsou skutečně využívány, popř. by je nově bylo výhodné zahrnout do řešení.

Výsledkem této fáze jsou shromážděné formuláře, poznámky z konzultací a myšlenkový model řešení – první představa o řešení, kterou jsme si vytvořili během konzultací. Tyto představy jsou zpracovány volnou formou. Pokud je proto budeme chtít použít pro návrh databáze, budeme je muset nejprve zpracovat - formalizovat.

K takové formalizaci dochází postupně v průběhu fáze tvorby ERD modelu.

1.2.2 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 představují entity problému, které jsou důležité z pohledu jeho řešení. K těmto entitám shromažďujeme také podrobnější informace popisující vlastnosti entity - tyto označujeme jako *atributy*.

Entity v ERD diagramu chápeme jako virtuální reprezentace objektů reálného světa. Entita si lze jednoduše představit jako tabulku, která má jednotlivé sloupce - atributy entity. Tyto atributy představují typy informací, které mají být v entitě udržovány. Např. pro entitu člověk můžeme vést atributy jméno, věk, pohlaví, výška, váha, ...

Do jednotlivých řádků tabulky zapisujeme hodnoty atributů pro určitý konkrétní výskyt entity - v našem případě hodnoty konkrétního člověka.

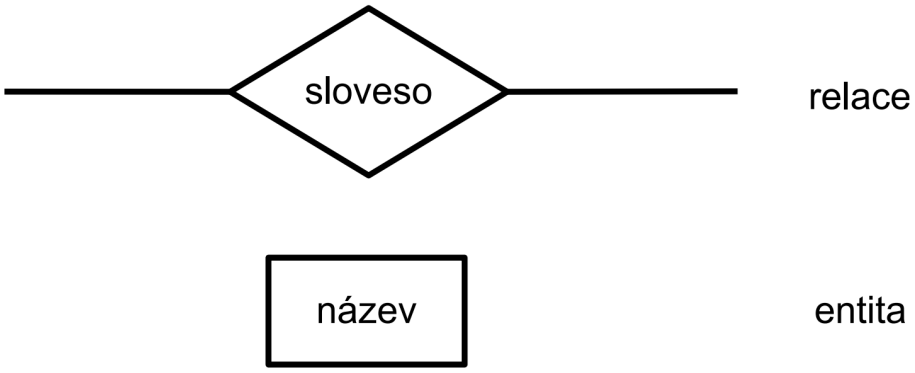
Kromě entit, které reprezentují konkrétní objekty reálného světa, musíme často do ERD přidávat uměle entity nové, jejichž účelem je technické zajištění možnosti převodu ERD do databázového systému.

Právě tvorba těchto „umělých“ entit a jejich propojování relacemi je pro začátečníka obtížná na pochopení, proto prosím věnujte zvýšenou pozornost následujícím stránkám a také praktické ukázce vytváření tabulek a vztahů mezi nimi v rámci výkladu práce se zvoleným databázovým systémem.

Upozornění
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 – entity vytipováváme z podstatných jmen, která se nám objevují v konceptuální fázi tvorby modelu. Vytvoříme si, proto, seznam podstatných jmen a z něj vyloučíme synonyma tak, aby každý objekt nebo atribut se v seznamu objevil pouze jednou.

V ERD diagramu značíme entity obdélníčkem, do kterého vepisujeme jméno entity. Relace značíme jako spojnice mezi entitami, která má na zakončeních relace označenou tzv. četnost spojení a uprostřed kosočtverec se slovesem definujícím povahu vztahu mezi relací propojenými entitami. Základní konstruktory ERD jsou znázorněny na obr. 1.1.



Obrázek 1.1: Konstruktory ERD diagramu

S použitím na obr. 1.1 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) jsou nejčastějším případem, ale existují i tzv. ternální vazby mezi třemi entitami (popřípadě vztahy mezi ještě větším množstvím entit). Tyto vícečetné vztahy se ale vyskytují mnohem méně a mají navíc zásadní nevýhodu – není možné je přímo převést do databázového prostředí.

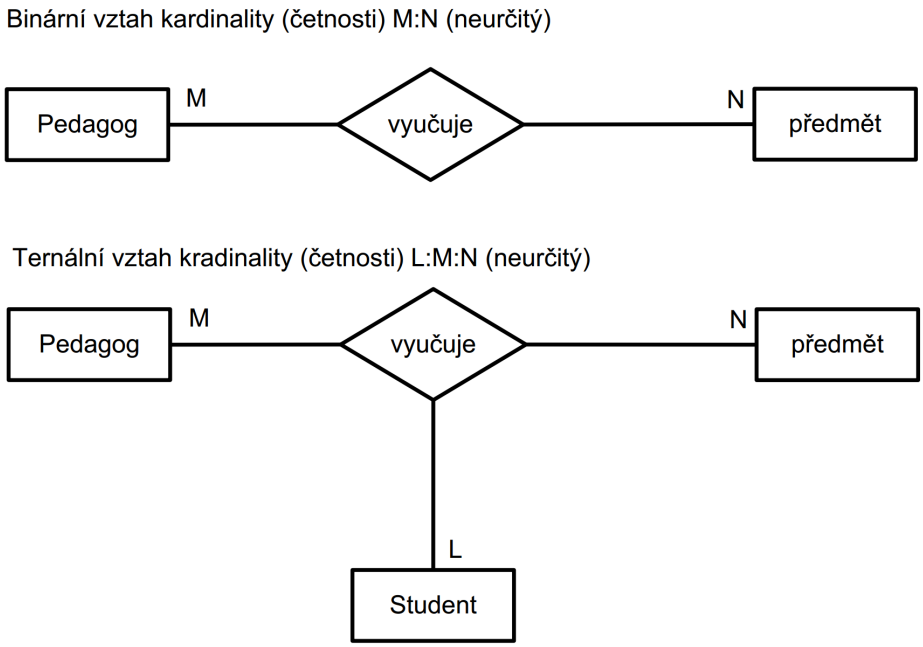
Tedy do databáze lze zavést pouze binární vztahy. Vícečetné vztahy v počátečních fázích práce na ERD diagramu slouží k základní formalizaci informací, které máme o řešeném problému. Následně tyto vícečetné vztahy postupně odstraňujeme až nám v ERD zůstanou pouze vztahy binární. Způsob, jakým přesně tento problém řešíme, si ale necháme na později.

Zatímco entity jsou popisovány prostřednictvím podstatných jmen, vztahy mezi nimi popisujeme prostřednictvím sloves. ERD tak lze číst jako jednoduché věty popisující řešený 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 o binární vztah. Při zkoumání kardinality se ptáme:

- kolik předmětů může učit jeden pedagog → nevíme (N)
- 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. 1.2).



Obrázek 1.2: Binární a ternální vztah (ERD diagram)

Upozornění
K relacím je potřeba také dodat, že ve finálním návrhu, který má být implementován v prostředí zvolené databáze, by se neměly vyskytovat také neurčité a ternální vazby, ty totiž také nejsou v databázích implementovatelné.

V počátečních fázích tvorby ERD tyto vazby používáme pro formalizaci našeho myšlenkového modelu, se kterým dále pracujeme. Tedy primárně nás na počátku nezajímá, jakým způsobem bude přesně vypadat naše databáze – soustřeďujeme se pouze na to, aby v našem modelu byla obsažena schopnost vést veškeré potřebné údaje.

Před tím, než se vrhneme na řešení neurčitosti a ternálních vazeb se vrátíme k atributům, tedy specifikaci dat, která o entitě budeme shromažďovat.

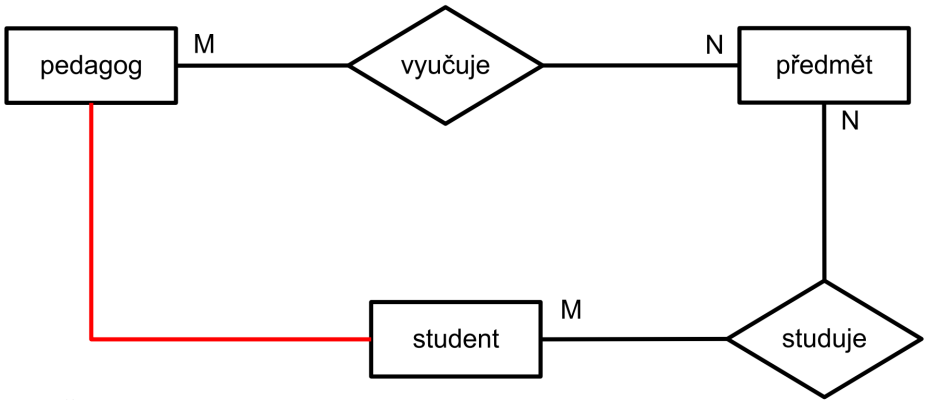
Zkusme se zamyslet nad atributy entity Pedagog. Tato entita bude zcela jistě obsahovat atributy Jméno, Příjmení, katedra a takovým způsobem bychom mohli pokračovat dále. Všem zde zmíněným atributům schází jedna podstatná vlastnost – nejsou schopné jednoznačně identifikovat výskyt entity – tedy konkrétního pedagoga. Atributům, které mají tuto schopnost, říkáme *kandidátské klíče*. Z těchto klíčů vybíráme jeden, který označíme jako *klíč primární*. Kandidátskými klíči pro entitu pedagog by mohly být např. následující atributy: identifikační číslo z evidence zaměstnanců, rodné číslo, číslo pojištěnce apod. Z těchto klíčů vybereme jeden klíč – např. identifikační číslo a ten budeme označovat jako primární.

Navrhněme atributy všech tří entit:
Pedagog: #IČPedagog, Jméno, Příjmení, Katedra, jménoKatedry
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.

U složeného primárního klíče není tedy unikátní výskyt hodnoty atributu každého jednoho atributu primárního klíče, ale jejich kombinace.

Nyní se vraťme k řešení vztahů M:N a ternálním vazbám. Začněme ternálními vazbami. Ty můžeme nahradit dvojicí binárních vztahů, které ovšem zůstávají neurčité. Řešení je demonstrováno na obr. 1.3.



Obrázek 1.3: Řešení ternálních vztahů v rámci ERD

Tedy jeden ternální vztah mezi entitami pedagog, předmět a student jsme nahradili dvěma binárními vztahy pedagog – předmět a student – předmět. Všimněte si na obr. 1.3 červené spojnice mezi entitami pedagog a student. Za určitých okolností by totiž mohlo mít smysl i vytvoření třetí relace mezi těmito entitami, ale v našem případě tomu tak není.

Cílem relací v ERD diagramu je propojit entity a umožnit tak uživateli pomocí definovaných spojení dostat se ke všem potřebným souvisejícím záznamům. V ERD je tedy samostatně stojící entita spíše výjimka.

Protože ani vztahy kardinality M:N nejsou realizovatelné pomocí databázových systémů (resp. technicky realizovatelné jsou, ale s výslednou databází není možné efektivně pracovat), musíme se zbavit i těchto relací.

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.

Pokud jsou v našem případě primární klíče entit následující: Pedagog - #IČPedagog, student - #IČStudent a předmět - #IČPředmět, budou nové entity vypadat následovně:

StudentPředmět: #IČPředmět, #IČStudent

PedagogPředmět: #IČPedagog, #IČPředmět

1.2.3 Relační diagram

Tuto změnu budeme však řešit pomocí jiného diagramu – diagramu relačního (RD). RD podobně jako ERD má dva základní konstruktory: *tabulku* a *relaci*. Entity transformujeme v RD do podoby tabulek a jejich atributy transformujeme do podoby sloupců tabulek. Relace zůstávají v nezměněné podobě, s tím že relační diagram je schopen zachycovat pouze binární relace. Oproti ERD se také liší grafická podoba relace. Znázorňujeme ji pomocí prosté spojnice mezi tabulkami.

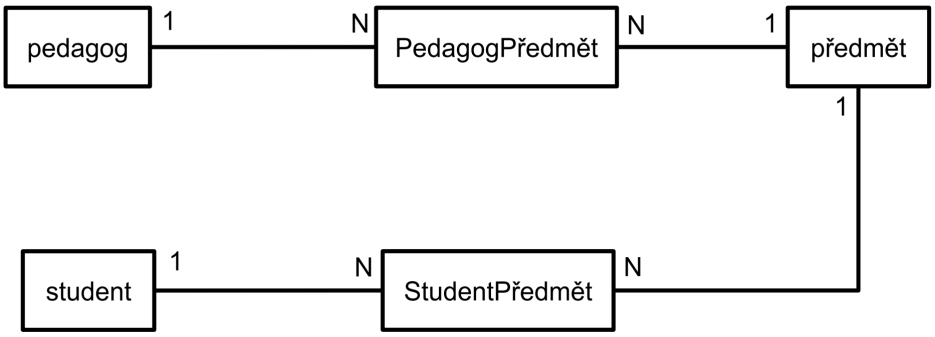
Grafické řešení je zobrazeno na obr. 1.4.

Pro zjednodušení notace někdy v relacích používáme šipky místo číselného označení kardinality vztahu. Modifikovaný RD by pak mohl vypadat podobně jako na obr. 1.5.

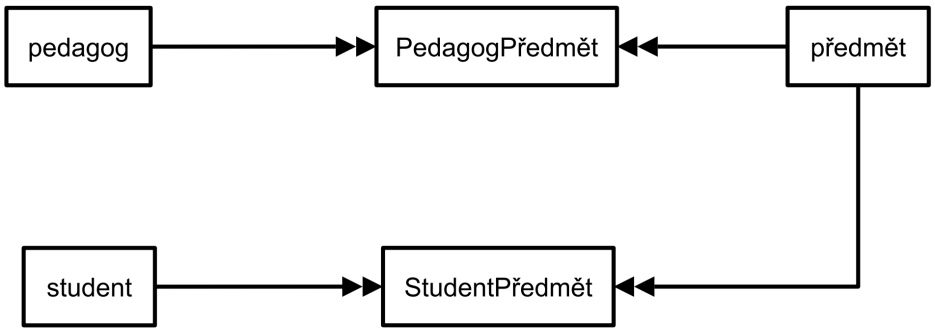
1.2.4 Pravidla návrhu tabulek

V kapitole 1.2 jsme se zabývali metodikou správného návrhu databáze jako propojení tabulek pomocí relací. Seznámili jsme se také s řešením dvou základních problémů při návrhu struktury databáze. Bohužel tyto problémy nejsou problémy jedinými, které z hlediska struktury musíme řešit.

Zbývající problémy však souvisí se samotnou strukturou tabulek.



Obrázek 1.4: Grafické řešení vtahů kardinality M:N



Obrázek 1.5: RD – relace pomocí šipek

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

Proces, v rámci kterého měníme strukturu tabulek, nazýváme *normalizací*. Snažíme se tedy změnit strukturu tabulek tak, aby byla v jedné z tzv. *normálních forem*. Těchto normálních forem je celkem pět a každá řeší jeden problém týkající se integrity databáze.

První normální forma řeší problém opakujících se atributů. Představme si, že tabulka pedagog, kterou jsme si strukturálně definovali výše, má evidovat údaje o kontaktních telefonních číslech. První co nás napadne je nepochybně doplnění sloupce telefon do tabulky pedagog. To je samo o sobě v pořádku, problém ale nastane, pokud začneme uvažovat o možnosti, že pedagog kromě čísla do kanceláře může mít i číslo na mobil a taky domů a na chatu a vlastně má ještě jeden mobil a zde nastává problém.

Problém opakujících se sloupců spočívá v tom, že takové sloupce se velmi obtížně prohledávají tím, že není možné předem říci, kde přesně se budou požadované informace nacházet. Řekněme, že jsme získali k pedagogům seznam nových kontaktních telefonů a chceme aktualizovat naši tabulku v databázi a v případě, že v ní kontaktní telefon není obsažen - potřebujeme ho do databáze doplnit. Pro každý kontaktní telefon v novém seznamu se budeme muset podívat do všech sloupců tabulky, které by tento údaj mohly teoreticky obsahovat – to zabere spoustu času a také programátorského úsilí, protože realizace nějakou standardní cestou není možná. Přitom se takovému problému můžeme velmi jednoduše vyvarovat přizpůsobením struktury databáze.

Vytvoříme si tabulku kontakty, do které zavedeme jako složený primární klíč primární klíč původní tabulky pedagog a samotné telefonní číslo bude představovat druhou část primárního klíče. Do nové tabulky můžeme zavést ještě komentář doplňující dodatečné informace k telefonnímu číslu. Nová struktura tabulek bude vypadat následovně:

Pedagog: #IČPedagog, Jméno, Příjmení, Katedra, jménoKatedry
Student: #IČStudent, Jméno, Příjmení
Předmět: #IČPředmět, Jméno, Kredity, hodincv, hodinpř
StudentPředmět: #IČPředmět, #IČStudent
PedagogPředmět: #IČPedagog, #IČPředmět

Kontakty: #IČPedagog, #Telefon, komentář

Relace mezi tabulkami pedagog a kontakty bude vedena mezi sloupci #IČPedagog obou tabulek. (Relaci je možné ale definovat mezi libovolně pojmenovanými sloupci – není podmínkou, že se budou jmenovat stejně).

Definice
Databáze je v první normální formě (1. NF) pokud se v tabulkách nevyskytují opakující se sloupce nebo skupiny sloupců.

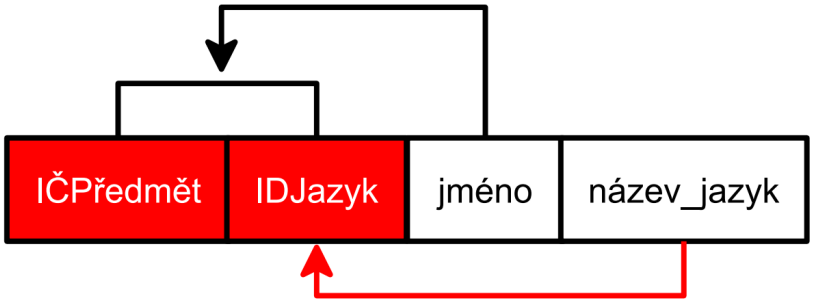
Kontrolní příklad
Uvažujte tabulku předmět v následující struktuře:
Předmět: #IČPředmět, Název, NázevAngličtina

Navrhněte změnu ve struktuře tabulek tak, aby mohly být zavedeny názvy a (nově) anotace předmětů v češtině, angličtině, ruštině a němčině.
Svá řešení konzultujte s Vašimi kolegy v rámci cvičení nebo s vyučujícím na přednáškách nebo cvičení.

Druhá normální forma (2. NF) požaduje, aby všechny neklíčové sloupce byly plně závislé na primárním klíči. Abychom demonstrovali tento problém, zkusme vyřešit problém s anotacemi předmětů a jejich jazykovými mutacemi.

Z hlediska 1. NF nejsou přípustné opakující se sloupce, proto řešení bude vypadat následovně:
Předmět: #IČPředmět, kredity, hodincv, hodinpř
JménoPředmět: #IČPředmět, #IDJazyk, jméno, názevjazyk

Takové tabulky jsou nepochybně v 1. NF, jelikož neobsahují opakující se sloupce. Bohužel ale nejsou v 2. NF. Podívejme se na grafické znázornění závislostí v tabulce JménoPředmět (viz. obr. 1.6).



Obrázek 1.6: Funkční závislosti tabulky JménoPředmět

Na obr. 1.6 je červeným obdélníkem znázorněn primární klíč tabulky. Červenou šipkou je pak zvýrazněna problematická závislosti. Název jazyka není závislý na celém primárním klíči, ale pouze na identifikátoru jazyka, tedy na části primárního klíče.

Připomínám, že v efektivní databázi jsou evidované údaje pouze na jednom místě, proto jediné místo, kde můžeme nalézt název jazyka je právě tabulka jménoPředmět a to není dobré – pokud např. aktuálně nemáme hotový překlad názvu předmětu do jazyka X, pak jazyk X pro naši databázi není znám a to je chyba (např. z hlediska tvorby GUI).

Řešením je rozdělením tabulky na dvě:
JménoPředmět: #IČPředmět, #IDJazyk, jméno
Jazyk: #IDJazyk, názevjazyk

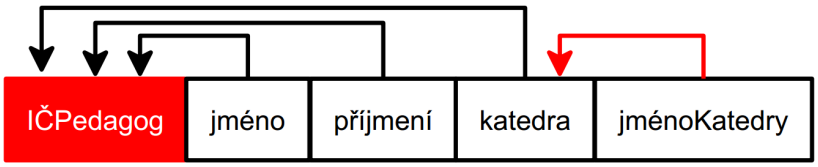
Ve skutečnosti je výše uvedené řešení sice „informaticky křišťálově čisté“ nicméně, z hlediska řešení naší databáze poněkud nepraktické. Opravme proto znovu tabulku předmět následujícím způsobem:
Předmět: #IČPředmět, **jméno**, kredity, hodincv, hodinpř

Sloupec jméno v tabulce předmět bude obsahovat redundantní (nadbytečná) data. Protože ale nebudeme vytvářet jazykové mutace jednotlivých formulářů jeho zavedení nám při tvorbě formulářů podstatně ulehčí práci.

Kontrolní příklad
Uvažujte tabulku Diplomky o následující struktuře:
Diplomky: #IČStudent, #IČPedagog, Název, katedraPedagog

Uveďte tabulku do 2.NF. Svá řešení konzultujte s Vašimi kolegy v rámci cvičení nebo s vyučujícím na přednáškách nebo cvičení.

Třetí normální forma (3. NF) požaduje, aby všechny neklíčové sloupce byly plně netranzitivně závislé na primárním klíči. Proveďme analýzu funkčních závislostí tabulky pedagog (viz. obr. 1.7).



Obrázek 1.7: Funkční závislosti sloupců tabulky Pedagog

Jméno katedry přímo závisí na katedře (třeba číslu katedry), nikoliv však na samotném pedagogovi. Připomínám, že v efektivní databázi jsou evidované údaje pouze na jednom místě, proto jediné místo, kde můžeme nalézt jméno katedry je v našem případě tabulka pedagog.

To s sebou nese určité problémy. Například pokud chceme zřídit novou katedru, která ale nemá žádné přiřazené pedagogy, struktura tabulky z obr. 1.7 to neumožňuje. Ještě ničivější je situace kdy na katedru máme napojeny další tabulky s údaji, které potřebujeme evidovat i do budoucna, ačkoliv samotná činnost katedry končí a odebíráme z ní všechny pedagogy. V takovém případě v navázaných tabulkách zůstanou záznamy katedry buďto „bezprizorní“ anebo se dokonce smažou, což je v přímém rozporu s našimi zájmy.

Problém odstraníme tak, že tabulku rozdělíme na dvě:
Pedagog: #IČPedagog, jméno, příjmení, katedra
Katedra: #katedra, jménoKatedry

Kontrolní příklad
Uvažujte tabulku Pedagog v následující struktuře:
Předmět: #IČPedagog, jméno, IDKatedra, IDZdravotníPojišťovna, jménoZdravotníPojišťovny

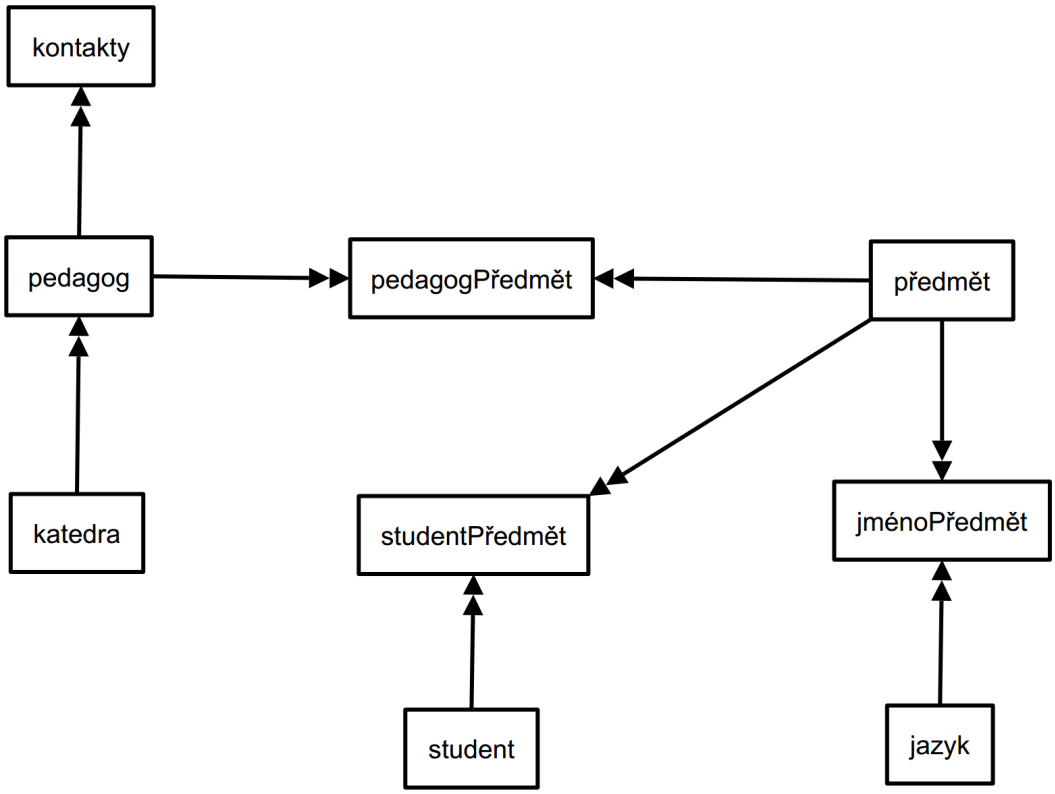
Uveďte tabulku do třetí normální formy. Svá řešení konzultujte s Vašimi kolegy v rámci cvičení nebo s vyučujícím na přednáškách nebo cvičení.

Pro většinu databázových aplikací normalizace do 3. NF postačuje. 4. a 5. NF se pak zabývají určitými specifickými problémy souvisejících s použitím tzv. kandidátských klíčů (klíčů, které by mohly sloužit jako primární klíč, ale my jsme je jako primární klíč nevybrali). Jelikož naším primárním cílem není úplný výklad databázové teorie, bude nám tento teoretický výklad stačit.

Podívejme se, k jaké struktuře tabulek jsme postupně došli:
Pedagog: #IČPedagog, jméno, příjmení, katedra
Katedra: #katedra, jménoKatedry
JménoPředmět: #IČPředmět, #IDJazyk, jméno
Jazyk: #IDJazyk, názevjazyk
Předmět: #IČPředmět, jméno, kredity, hodincv, hodinpř

Student: #IČStudent, Jméno, Příjmení
StudentPředmět: #IČPředmět, #IČStudent
PedagogPředmět: #IČPedagog, #IČPředmět
Kontakty: #IČPedagog, #Telefon, komentář

A ještě graficky (viz. obr. 1.8):



Obrázek 1.8: Funkční závislosti sloupců tabulky Pedagog

Závěrem kapitoly důležité upozornění
Výše uvedená pravidla a problémy, které jsou řešeny normalizací databáze, jsou velmi důležité. Jejich nedodržení může vést k celé řadě nepříjemných problémů, kterých se obvykle chceme vyvarovat. Přesto normalizace není žádné dogma, od kterého se v žádném případě nelze odchýlit – odchýlit se můžeme, je ale nutné, abychom tak učinili vědomě – tedy s tím, že jsme si plně vědomi následků, které takový krok pro nás má.
Důvodem pro porušení pravidel může být jednoduchost řešeného problému. Nebo určitá specifika, která nám umožní některé problémy v rámci zjednodušení práce ignorovat.
V tomto předmětu ale postupujte tak, aby Vaše výtvory splňovaly pravidla normalizace.

Kontrolní příklad
Navrhněte strukturu tabulek pro řešení problému inventarizace majetku ve firmě. Kromě údajů o „inventáři“ by navržená struktura tabulek měla umožňovat tzv. pasportizaci objektu a evidenci zaměstnanců (s k nim přináležejícího inventáře). Svá řešení konzultujte s Vašimi kolegy v rámci cvičení nebo s vyučujícím na přednáškách nebo cvičení.

Kapitola 2

MS Access

Průvodce studiem
V této kapitole se konečně dostaneme k praktické realizaci databáze v prostředí MS Access 2007.
Po prostudování této kapitoly budete
umět

- Implementovat navrženou strukturu databáze v MS Access 2007,
- Navrhovat
  - Formuláře
  - Dotazy
  - Sestavy

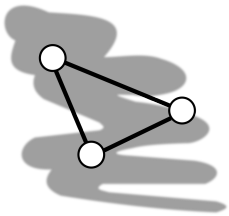
Čas nutný ke studiu
Na prostudování této kapitoly budete potřebovat několik hodin – postupujte pomalu a postupy uvedené v těchto skriptech si hned prakticky zkoušejte – získání praktických dovedností je velmi důležité!
Při přechodu mezi jednotlivými podkapitolami si udělejte přestávku, tak ať můžete nabrat nových sil do dalšího studia.

2.1 Úvod do MS Access

MS Access je standardní součástí MS Office ve verzi Profesional nebo vyšší.

Kompatibilita
Datové formáty jednotlivých verzí spolu, na rozdíl od ostatních programů rodiny Office, nejsou kompatibilní a je nutné je před použitím v jiné verzi Access konvertovat. Nekompatibilní jsou například verze MS Access XP a 2003.
V roce 2010 představil Microsoft novou verzi MS Office 2010, kterou je možné pořídit ve 32-bitové i 64-bitové verzi. Tyto dvě verze jsou ale vzájemně nekompatibilní! V současnosti většina operačních systémů je plně 64-bitových, preferovány by proto měly být právě 64-bitové verze softwarových balíků, byť 32-bitové verze aplikací je možno na těchto systémech provozovat.
Bez ohledu na to, kterou verzi databáze zvolíte je potřeba, abyste tuto volbu drželi dále na všech počítačích, kde hodláte Vámi vytvořené databáze provozovat. Text uvedený v této kapitole je možné bez problémů aplikovat bez ohledu na verzi MS Access.

Kromě výše zmíněné distribuce MS Access je možné využít MS Access Runtime. Původně byl tento modul ve starších verzích MS Access (verze 2003 a starší), dodáván za úplatu jako součást MS Office Developer Tools, tedy sady nástrojů pro vývoj aplikací v MS Office. Vývojáři databází jej pak



Dostupnost MS Access
Studenti na VŠB-TU Ostrava mají dostupný MS Access v poslední stabilní verzi pro nekomerční použití v rámci služby Office 365 [2], dostupné z https://portal.office.com/Home. Do služby se přihlašujete pomocí uživatelského jména ve tvaru abc123@vsb.cz, kde abc123 je Vaše uživatelské jméno do systémů univerzity, na které jste zvyklí.
V rámci Office 365 máte k dispozici 5 licencí.

mohli distribuovat se svými databázemi na instalačních médiích. Od verze 2007 je MS Access Runtime dostupný ke stažení zdarma přímo z WWW stránek Microsoftu. Verzi Runtime modulu v poslední verzi (2016) je možné nalézt v odkazu [5].

MS Access Runtime je odlehčenou verzi MS Access, která může být distribuována spolu s databázemi v MS Access vytvořenými. Databáze jsou pak provozovatelné i na systémech, které nemají nainstalován plnohodnotný MS Access. Nevýhodou MS Access Runtime je nepřítomnost standardního GUI pro ovládání databáze – tvůrce databáze proto musí navrhnout vlastní GUI. Grafické uživatelské rozhraní je odvozeno od GUI MS Office a adaptace uživatele je tedy pohodlná a rychlá.

GUI
GUI se pro MS Access 2007 (podobně jako zbytek MS Office) zásadně změnilo. Pokus jste z nějakého důvodu nuceni pracovat se starší verzí s původním rozhraním, můžete nalézt návod na použití ve skriptech *Počítače a ochrana data – Návody pro cvičení* [8] v prvním vydání.
Od verze 2007 k zásadním změnám rozhraní již nedošlo - měli byste se proto zorientovat v obrázcích obrazovek bez problému i v případě, že použijete jinou verzi MS Access než která byla použita pro navržení těchto skript.

Spusťme tedy konečně MS Access a pusťme se do práce. Po spuštění MS Access se objeví úvodní obrazovka umožňující otevřít existující databázi nebo definovat databázi novou, vlastní (viz. obr. 2.1).

Obrázek 2.1: Úvodní obrazovka MS Access

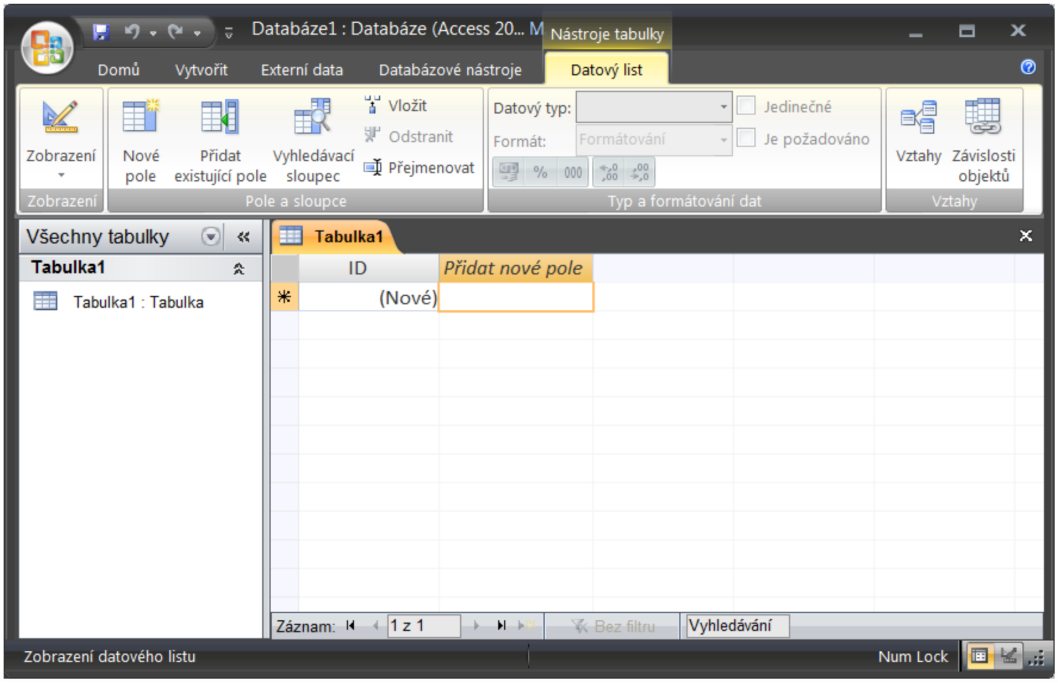
Poslední editované databáze naleznete v pravé části obrazovky. V prostředním pruhu, nahoře naleznete tlačítko prázdná databáze, která nám umožní definovat novou – prázdnou databázi. Kromě této možnosti zde jsou nabídnuty šablony databází, které sice vytvoří novou databázi, ale zároveň v ní nedefinují některé standardizované prvky (tabulky).

V rámci učení budeme pracovat s prázdnou databází. Pokud by se Vám v praxi stalo, že by Vašim požadavkům vyhovovala některá šablona, zvažte, zda pro Váš problém nebude existovat nějaké ucelené

softwarové řešení a zda se tedy použití databáze není možné vyhnout zcela. Šablony jsou totiž velmi specifické a obvykle pro řešení konkrétních úloh nevyhovují.

Vytvoříme si tedy 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. 2.3). Všimněte si, že na rozdíl od ostatních programů z rodiny MS Office, MS Access vyžaduje vytvoření databáze přímo na disku – není možné pracovat s databází pouze v paměti. Tato změna je způsobena tím, že některé operace s databází se automaticky ukládají bez nutnosti aktivního zásahu uživatele. To je pro práci s databází obvykle výhodné (tedy pokud jste si nevratně nesmazali části databáze). Každopádně s touto vlastností je potřeba počítat.



Obrázek 2.2: Základní rozhraní prázdné databáze MS Access

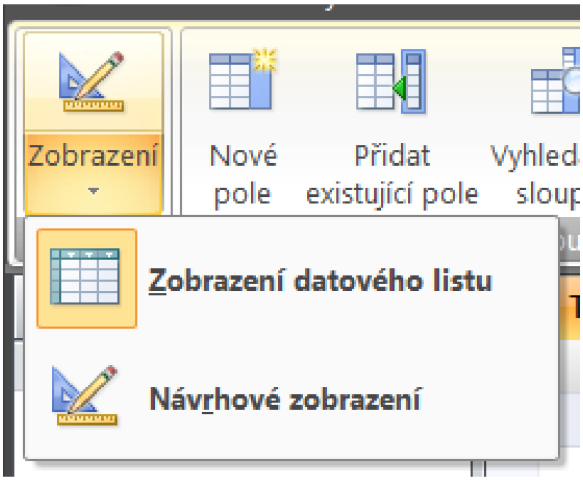
V nové databázi se automaticky vytvoří nová prázdná tabulka nazvaná Tabulka1.

2.2 Tabulky

Nová tabulka se zobrazuje v režimu datového listu, v rámci kterého můžeme definovat nové sloupce – z hlediska efektivity zadávání, je však tento způsob definice struktury tabulky neefektivní. V rámci definice struktury totiž musíme kromě samotných sloupců definovat jejich datové typy a také definovat primární klíč tabulky. Tyto činnosti v tomto pohledu na tabulku není možné provádět. Musíme se proto přepnout do návrhového režimu tabulky – viz. obr. 2.3.

Režim práce s objekty
Přepínání se mezi jednotlivými režimy práce objektů tvoří velmi podstatnou (a častou) součást práce s databází. Nástroj pro přepínání naleznete vždy na stejném místě, bez ohledu na typ objektu, se kterým budeme pracovat.

Budeme vyzváni k pojmenování tabulky, řekněme, že definici struktury databáze začneme tabulkou Pedagog – Pedagog proto vypíšeme do dialogového okna uložit jako. Okno návrhu se nám výrazně změní – viz. obr. 2.4.



Obrázek 2.3: Režimy práce s objektem tabulky

V horní části se definují jednotlivé sloupce tabulky a jejich datové typy. Ve spodní části okna definujeme další vlastnosti sloupce. Připomeňme si strukturu tabulky Predagog: #IČPedagog, jméno, příjmení, katedra.

IČPedagog je primárním klíčem, jeho datový typ bude odvozen z toho, jak budeme tento sloupec chápat – u tohoto IČ nás totiž může, ale za určitých okolností nemusí, zajímat význam tohoto čísla. Pokud jako IČ zavedeme číslo nějaké školní průkazky – pak může být datový formát buďto číslo nebo textový řetězec, pokud použijeme rodné číslo, nebo číslo sociálního pojištění, pak se bude jednat o textový řetězec, pokud chceme pouze rozlišit jednotlivé pedagogy, pak nám postačuje datový typ automatické číslo.

Automatické číslo funguje tak, že zadávání nového řádku do tabulky se najde nejvyšší použité číslo v daném sloupci a automaticky se navýší o 1. Řekněme tedy, že pro IČPedagog použijeme automatické číslo. Zároveň využijeme toho, že přednastavené ID má definované jak datový formát automatického čísla, tak je označen jako primární klíč, bude nám tedy stačit přepsat název (ID) na IČPedagog.

Jméno a příjmení budou očividně textové řetězce. Ve spodní části pro ně ale budeme muset definovat jejich délku ve vlastnosti velikost pole. Velikost přitom může pro tento datový typ být maximálně 255 znaků, pokud bychom potřebovali zadávat větší objemy textu lze pouzí datový typ memo umožňující uložit přibližně 64 000 znaků nebo objekt OLE, který je omezen pouze dostupnou pamětí. Datový typ objekt OLE je ale primárně určen pro uchovávání netextových dat, jako jsou například obrázky apod.

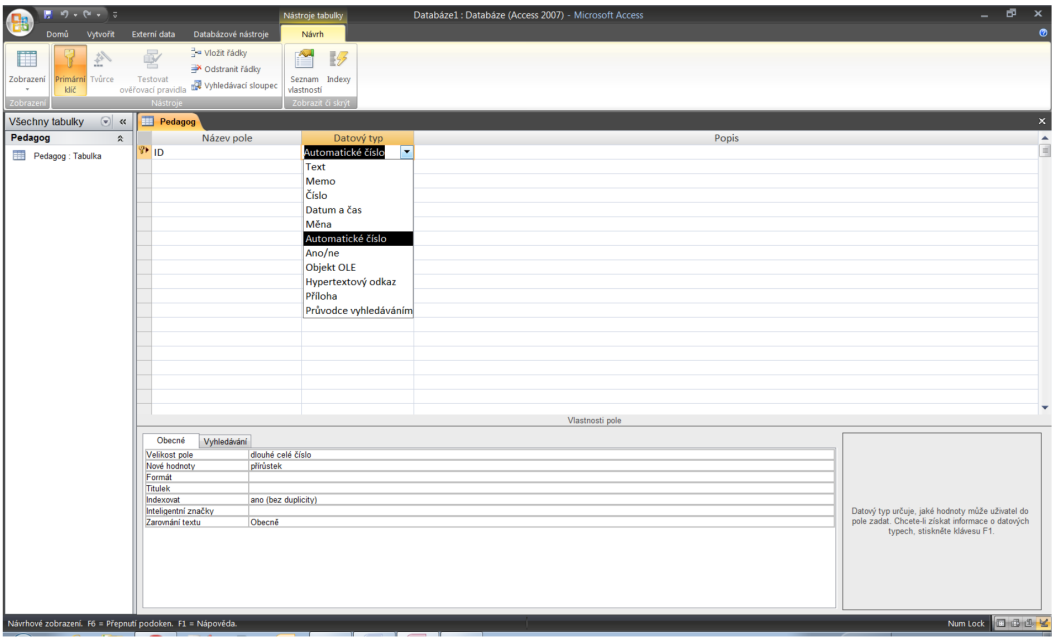
Další dvě vlastnosti, které u datového typu text často definujeme, jsou je nutno zadat a povolit nulovou délku. Tyto dvě vlastnosti souvisí s povinností uživatele při definování nového řádku zadat do daného sloupce údaje.

My pro sloupce jméno a příjmení zvolíme délku 50 znaků, což by nám mělo poskytnout dostatečný prostor i pro případná exotická jména, která mohou být delší.

Sloupec katedra bude sloužit pro napojení tabulky katedra – bude tedy sloužit jako tzv. cizí klíč, z tohoto důvodu musí být stejného datového typu, jako bude primární klíč tabulky katedra. Pokud uvážíme možnost identifikace katedry, jak je používán na VŠB, pak bude sloupec katedra tvořen třemi znaky. Protože prvním znakem může být nula, nelze použít datový typ číslo – katedra bude proto datového typu text o délce 3 znaky.

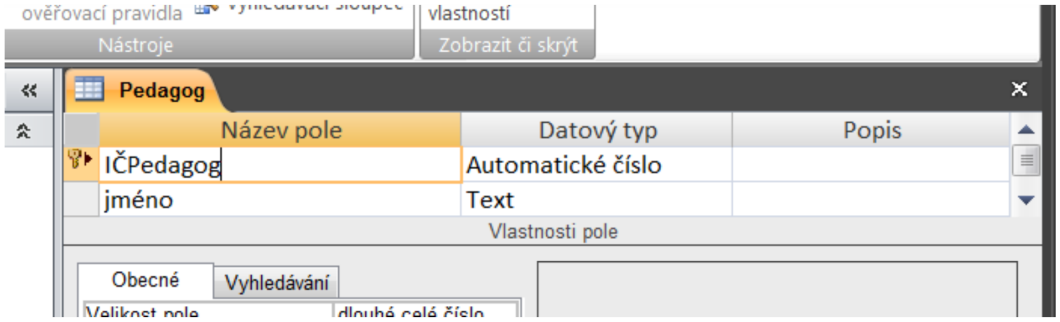
Shrňme tedy strukturu tabulky Pedagog:
IČPedagog – primární klíč, automatické číslo (dlouhé celé číslo)
Jméno – text, 50 znaků
Příjmení – text, 50 znaků
Katedra – text, 3 znaky

Výsledek našeho snažení uložíme a tabulku uzavřeme. Uzavření provedeme křížkem vedle záložky se jménem tabulky, nikoliv křížkem vpravo nahoře hlavního okna aplikace – tím bychom zavřeli celou databázi.



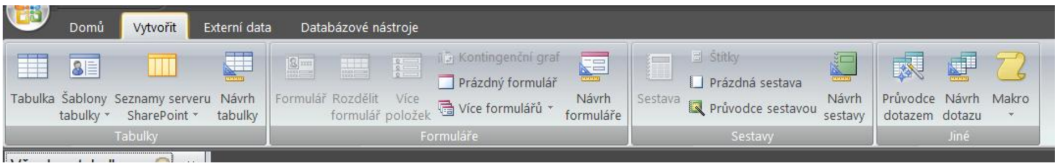
Obrázek 2.4: Návrhové zobrazení tabulky

Na druhou stranu díky automatickému ukládání řady operací je databáze v MS Access relativně odolná vůči náhodnému zavření.



Obrázek 2.5: Návrhové zobrazení tabulky

Další tabulky (nebo objekty jiného typu můžeme vytvořit z hlavního panelu nástrojů – záložka vytvořit, viz. obr. 2.6. Úplně vlevo pak je volba tabulka. Právě tato volba umožňuje vytváření nových tabulek.



Obrázek 2.6: Panel nástrojů vytvořit

Vytvořme si další dvě základní tabulky v naší databázi: předmět a student. Jelikož budeme postupovat analogicky k tvorbě tabulky pedagog, vypíši zde pouze názvy sloupců s jejich definicemi bez dalšího komentáře.

Předmět:

#IČPředmět – primární klíč, automatické číslo

Jméno – text, 255 znaků

Kredity – celé číslo
hodincv – celé číslo
hodinpř – celé číslo

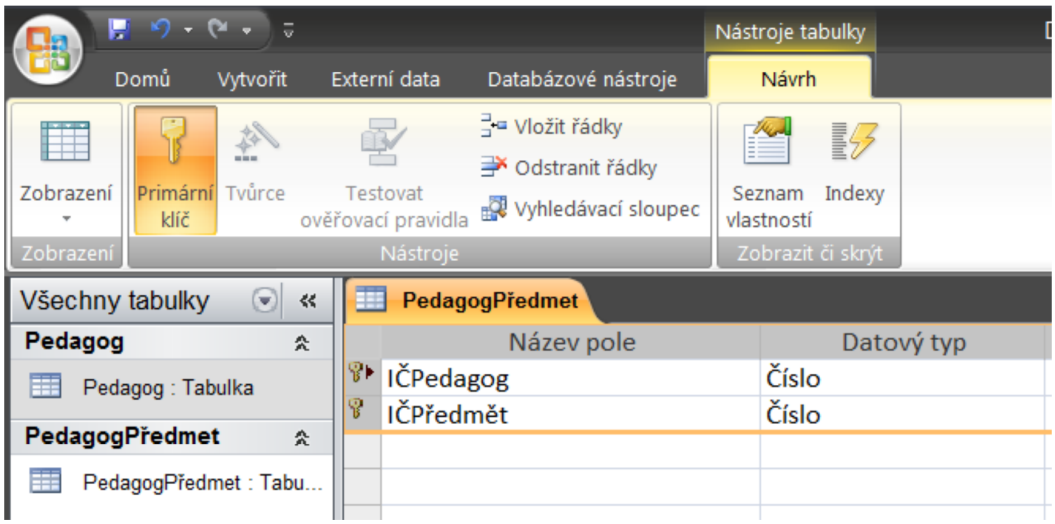
Student:
#IČStudent – primární klíč, automatické číslo
Jméno – text (délka 50 znaků)
Příjmení – text (délka 50 znaků)

Tyto tři základní tabulky budeme potřebovat propojit. Z teoretického výkladu o databázích již víme, že k tomuto účelu si budeme muset vytvořit pracovní tabulky StudentPředmět a PedagogPředmět.

StudentPředmět
#IČStudent – primární klíč, dlouhé celé číslo
#IČPředmět – primární klíč, dlouhé celé číslo

PedagogPředmět
#IČPedagog – primární klíč, dlouhé celé číslo
#IČPředmět – primární klíč, dlouhé celé číslo

V případě pracovních tabulek by ale nějaký výklad byl více než žádoucí. S těmito tabulkami se totiž pojí dva zádrhele. Prvním z nich je složený primární klíč. Ten vytváříme tak, že označíme oba sloupce, které budou tvořit budoucí složený primární klíč (např. IČPedagog a IČPředmět) a klikneme na ikonu klíče na záložce Návrh (viz. obr. 2.7).



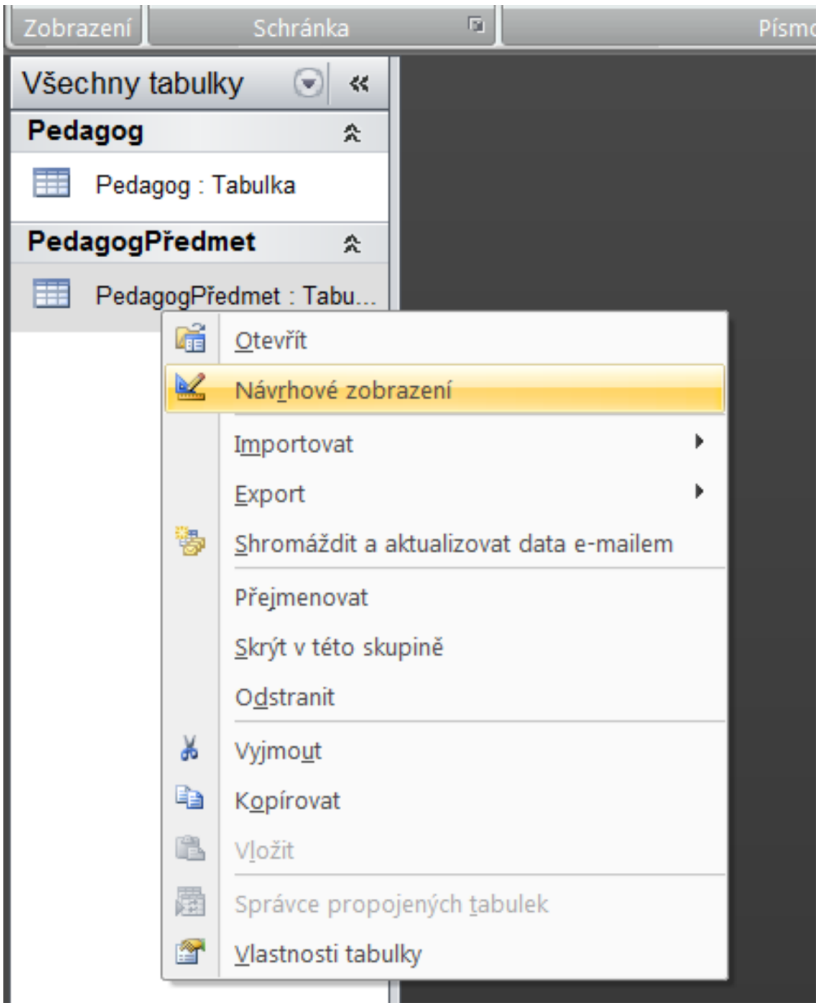
Obrázek 2.7: Panel nástrojů návrh

Oba sloupce se tak stanou složeným primárním klíčem. Postupné volení primárního klíče po jednom sloupci není možné, protože každá volba primárního klíče nejprve zruší existující nastavení primárního klíče a teprve potom označený sloupec nebo sloupce definuje jako primární klíč.

Všimněte si také, že sloupce v nových tabulkách mají formát dlouhého celého čísla, nikoliv automatického čísla. Důvodem je to, že v tabulkách pedagog, předmět, student nám na IČ až tak nezáleží – spokojíme se s faktem, že se přidělí unikátní identifikátor předmětu, pedagoga nebo studenta. V tabulkách StudentPředmět a PedagogPředmět nás ale zajímá již konkrétní předmět nebo pedagog nebo student a ti už mají přidělené určité konkrétní IČ. Automatické číslo proto není možné použít.

Analogicky budeme definovat i tabulku StudentPředmět.

Co dělat v případě, že již máme definované tabulky, ale později zjistíme, že jejich struktura přesně neodpovídá našim požadavkům. Bylo by samozřejmě nesmyslné, pokud by takovou editaci MS Access neumožňoval. Do editačního režimu se můžeme dostat kliknutím pravým tlačítkem na tabulce (v seznamu tabulek – viz. obr. 2.8) a volbou návrhového zobrazení.



Obrázek 2.8: Přepnutí do režimu návrhu

Druhou možností je kliknutí levým tlačítkem na tabulce (označení tabulky) a volba režimu návrhu na hlavním panelu nástrojů vlevo nahoře (jako při vytváření tabulek). Režim návrhu nemusí být dostupný za všech okolností. Například pokud budeme mít otevřen formulář, který zpracovává data z tabulky, kterou hodláme strukturálně měnit – tato změna nebude možná, dokud neuzavřeme formulář (toto platí i opačně).

Posledním režimem práce s tabulkou je režim zadávání dat. 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.

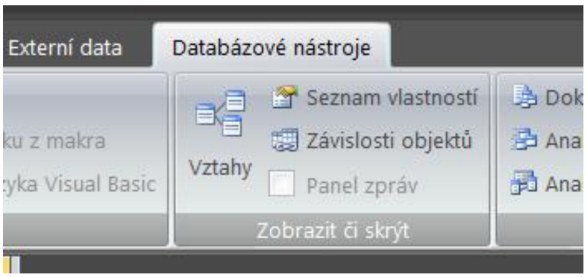
Osobně zadávání dat přímo do tabulek nedoporučuji – je obvykle lepší si nad tabulkami vytvořit formuláře, tak ať uživatel má lepší možnosti manipulace s daty včetně ošetření navazujících údajů. Formuláři se budeme zabývat později.

Kontrolní příklad
Upravte stávající strukturu tabulek tak, aby byla schopna udržovat informace o vykonaných zkouškách a zápočtech jednotlivých studentů (nepřidávejte žádnou tabulku).

2.3 Relace

Máme nadefinovány základní tabulky, nyní se pokusíme definovat relace – tedy vztahy mezi nimi. Před započetím 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 přikládána – obsahuje kompletně vyřešený příklad, který v této publikaci společně řešíme.

Editor relací spustíme kliknutím na ikonu vztahy panelu nástrojů Databázové nástroje (viz. obr. 2.9).



Obrázek 2.9: Databázové nástroje

Zobrazí se dialogové okno pro specifikaci tabulek, které chceme pomocí relací propojit, viz. obr. 2.10.

Dialogové okno by mělo obsahovat všechny tabulky, které jsme v databázi vytvořili. Vybírat je můžeme po jednom anebo najednou. Vybrané tabulky zavedeme do relačního diagramu pomocí tlačítka přidat (nebo dvojitým kliknutím na vybrané tabulce).

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.

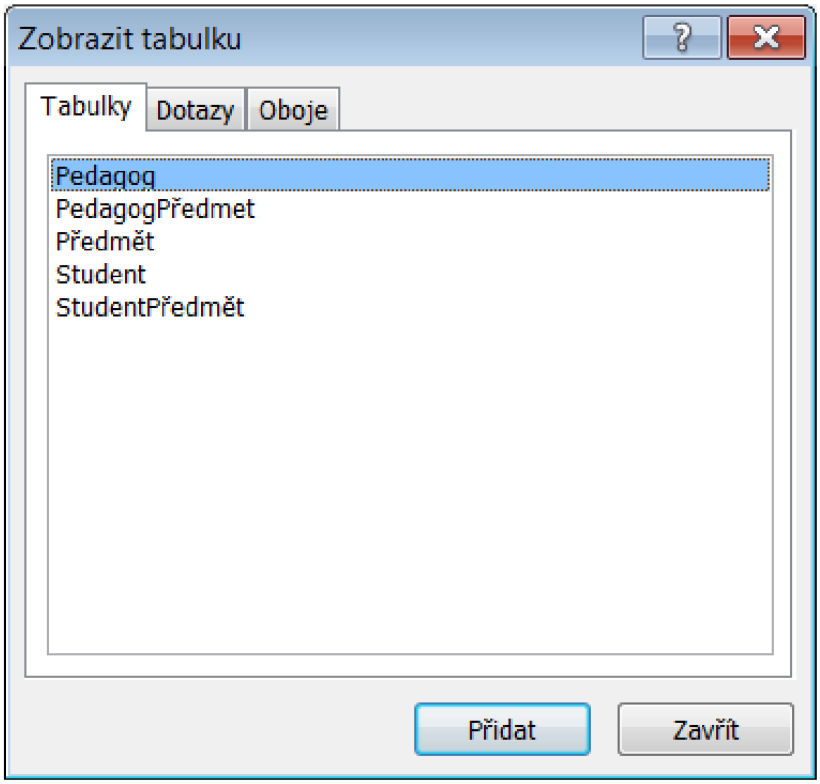
Všech pět tabulek rozmístníme v digramu dle našeho gusta a s ohledem na čitelnost výsledného diagramu. Přesun a změna velikosti se provádí jako u standardních oken Windows – chytnu za titulek a táhnu, chytnu za okraj a změním velikost. V případě, že v diagramu přebývají nějaké tabulky, nechtěnou tabulku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

Všimněte si, že primární klíče v tabulkách jsou označeny ikonou klíč. Nyní nám už pouze zbývá vyřešit otázku, jak vytvořit relaci – postup je prostý – chytnete pole ve zdrojové tabulce a táhnete na související pole v cílové tabulce. Poté co pustíte levé tlačítko myši se objeví dialogové okno pro definici vlastností relace (viz. obr. 2.11). Na směru tažení přitom nezáleží.

Zkusme postupovat společně. Uchopte pole IČStudent z tabulky Student a přetáhněte jej na pole IČStudent z tabulky StudentPředmět – objeví se dialogové okno podobné obr. 2.11. Stejného výsledku bychom dosáhli, kdybychom táhli IČStudent z tabulky StudentPředmět do tabulky Student.

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. 2.11 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 zaškrtávací pole Zajistit referenční integritu, které implicitně není zaškrtnuto. Co tato volba přesně znamená. V úvodní kapitole jsme sestavovali ERD model, také jsme



Obrázek 2.10: Výběr tabulek pro zavádění relací

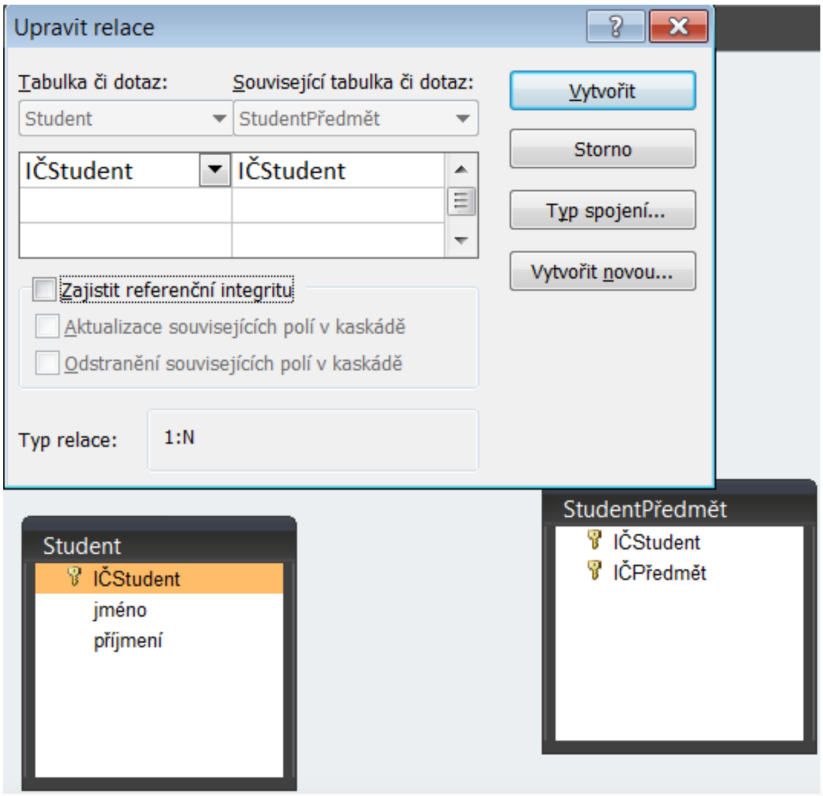
zmínili pravidla pro správnou tvorbu databází – z nichž jedno bych rád v tomto kontextu zopakoval – údaje jsou vedeny pouze na jednom místě a k ostatním údajům jsou vztaženy prostřednictvím relací.

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 by se ukázalo, že řadu předmětů již „absolvoval“ – integrita dat v naší databázi by byla narušena.

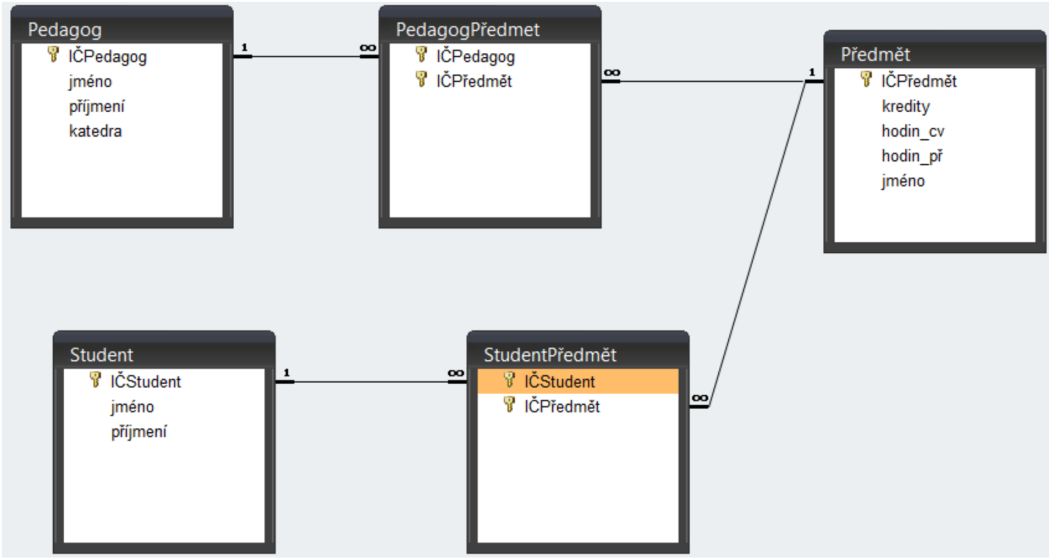
Aby bylo možné se těmto chybám vyvarovat, je nutné aktualizovat související pole v dalších tabulkách. MS Access tyto úkoly zvládá automaticky, pokud se mu řekneme, aby to automaticky dělal, tedy zaškrtneme políčko zajistit referenční integritu a potom aktualizace souvisejících polí v kaskádě a odstranění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

Až budeme spokojeni s nastavením, klikneme na tlačítko vytvořit. Podobným způsobem nadefinujeme i ostatní relace. Výsledek by měl odpovídat obrázku 2.12.

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.



Obrázek 2.11: Definice vlastností relace



Obrázek 2.12: Relační diagram databáze

Povinnost relace
Zavedením referenční integrity jsme v podstatě zakotvili povinnost existence souvisejících záznamů (např. PedagogPředmět může obsahovat pouze existující pedagogy a předměty), někdy je ale taková „povinnost“ na závadu. Uvažujme např. tabulku Pedagog a jeho katedru – co když máme takové pedagogy, kteří nepřináležejí k žádné katedře?.
Pokud bychom v takovém případě nastavili referenční integritu pro relaci mezi tabulkami Pedagog a Katedra, museli bychom za každých okolností vyplnit v tabulce pedagog sloupec katedra (a ten by musel být obsažen v tabulce katedra). Pokud předpokládáme, že vazba mezi tabulkami by měla být „měkká“, referenční integritu nezavádíme (ponecháme relaci bez definované referenční integrity).

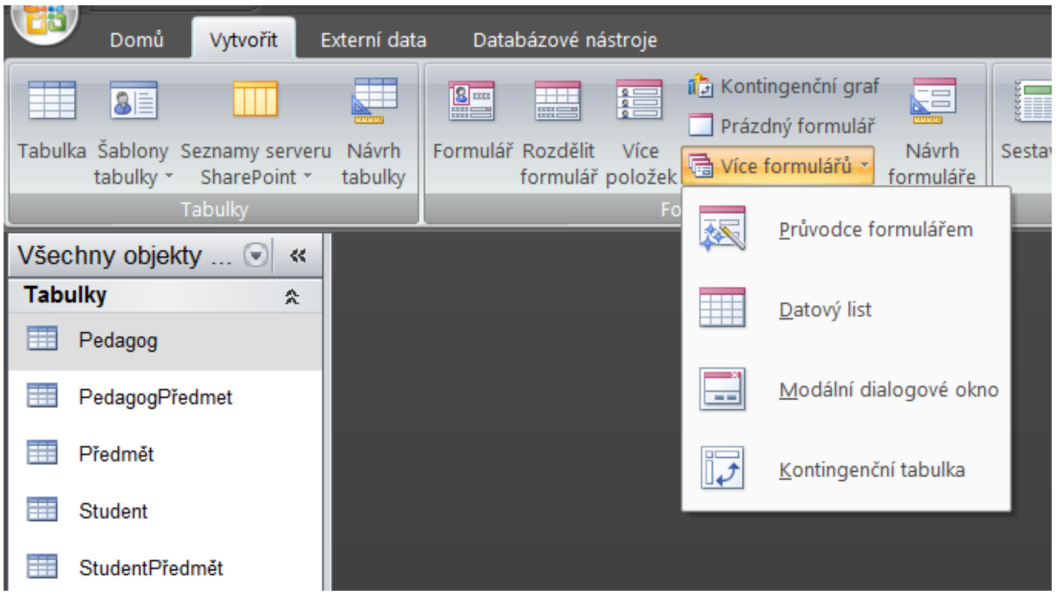
Kontrolní příklad
Doplňte stávající strukturu tabulek o zbývající tabulky, které jsme si navrhli v předchozích kapitolách a propojte je relacemi.

2.4 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.

Začněme s vytvářením formulářů prostřednictvím průvodce. Průvodce naleznete na panelu nástrojů na záložce Vytvořit → Více formulářů, viz. obr. 2.13.



Obrázek 2.13: Relační diagram databáze

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 (viz. obr. 2.14).

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.

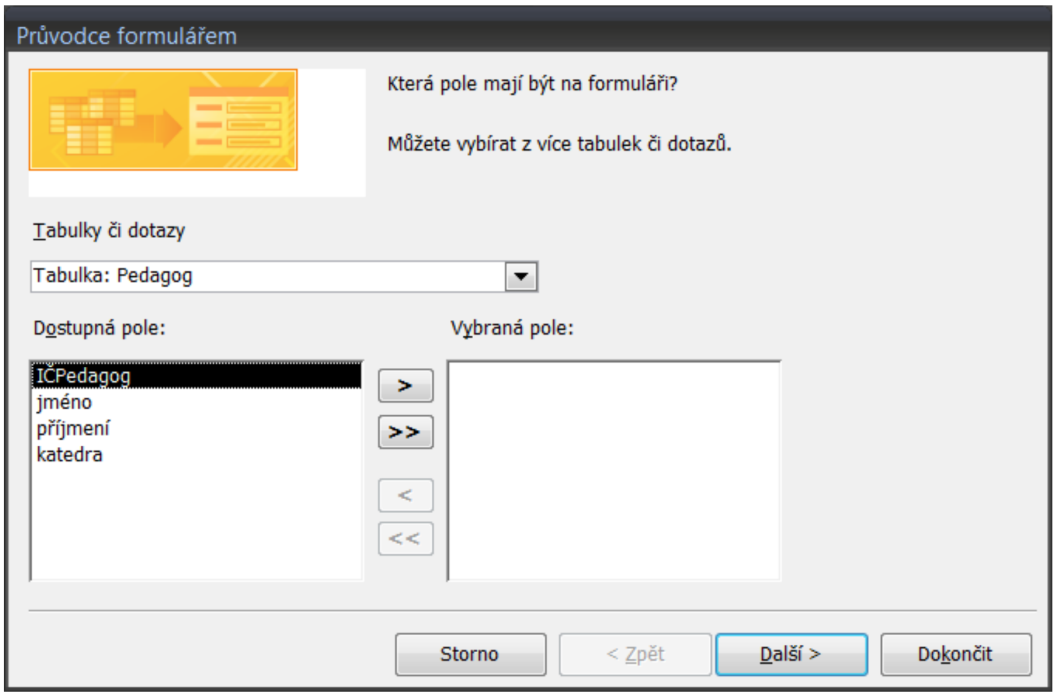
Objeví se okno definující tzv. rozvržení formuláře. Rozvržením formuláře rozumíme to, jak přesně bude formulář vypadat a jak se bude chovat. Rozvržení proto ovlivní, jestli se na jedné obrazovce formuláře může objevit jedna nebo více řádků tabulek, nad kterými formulář funguje.

Dostupná rozvržení 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 (kolik přesně závisí na finálním návrhu formuláře a rozlišení obrazovky).

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

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.



Obrázek 2.14: Průvodce tvorbou formuláře – volba polí

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. Náš výsledek by měl být podobný obr. 2.18.

Až se dostatečně pokocháte svým výtvorem, uzavřete formulář, ať můžeme zkusit něco trochu složitějšího, pokusíme se vytvořit formulář pedagogů, kterým bychom chtěli v rámci formuláře přiřazovat předměty, které učí.

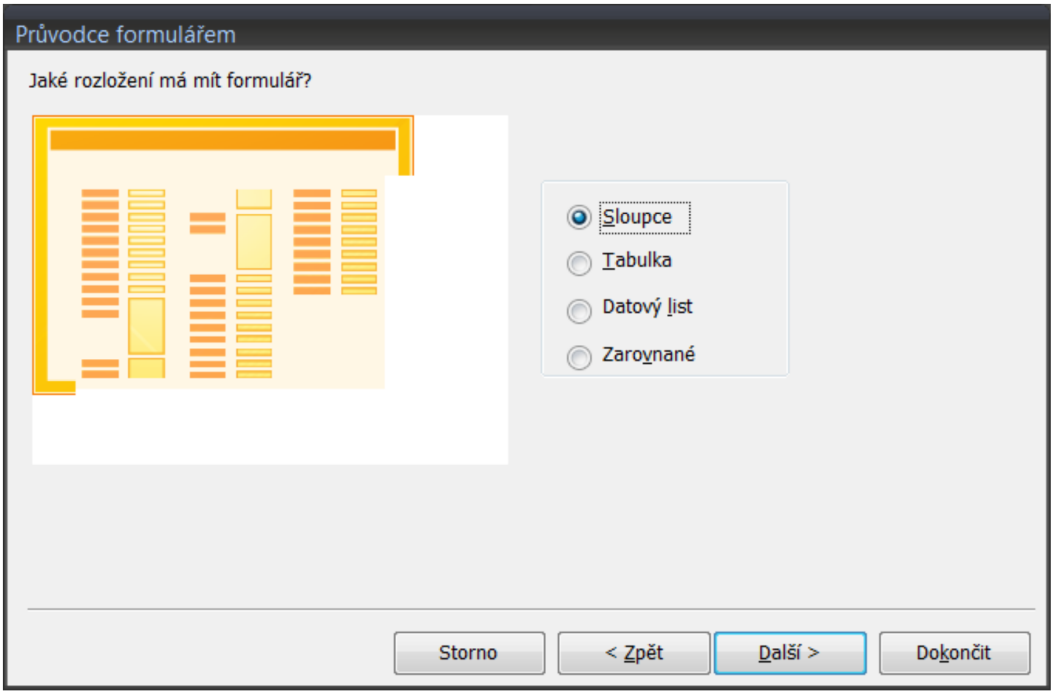
Spustíme znovu průvodce, znovu vybereme všechny pole tabulky Pedagog, navíc však vybereme i všechny pole tabulky PedagogPředmět. Všimněte si, že průvodce tvorbou formuláře se oproti předchozímu příkladu trochu změnil. Nyní máme k dispozici dialog pro volbu, podle čeho hodláme prohlížet data.

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.

Pokud bychom zvolili prohlížení podle tabulky PedagogPředmět propojení obou tabulek bylo provedeno na straně tabulky PedagogPředmět – tedy pro každou kombinaci předmětu a pedagoga se použijí vybraná pole obou tabulek. To je výrazný posun, protože při prohlížení pomocí tabulky pedagog, by se na formuláři zobrazil pouze 1x bez ohledu na to, kolik předmětů je k němu připojeno. Přesto oba pohledy za určitých okolností mohou mít své opodstatnění, takže je dobré o nich minimálně vědět.

Dále můžeme v tomto kroku ovlivnit, jakým způsobem se bude formulář chovat k „podřízeným“ datům (datům z tabulky PedagogPředmět). Tato data můžeme vložit buď do podformuláře, který se bude nacházet uvnitř hlavního formuláře, nebo budou pořízená data dostupná v samostatném formuláři, který bude z hlavního formuláře dostupný pomocí tlačítka.

V praxi volíme mezi těmito dvěma možnostmi v závislosti na množství dat, které máme na formu-



Obrázek 2.15: Výběr rozvržení formuláře

láři zobrazit. Data na formuláři by se měla pohodlně vejít na jednu obrazovku, bez nutnosti posunovat formulář směrem dolů nebo nahoru, popřípadě do stran. Pokud máme dostatek místa, můžeme tedy vložit podformulář, jinak volíme propojené formuláře.

V dalším kroku 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. 2.20.

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 nesetkali, jejich popis naleznete na obr. 2.21.

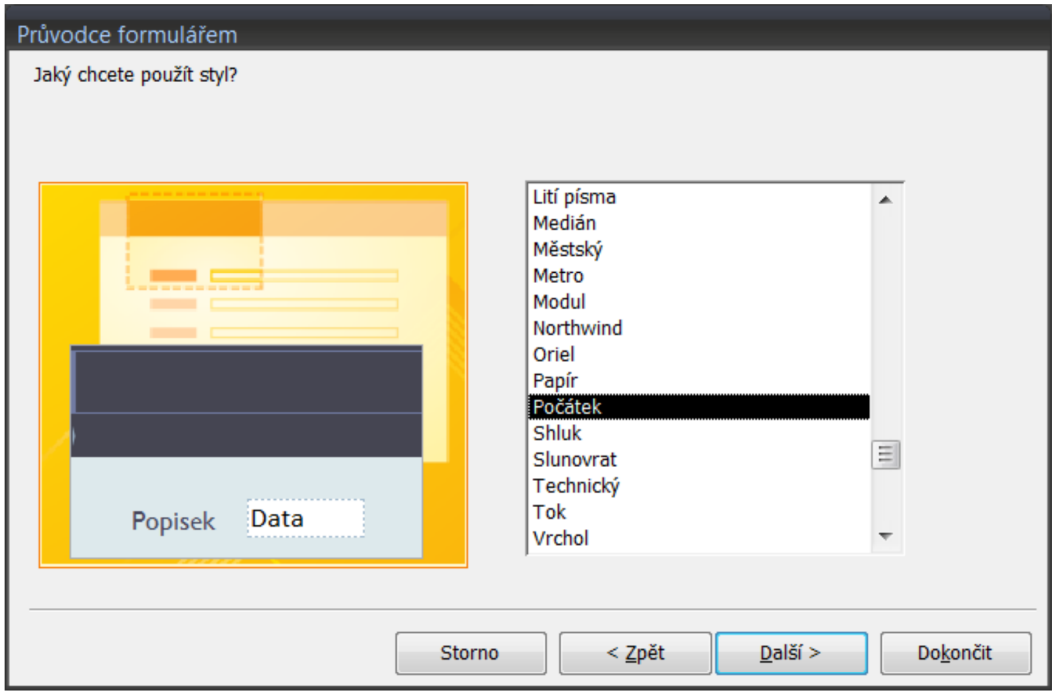
Kromě obligátního přepínání se mezi režimy práce s formuláři (o kterém jsme hovořili již při návrhu tabulky) zde máme ještě sekce písmo, mřížka, ovládací prvky a nástroje. Sekce písmo funguje velmi podobně, jako třeba ve Wordu nebo Excelu. Klikáním na jednotlivé ikony v této sekci tedy ovlivňujeme aktuální podobu vybraných prvků na formuláři z hlediska typu, barvy, velikosti, řezu písma apod.

Sekce mřížka obsahuje nástroje ovlivňující vzhled ohraničení jednotlivých prvků, které byly použity na formuláři. Největší množství nástrojů je obsaženo v sekci ovládací prvky. Tato obsahuje „stavební“ prvky formulářů, které můžete použít. Některé z těchto nástrojů si později společně vyzkoušíme, z prostorových důvodů to ale určitě nebudou všechny.

Konečně sekce nástroje obsahuje přepínače umožňující zobrazit seznam polí (přidat existující pole) a také vlastnosti jednotlivých prvků formuláře.

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.

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



Obrázek 2.16: Definice vizuálního stylu formuláře

vlastnosti (kliknutím na panel nástroje, sekce nástroje, ikona seznam vlastností).

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.

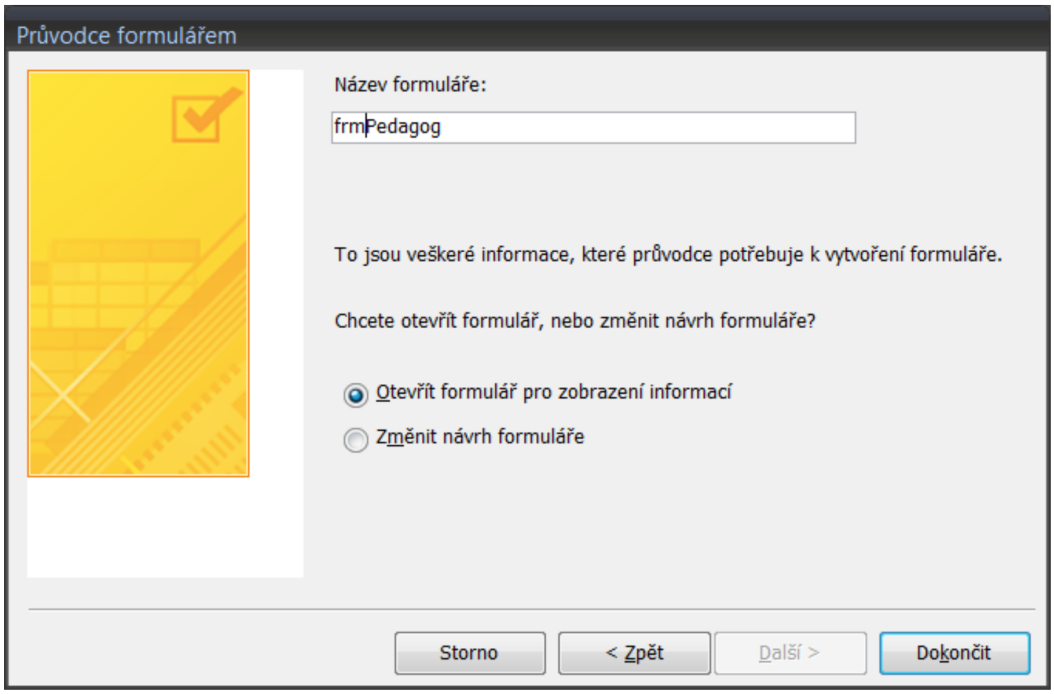
Námi vytvořený formuláře má k ideálnímu hodně daleko. Podívejme se na problémy, které brání v jeho efektivním využívání případnými koncovými uživateli.

1. Z vizuálního hlediska je nadpis formuláře chybný – měl by jej charakterizovat a být interpretovatelný i laikem.
2. Podformulář PedagogPředmět by mohl být umístněn trošku více vlevo a nahoře.
3. Popisek podformuláře je zbytečný.
4. V podformuláři je obsaženo redundantní pole IČPedagog, které je již obsaženo v hlavním formuláři – pro koncového uživatele nic nepřináší, a proto se jej budeme moci zbavit.
5. V podformuláři je předmět identifikován pomocí IČPředmět – to je nepřípustné, protože nelze očekávat, že koncový uživatel si bude pamatovat identifikační číslo každého předmětu.

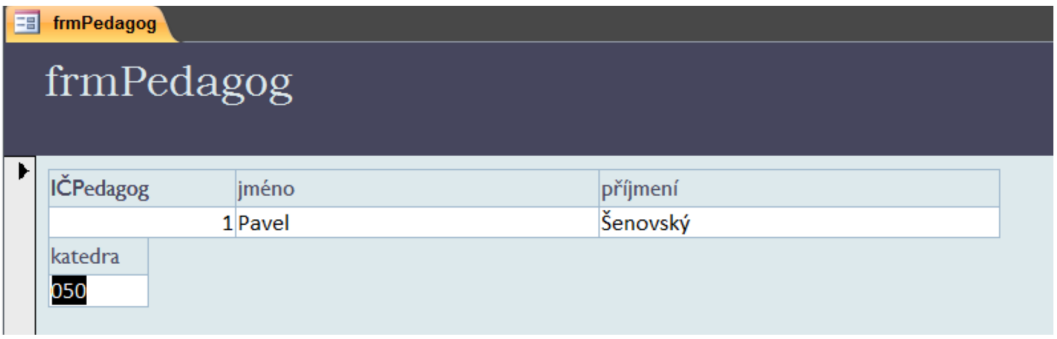
Zkusme postupně odstranit problémy. Nejjednodušší je změna nadpisu (nebo obecně jakýchkoliv popisků). Popisky jsou statické texty umístěné na formuláři (nebo v sestavě). Můžeme je tedy bez obav měnit k obrazu svému, aniž bychom ohrozili fungování samotné databáze. Editaci provedeme jednoduše tak, že kliknutím levým tlačítkem označíme a opětovným kliknutím levým tlačítkem se přepneme do editačního režimu. Zde můžeme měnit obsah popisku podobně jako v textovém procesoru. Řekněme, že jako nadpis použijeme Pedagog.

Přesun a odebrání popisku podformuláře provedeme v jednom kroku. Nejprve provedeme odebrání popisku. Od verze Access 2007 je ale popisek a k němu přináležející prvek formuláře (podformulář, textové pole apod.) pevně spojeno. Prosté označení a smazání (stisknutí klávesy delete) povede tedy k tomu, že dojde ke smazání jak popisku, tak podformuláře, což je v našem případě nežádoucí. Z tohoto důvodu budeme muset nejprve oba prvky rozpojit. To je možné provést z kontextového menu popisku a volbou Rozložení – odebrat (viz. obr. 2.24). Kontextové menu zobrazíme kliknutím na prvku formuláře pravým tlačítkem.

K rozpojování je potřeba dodat, že i dvojice popisek textové pole jsou na hlavním formuláři vzájemně spojeni do skupiny. To umožňuje, abychom velmi rychle a pohodlně dle potřeb měnili pořadí polí na formuláři, aniž by se formulář „rozházel“. Toto chování ovšem nemusí být za všech okolností žádoucí – i v takových případech pak bude nutné odpojit prvek formuláře z jeho zakotvení.



Obrázek 2.17: Pojmenovávání formuláře



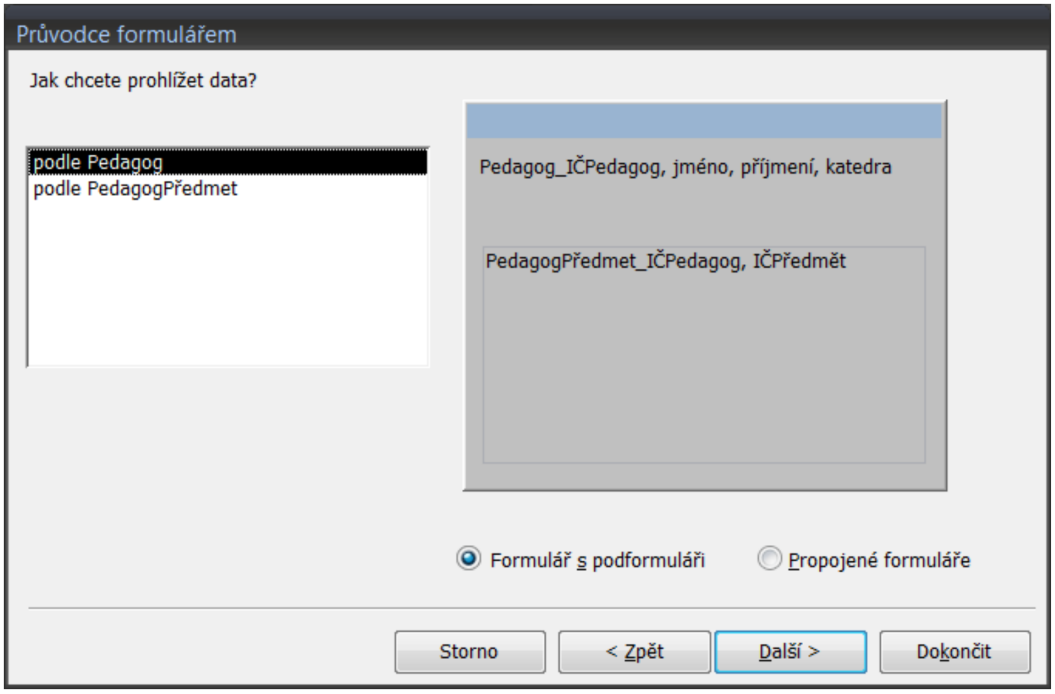
Obrázek 2.18: Výsledek tvorby formuláře prostřednictvím průvodce

Po odpojení popisku od podoformuláře můžeme bez obav popisek smazat a podformuláře přesuneme více doleva a nahoru. V podformuláři samotném pak jednoduše smažeme IČPedagog, smazání se provede včetně popisku a IČPředmět se automaticky přesune do uprázdněného místa. Zbývá nám tedy vytvoření pole se seznam, které by obsahovalo seznam předmětů, abychom si již nadále nemuseli pamatovat identifikační čísla jednotlivých předmětů.

Nyní se vrhneme na tvorbu pole se seznamem – tento prvek nalezneme v sadě nástrojů. Některé prvky sady nástrojů jsou popsány na obr. 2.25.

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!

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, technicky bychom aktualizaci mohli urychlit použitím makra – ale ta spadají do pokročilých témat, která nebudeme realizovat v tomto předmětu).



Obrázek 2.19: Definice podle čeho prohlížet data

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.

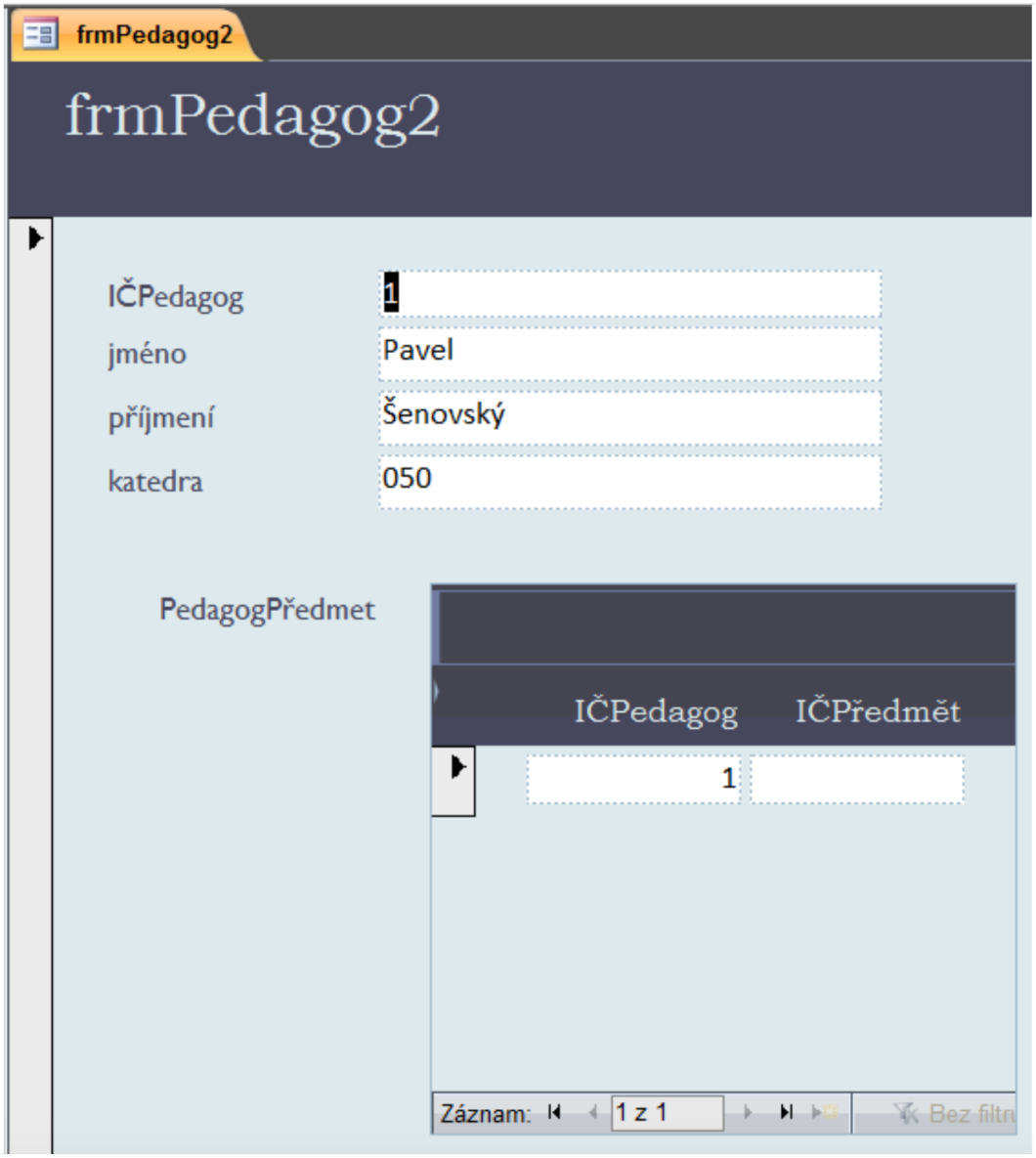
V dalším kroku vybereme tabulku Předmět a klikneme na další. Objeví se naše staré známé okno pro výběr polí, která chceme použít. Logicky musíme vybrat hodnotu, která je obsažena i v tabulce PedagogPředmět, jinak by tabulku Předmět nebylo jak zavést do formuláře. Dále vybereme Jméno, to je totiž ta položka,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

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.

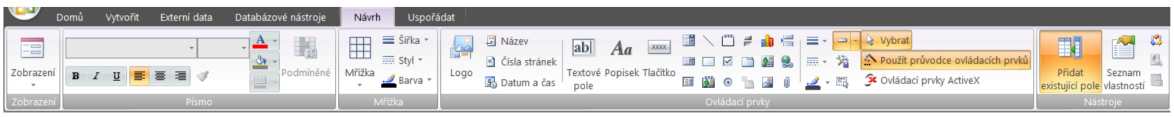
Po zaškrtnutí se při výběru nebude zobrazovat IČPředmět, ačkoliv bude nadále v rámci pole se seznamem vedeno – musí být, bez něj nejsme schopni napojit tento prvek do formuláře.

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.

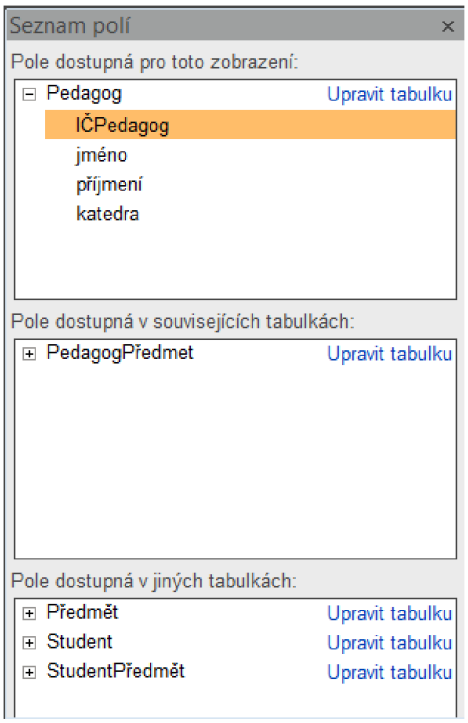
Dokončíme průvodce a v návrhovém režimu pak pouze doopravíme vzhled, tedy především odstraníme nyní již nadbytečné textové pole IČPředmět, to je nahrazeno nově vytvořeným polem se seznamem. Dále bychom mohli zmenšit pracovní plochu tak aby se nám vzhled podformuláře začal líbit. Výsledek naší práce by se měl blížit výsledku zobrazenému na obr. 2.29.



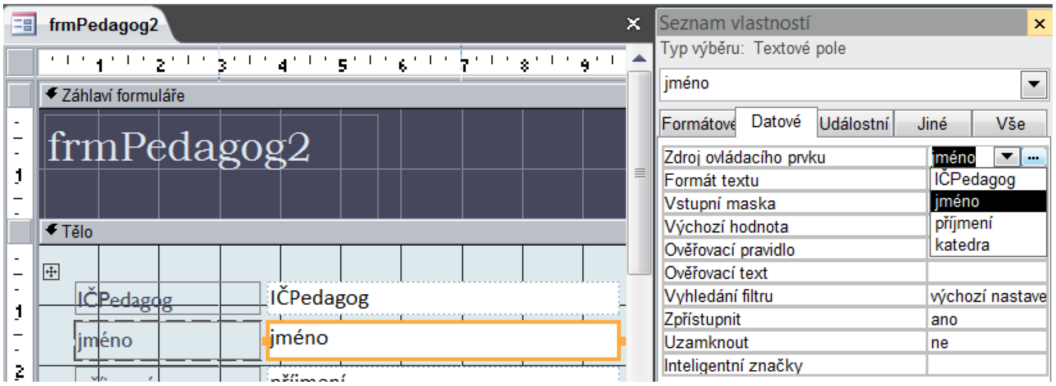
Obrázek 2.20: Formulář s podformulářem vytvořené prostřednictvím průvodce



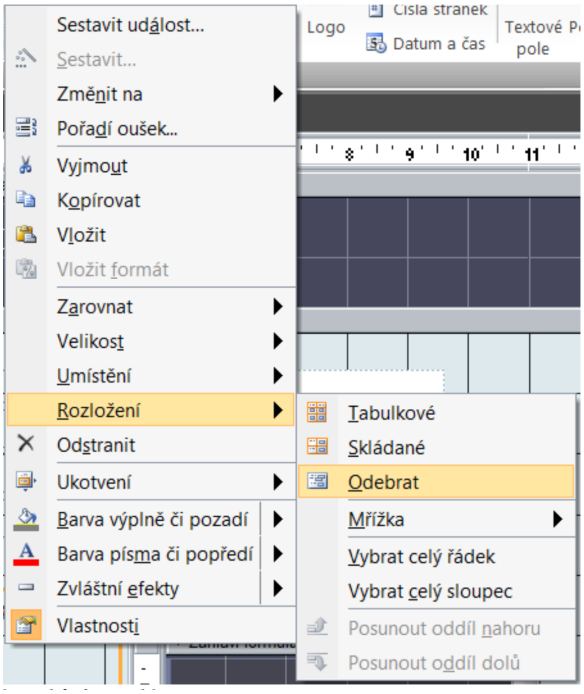
Obrázek 2.21: Panel nástrojů návrh formuláře



Obrázek 2.22: Seznam polí



Obrázek 2.23: Vlastnosti textového pole jméno



Obrázek 2.24: Rozpojení prvků formuláře

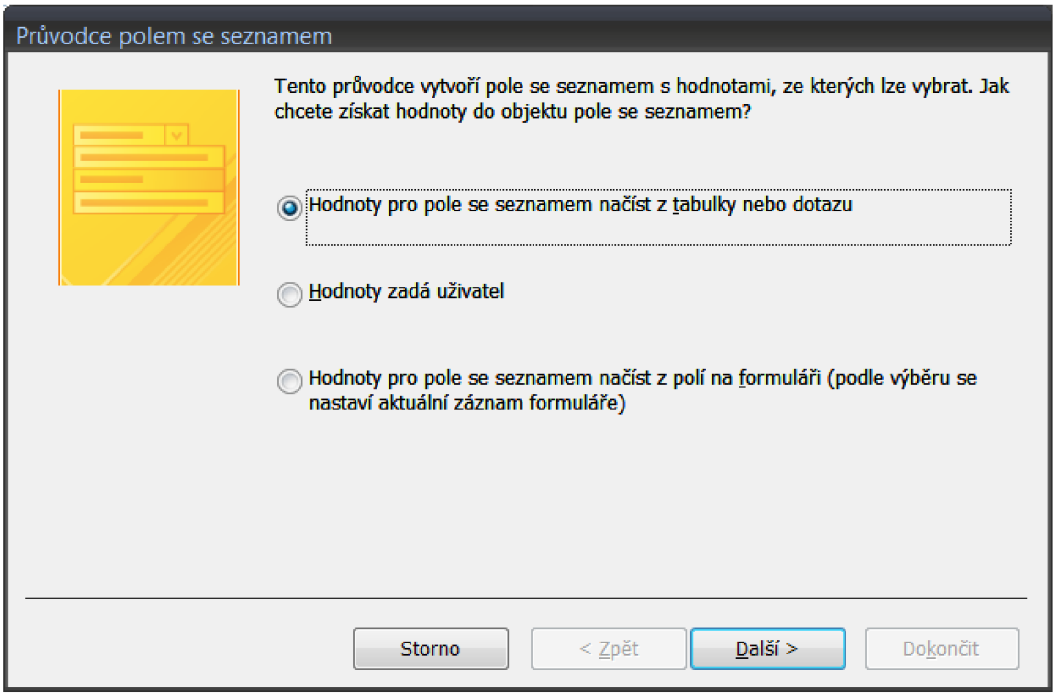
Kontrolní příklad, kterým si připravíme půdu pro realizaci pole se seznamem

Abychom zajistili použitelnost pole se seznamem, musíme zajistit, že koncový uživatel bude mít k dispozici grafické uživatelské rozhraní (GUI) pro zadávání a editaci obsahu pole se seznamem. V našem případě tedy potřebujeme formulář pro definici předmětů.

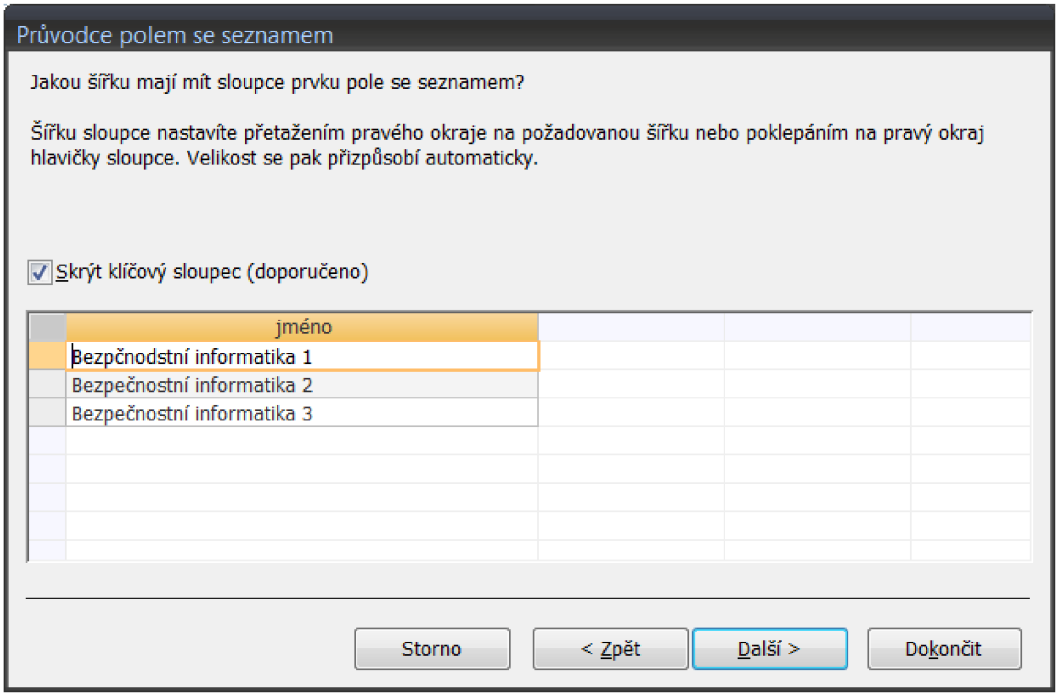
Nyní již máte dostatečné znalosti k tomu, abyste takový formulář vytvořili sami. Řekněme, že bude mít rozložení typu tabulka. Nezapomeňte po vytvoření formuláře zadat několik předmětů, ať máme s čím pracovat v poli se seznamem.

Obrázek 2.25: Panel nástrojů – ovládací prvky

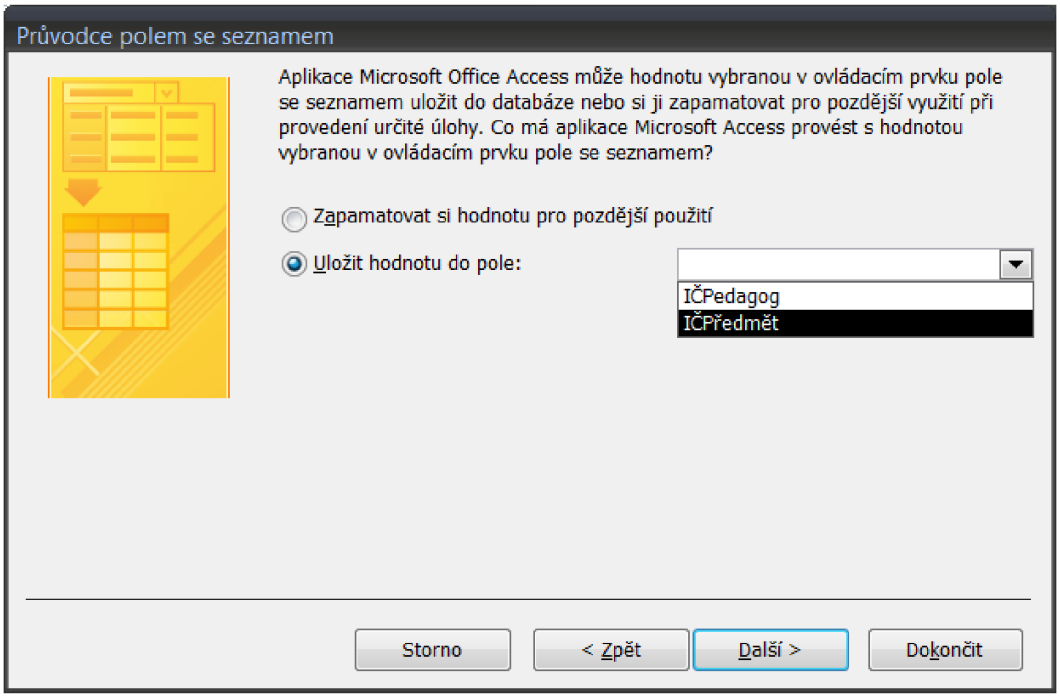
	vybrat objekty
	průvodci ovládacími prvky, při „zaškrtnutí“ tohoto nástroje se po vytvoření některých prvků na formuláři dojde ke spuštění průvodce nastavením vlastností tohoto prvku (ujistěte se, že tento nástroj je zaškrtnutý)
Aa	popisek. Jedná se o statický text, který obvykle nebývá navázán na tabulky a zůstává tak v rámci formuláře neměnný
ab|	textové pole, ve skutečnosti jde o dvojici prvků textové pole a k němu přináležející 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



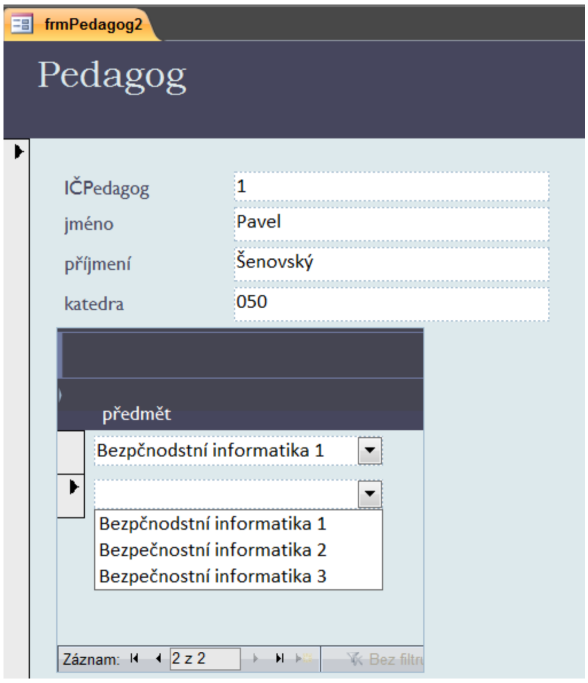
Obrázek 2.26: Průvodce pole se seznamem



Obrázek 2.27: Skrýt či neskrýt klíčový sloupec



Obrázek 2.28: Co udělat s vyplněnou hodnotou



Obrázek 2.29: Upravený podformulář s polem se seznamem

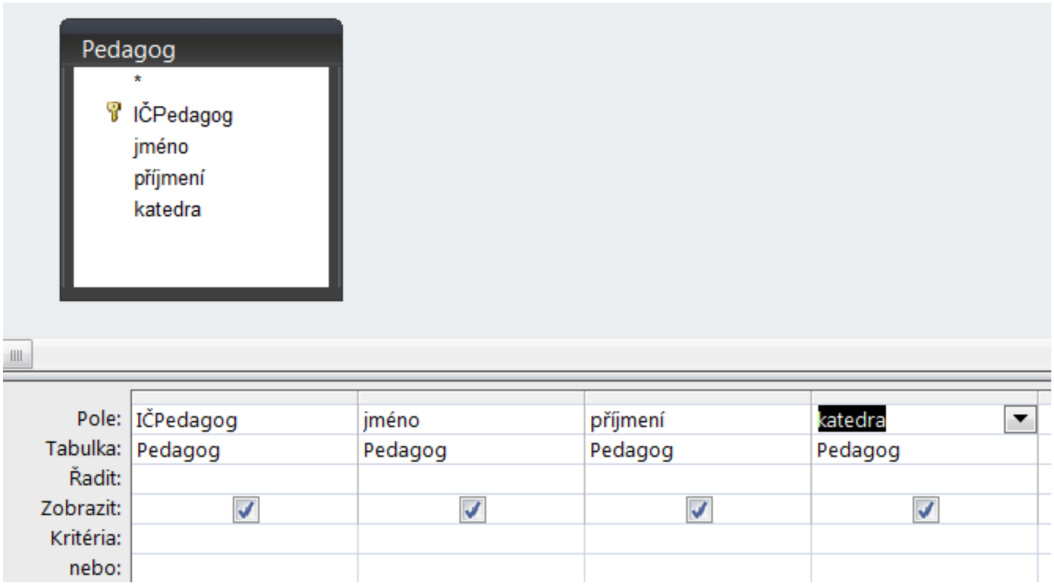
Kontrolní příklad
Vytvořte formulář pro práci se seznamem kateder. Přizpůsobte námi vytvořený formulář pedagogů k tomu, aby bylo možné zadávat katedry pomocí pole se seznamem.

2.5 Dotazy

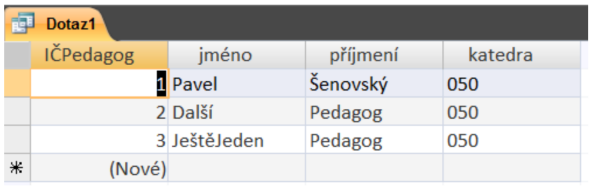
Předtím než se pustíme do vytváření dotazů bychom si mohli v krátkosti říci, co to dotazy jsou a k čemu je možné je využít. Dotazy, jak jejich název napovídá, umožňují vytvořit výběr dat z databáze i s možností nadefinovat si omezující podmínky. Dotazy používáme k získání dat, která pak můžeme použít 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 2.30 a 2.31.



Obrázek 2.30: Dotaz návrhové zobrazení



Obrázek 2.31: Dotaz – režim zobrazování údajů

Horní část okna návrhového režimu funguje podobným způsobem jako okno návrhu relací a to včetně ikony 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.

Zaškrtávací 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 kdekoliv 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 záludností 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 přepsat jako > 2 and < 2. Podobně je možné postupovat i u podmínek 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:

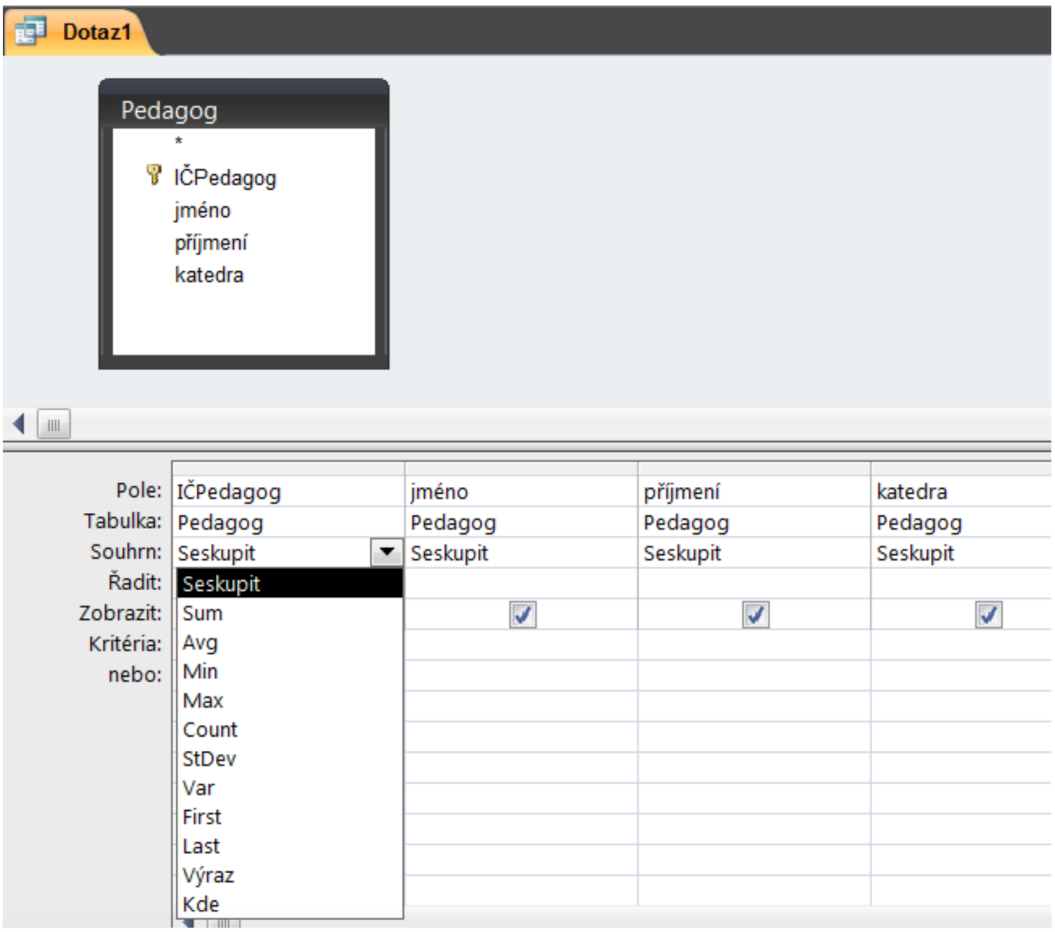
Like [Zadejte příjmení:]

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

Dalším typem dotazu jsou tzv. *sumární dotazy*. Výsledkem sumárních dotazů jsou vždy sumarizované údaje a to buď na jednom nebo více řádcích. Nezjišťujeme totiž záznamy ale charakteristiky skupin záznamů jako je minimu, 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. 2.32.

Kontrolní příklad

Definujte dotaz, který vypíše všechny předměty začínající uživatelem zadanými znaky.



Obrázek 2.32: Sumární dotaz

Kontrolní příklad
Definujte dotaz schopný spočítat průměrný, minimální a maximální počet bodů, které získal student v předmětech.

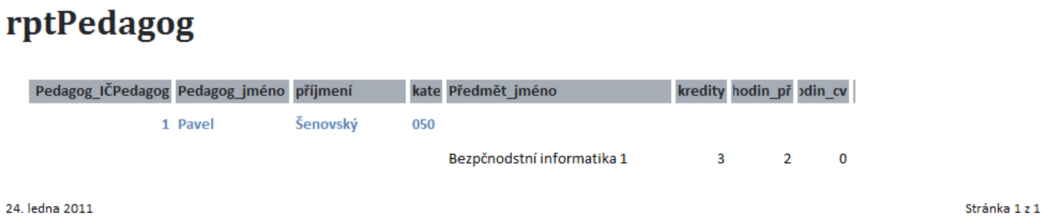
2.6 Sestavy

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 (obr. 2.33):

Výslednou sestavu by samozřejmě bylo možné dále upravovat.



Obrázek 2.33: Sestava z tabulek Pedagog, PedagogPředmět a Předmět

Kontrolní příklad
Vytvořte novou sestavu, která bude schopna zobrazit studijní výsledky studentů.

Kapitola 3

OpenOffice/LibreOffice Base

Průvodce studiem
V této kapitole se podíváme na využití alternativního databázového prostředku OpenOffice nebo LibreOffice Base.

Po prostudování této kapitoly budete
umět

- navrhnout strukturu databáze
- tvořit
  - formuláře
  - sestavy
  - dotazy

Čas nutný ke studiu
Rychlost prostudování závisí přímo na tom, za jakým účelem kapitolu procházíte. Pokud ji procházíte pouze jako doplněk k výkladu o MS Access (nebo obecně databázích) nemělo by Vám studium zabrat více než půl hodiny. Na druhou stranu, pokud hodláte používat LibreOffice Base v praxi (jako primární databázový prostředek), připravte se na hodiny perné práce.

LibrefOffice Base je součástí kancelářského balíku LibreOffice [4]. LibreOffice vznikl v roce 2010 oddělením od projektu OpenOffice. Důvodem pro toto oddělení bylo odkoupení společnosti Sun (správce OpenOffice) společností Oracle, jejíž záměry s kancelářským balíkem nebyly jasné. Do projektu LibreOffice se při vzniku projektu přesunula většina aktivních vývojářů. OpenOffice [3] v současnosti stále ještě existuje, jeho vývoj však pokračuje pod křídly Apache Software Foundation velmi pomalu.

Tato databáze Base, je stejně jako celý balík kancelářských aplikací Open Office nebo LibreOffice je open source produktem dostupným k bezplatnému použití na všech rozšířených platformách. Svojí koncepcí se Base inspiruje nejvíce z MS Access, nicméně v oblasti funkčnosti za tímto databázovým prostředkem výrazně zaostává.

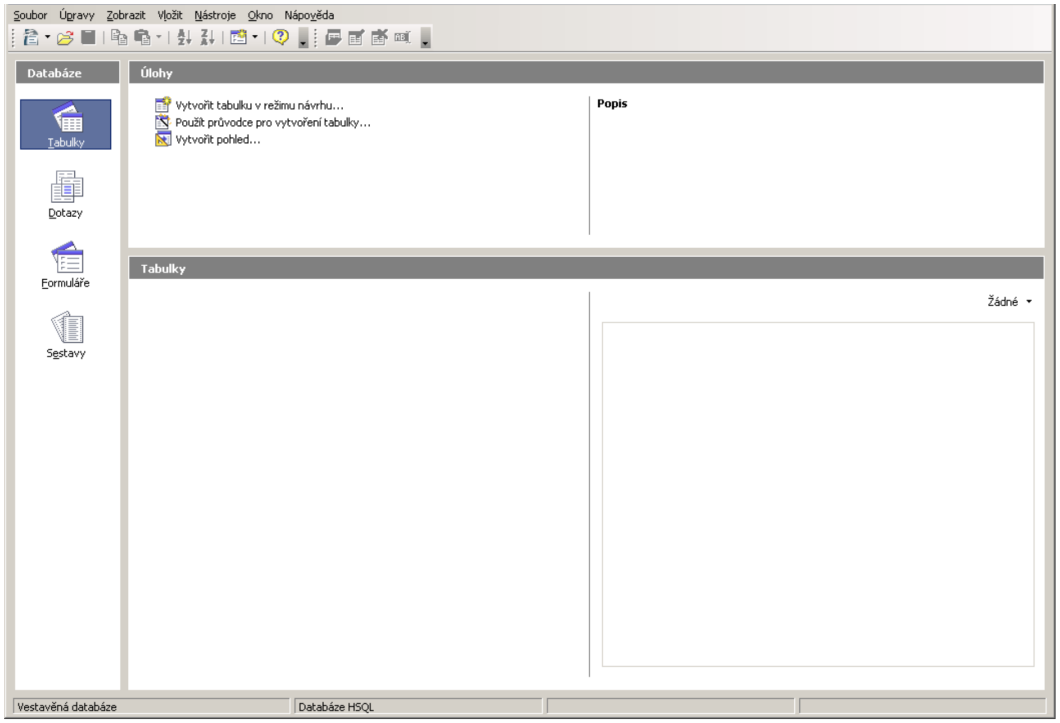
Pro základní seznámení s databázemi však tento produkt plně vyhovuje. Zkusme vytvořit nějakou databázi prakticky v tomto systému prakticky.

OpenOffice Base vs LibreOffice Base
Čistě z pohledu databázového mezi oběma produkty, možná trošku překvapivě, nejsou patrny rozdíly. V ostatních produktech balíků, ale tyto rozdíly patrné jsou a to dosti výrazné. Z tohoto důvodu doporučuji instalovat spíše LibreOffice než OpenOffice.

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. 3.1. Pro plnou práci s databází musí také být nainstalován Java Runtime Environment (JRE) [6]. Po instalaci je pak potřeba brát v úvahu, že společnost Oracle vydává pro Javu pravidelně balík oprav a je proto potřeba aktualizovat (což by mělo být prováděno u všech softwarů).

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.



Obrázek 3.1: 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. Implementace Basicu v balíku je velmi podobná Visual Basic for Applications (VBA) z MS Office, ale existují zde určité odlišnosti. V těchto skriptech bohužel není prostor se problematice programová pro tento (nebo jiný) kancelářský produkt věnovat.

Oproti MS Access se také poměrně výrazněji liší filosofie práce s jednotlivými objekty. Zatímco v MS Access je možné jednoduše se připínat mezi režimy návrhu a režimem editace dat, Base striktně rozlišuje mezi oběma režimy a uživatel musí prostřednictvím volby při spouštění 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 aplikaci Writer (textový editor) kancelářského balíku.

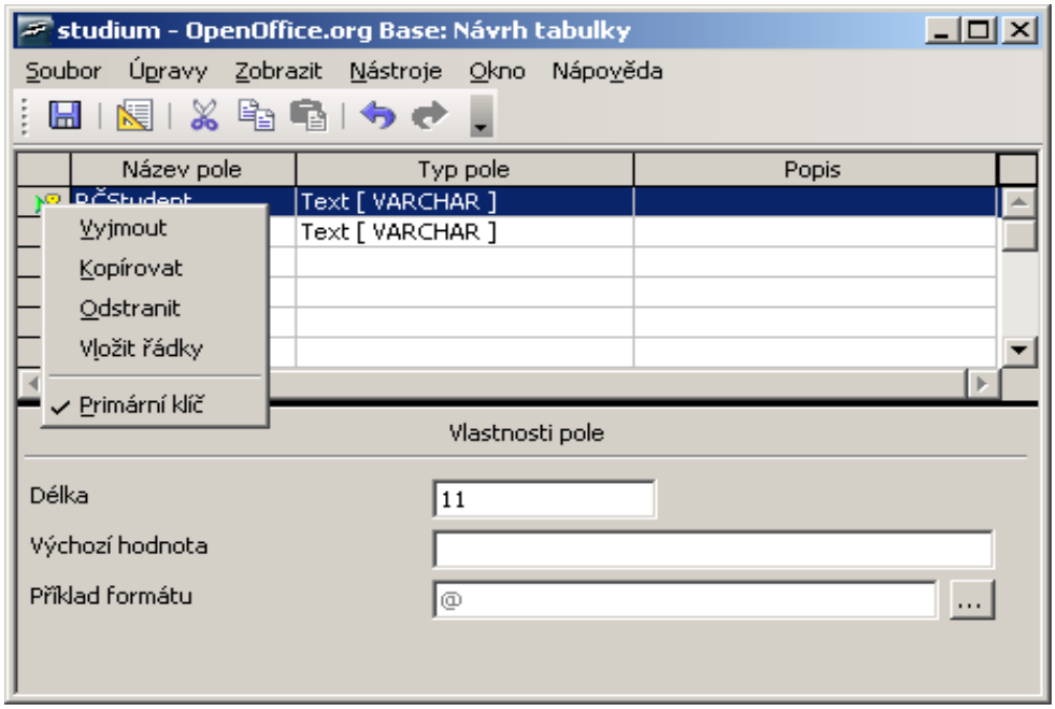
3.1 Tabulky

Jsou realizovány podobným způsobem jako v MS Access. Jediným podstatným rozdílem je realizace datových typů, které jsou odvozeny z Structured Query Language (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. 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. 3.2). Při změně primárního klíče dbejte na to, abyste nejprve odebrali stávající primární klíč a teprve potom definovali nový primární klíč. Současná verze totiž v některých případech prosté redefinice primárního klíče vyhazovala chyby.



Obrázek 3.2: Definice tabulky Student

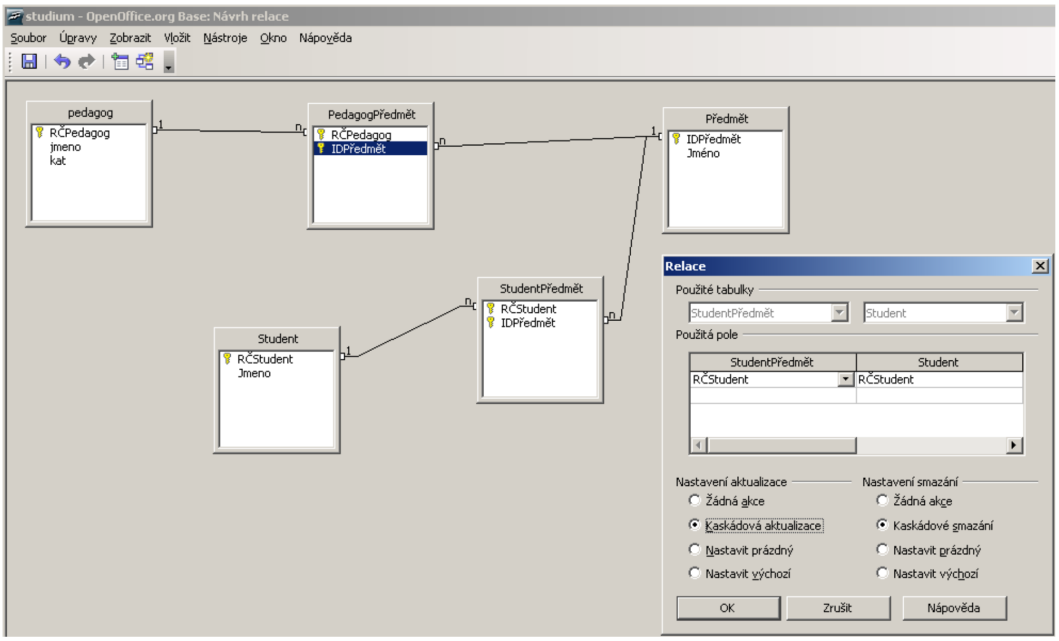
Kontrolní příklad
Navrhněte strukturu tabulek podle definice tabulek z kapitoly 2.2.

3.2 Relace

Opět, práce je velmi podobná MS Access. Base umožňuje editovat relace prostřednictvím volby menu Nástroje → Vztahy ... (viz. Obr. 3.3).

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 Base nedělá rozdíly mezi relací normální a relací, pro kterou je definována kaskádní aktualizace nebo výmaz.

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



Obrázek 3.3: Definice relací mezi tabulkami

Kontrolní příklad
Definujte relace pro Vámi zadané tabulky z předchozího kontrolního příkladu. Postupujte tak, aby pokud možno všechny tabulky byly zapojeny do celkové struktury báze dat.

3.3 Formuláře

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

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

Formulář jako takový je vytvářen v osmi krocích

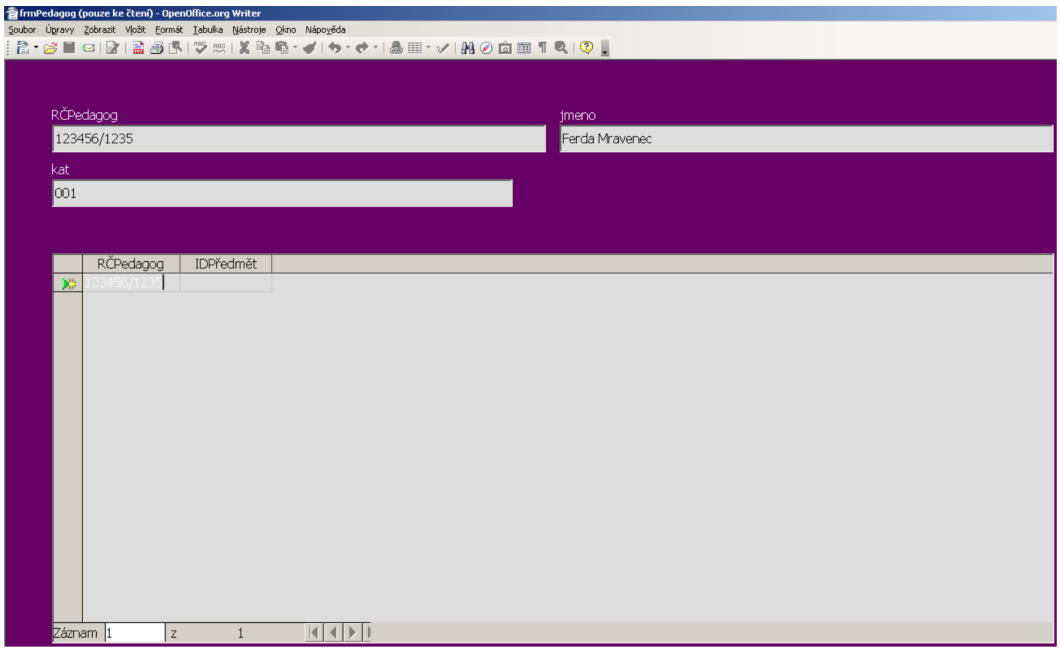
1. výběr polí (hlavní tabulka) umožňuje uživateli vybrat pole z hlavní tabulky
2. nastavit podformulář – nastavíme podřízenou tabulku máme-li takovou. Tabulku, na základě budeme „podformulář“ vytvářet přitom můžeme vybrat podle relace, která na něj vede z tabulky hlavní (tady se vyplácí pečlivý design 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 – poskytne 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. 3.4.

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 prohlížíte. 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* ...



Obrázek 3.4: 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.

Kontrolní příklad
Vytvořte formuláře pro práci se seznamem kateder a jednotlivými předměty.

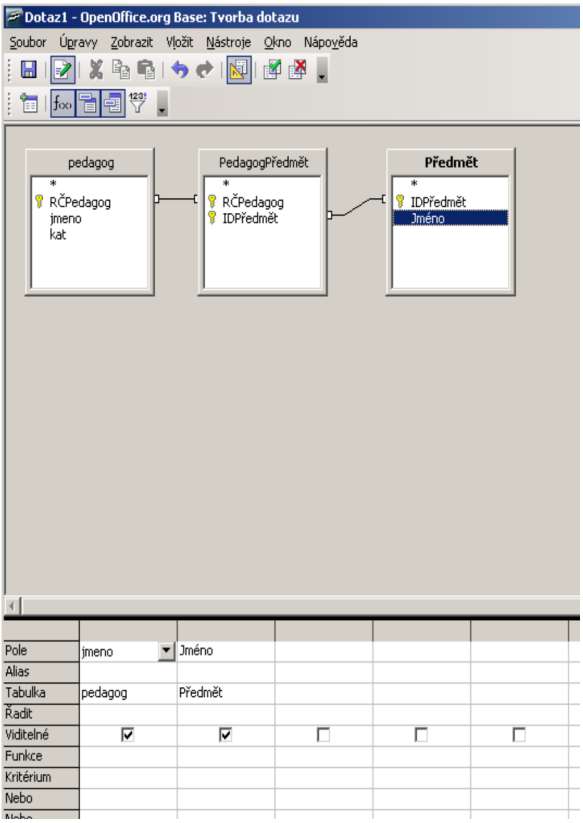
3.4 Dotazy

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. 3.5.

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. 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 Base nepodporuje. Zároveň se částečně liší také způsob vytváření tzv agregačních dotazů. 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 agragačním dotazem.

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



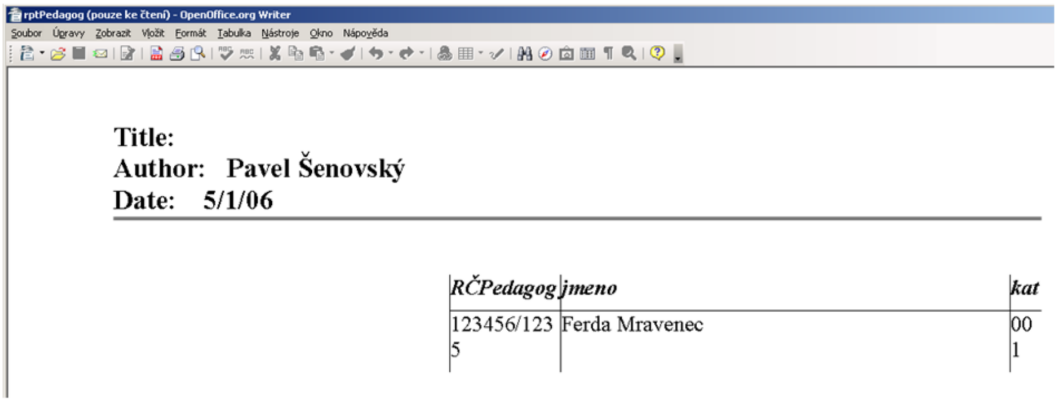
Obrázek 3.5: Tvorba dotazu

Kontrolní příklad
Definujte dotaz schopný spočítat průměrný, minimální a maximální počet bodů, které získal student v předmětech.

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. 3.6.

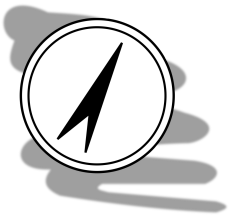


Obrázek 3.6: Příklad sestavy

Kontrolní příklad
Vytvořte novou sestavu, která bude schopna zobrazit studijní výsledky studentů.

Kapitola 4

Kexi



Průvodce studiem
Tato kapitola tvoří doplněk skript. Systém Kexi je malou, odlehčenou databází, která se hodí pro velmi malé projekty. Oproti nástrojům jako je MS Access nebo OpenOffice/LibreOffice Base. Neposkytuje lepší nástroje, ani výrazně vyšší komfort práce uživatele. Berte proto tento text jako doplňující informaci, k předchozím dvěma kapitolám.

Po prostudování této kapitoly budete
umět

- navrhnout strukturu databáze
- tvořit
  - formuláře
  - sestavy
  - dotazy

Čas nutný ke studiu
Databáze Kexi je databází jednoduchou. Pro pochopení způsobu práce proto budete potřebovat pouze minimum času.

Podobně jaké MS Access a Base je také databáze Kexi součástí kancelářského balíku, v tomto případě Calligra Suite [1]. Calliagra Suite je open source kancelářský balík dostupný především pro operační systémy na bázi Linux. Pro další operační systémy jsou dostupné některé jeho části, např. povedený grafický editor *Krita*.

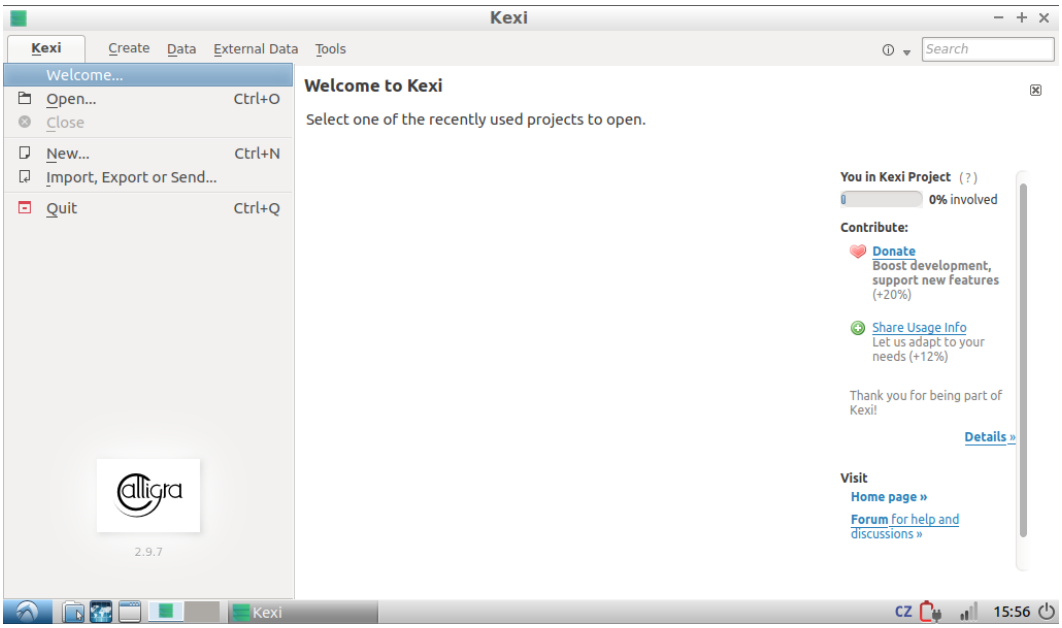
Databáze Kexi je určena pro rychlé práci s relativně malými objemy dat. Nejedená se tedy o produkt, který byl byl přímým konkurentem v dalších v těchto skriptech probíraných databázových systémů.

Díky značně omezené množině funkcí databáze, je ale patřičně omezeno ovládací rozhraní systému, které je velmi jednoduché a relativně intuitivní, viz obr. 4.1.

Podobně jako u MS Access a Base také Kexi ukládá všechny typy objektů na jedno místo - do souboru s příponou .kexi. Pro naši práci budeme muset vytvořit novou prázdnou databázi - kliknutím na *New* v obr. 4.1. Rozhraní je proti MS Access jednodušší, viz obr. 4.2.

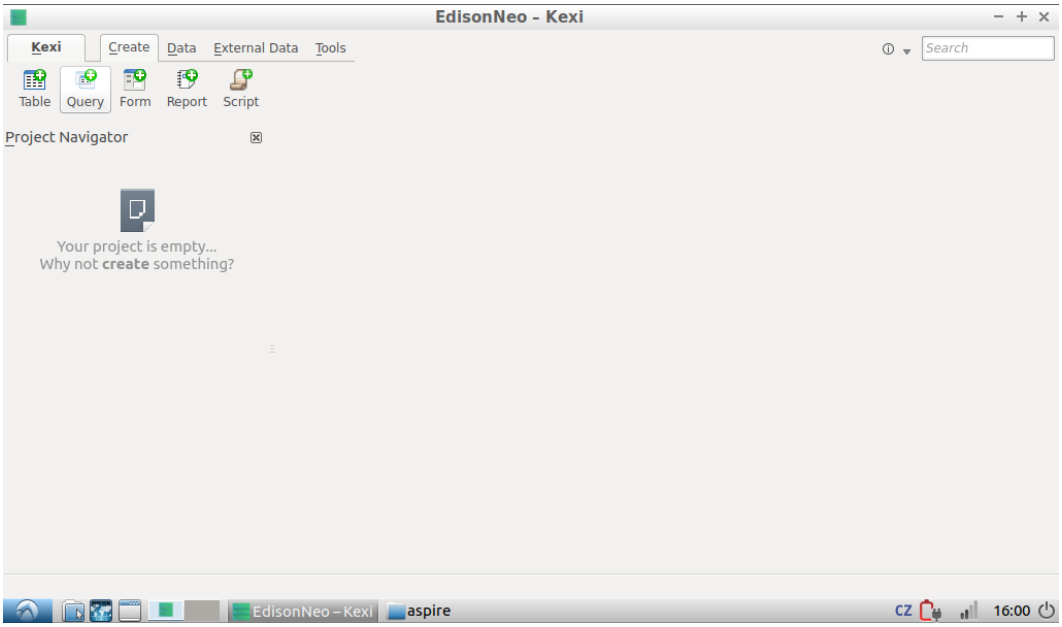
Databáze Kexi pracuje s následujícími typy objektů:

- tabulky
- dotazy
- formuláře
- sestavy
- skripty



Obrázek 4.1: Kexi úvodní obrazovka

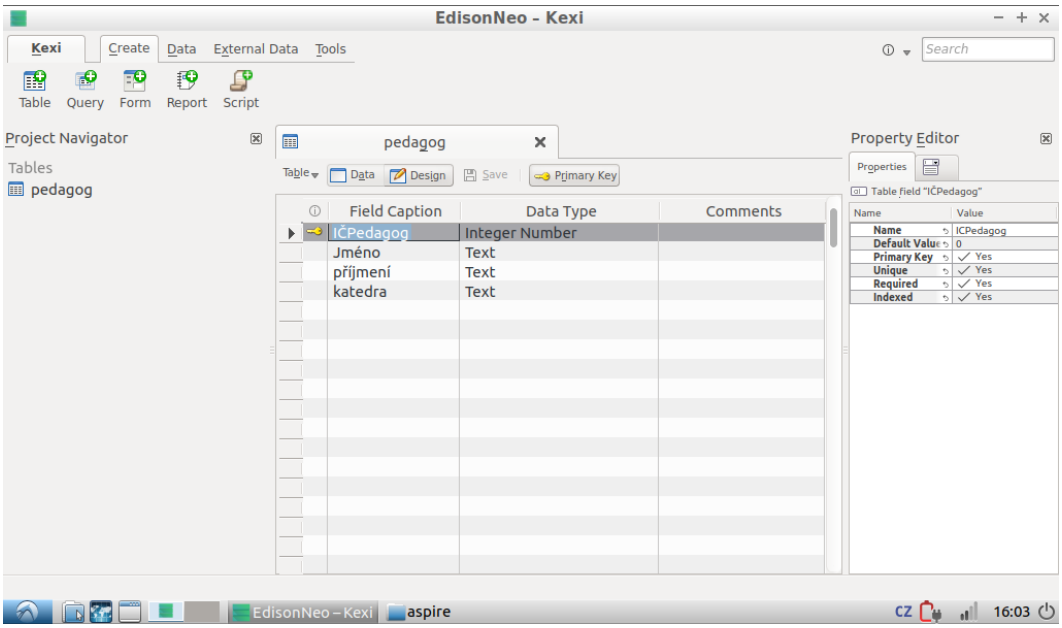
Většina z výše uvedených objektů by Vám již měla být známa - zastavím se nejprve u skriptů, které jsme neprobírali, ani tentokrát tomu nebude jinak. Skripty podporuje Kexi pouze v některých verzích. U připravované verze 3, která v v současnosti (leden 2017) ve fázi beta testování, však už podpora skriptů není přítomna. Databáze Kexi je skutečně databází velmi jednoduchou, takže, pokud je ke splnění úkolu vyžadováno použití skriptů, je nejspíše na čase přejít k nějakému pokročilejšímu databázovému systému.



Obrázek 4.2: Základní rozhraní databáze Kexi

4.1 Tabulky

Z pohledu tabulek jsou podporovány pouze jednoduché primární klíče. To pro náš účel představuje poměrně výrazný problém, protože vzorový příklad, se kterým pracujeme v těchto skriptech, potřebuje složený primární klíč ve dvou tabulkách. Před tím, než se pokusíme vyřešit tento problém, zkusme nadefinovat základní tabulky pedagog, předmět a student, viz obr. 4.3.



Obrázek 4.3: Definice tabulky - Kexi

Objekty různého typu lze vytvořit na záložce *Create* kliknutím na ikonu objektu s pluskem. Nově vytvořené objekty boudou pak dostupné pod těmito tlačítky.

Tabulku tedy vytvoříme kliknutím na tlačítko *Table* s plusem, a vyplníme způsobem, na jaký jsme zvyklí - tedy název pole a datový typ. Z pohledu datových typů jsme poměrně omezeni pouze na základní datové typy - k dispozici máme tedy pouze celá a desetinná čísla, datum a čas a text.

Kexi nepodporuje binární datové formáty a také dlouhý text (např. obdoba datového typu memo MS Access).

Podobně jako v případě MS Access i v případě Kexi můžeme nastavovat některé podrobnější vlastnosti datového pole. Tyto vlastnosti jsou dostupné ve vlastním dialogovém oknu editoru vlastností (property editor). Jak je vidět na obr. 4.3, je také nabídka vlastností pole tabulky omezená. Vlastnosti se tedy omezují na základní nastavení pole - např. délka textového políčka a také zdali se jedná o primární klíč, zda pole musí být vyplněno, zda má být indexováno apod.

Definici primárního klíče provedeme výběrem pole (např. kliknutím do něj) a kliknutím na tlačítko *Primary key*. Primární klíč v tabulce je označen ikonou klíče.

Mezi režimem definice tabulky a naplňováním tabulky se lze přepnou pomocí tlačítek *data* a *design*.

Jako tabulku pedagog z obr. 4.3 nadefinujeme také tabulky předmět a student. Tabulky, které nám sloužily pro propojení těchto základních tabulek (PedagogPředmět a StudentPředmět) je ale nutné nadefinovat lehce odlišným způsobem. Nemožnost vytváření složeného primárního klíče můžeme obejít vytvořením nového primárního klíče číselného typu a použití původních položek jako „cizích klíčů“. Struktura propojujících tabulek by tedy byla následující:

PedagogPředmět: #IDPedagogPředmet, IDPedagog, IDPředmět

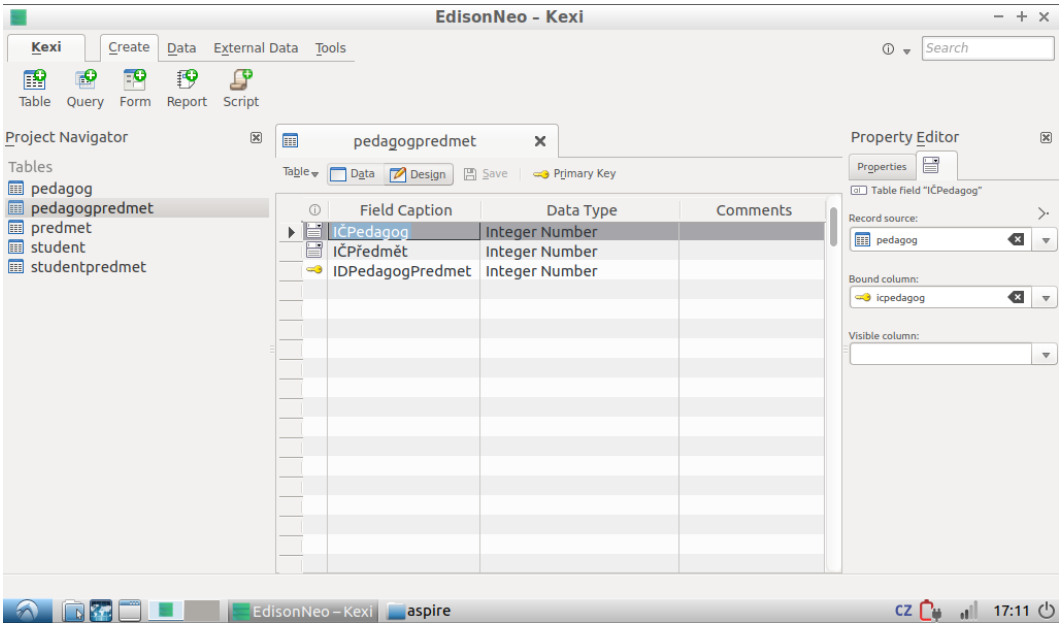
StudentPředmět: #IDStudentPředmět, IDStudent, IDPředmět

Z toho mála, co víme o fungování primárního klíče, by Vám mělo být jasné, že tímto způsobem lze zachytit jakoukoliv kombinaci cizích klíčů, ti pozornější z Vás si ale možná všimnou, že to není vše, co tato datová struktura umožňuje. Kombinace cizích klíčů se totiž může opakovat - a to je problém.

V případě, že bychom použili složený primární klíč - pak primární klíč jako kombinace hodnot jednotlivých polí, které tento klíč tvoří musí být unikátní. V případě, že ale složený primární klíč nahradíme jednoduchým primárním klíčem, např. ve formě automatického čísla - kontrola unikátnosti kombinace cizích klíčů už nebude zajištěna přímo databází - je tedy na nás jako uživatelích.

Zkusme alespoň nadefinovat propojení mezi tabulkami. I tady čelíme poměrně zásadnímu problému - Kexi nepodporuje ani relace, má v sobě zabudovanou ale podporu tzv. *lookup tabulek*. Lookup tabulky slouží jako číselníky, ze kterých se vyplní hodnota do pole. To znamená, že Kexi může ohlídat, že hodnota zaznamenávaná do pole IDPedagoga v tabulce PedagogPředmět je obsažena v políčku IDPedagog v tabulce Pedagog.

Definice lookup tabulky je znázorněna na obr. 4.4.



Obrázek 4.4: Nastavení lookup tabulky - Kexi

Nastavení lookup tabulky se provádí tak, že vybereme políčko, pro které se má nastavení provést. Propojení samotné se provádí v pravé části obrazovky - je ale nutné se přepnout z editoru vlastností pole do nastavení pole. Nastavit lze zdrojovou tabulku (record source), připojený sloupec (bound

column), nastavit je možné také sloupec, který se bude zobrazovat (visible column). V našem případě, bychom mohli místo IDPedagoga zobrazovat třeba příjmení pedagoga.

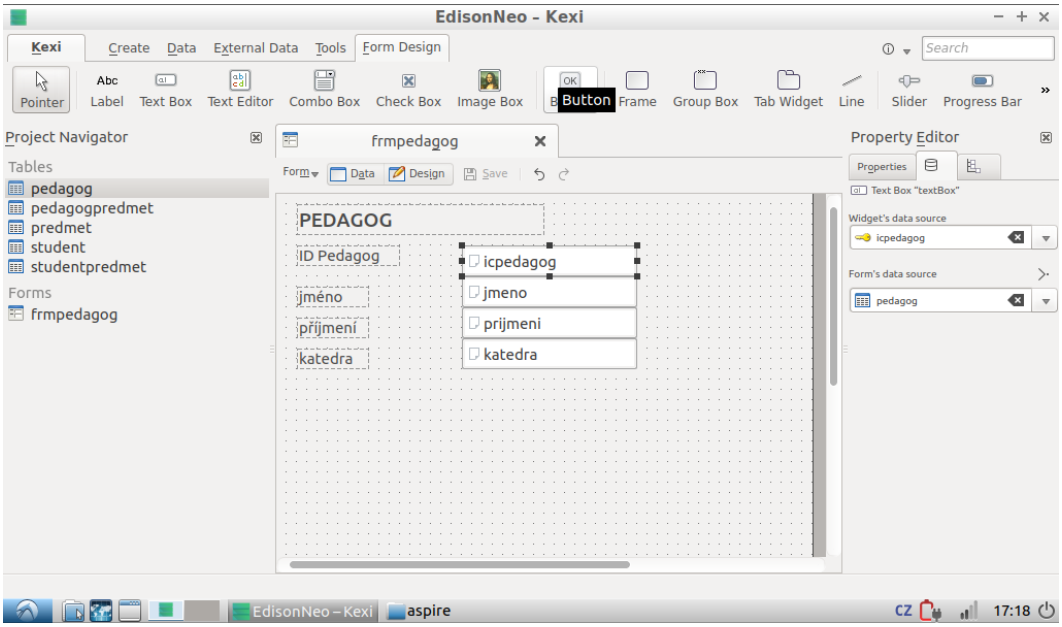
Nastavení viditelného sloupce není povinné.

4.2 Formulář

Kexi podporuje tvorbu jednoduchých formulářů. Jednoduchým se v tomto případě rozumí formuláře založené na jedné tabulce. Opět není to ideální, ale pro velmi jednoduché databáze to může postačovat. Kexi také nedává k dispozici průvodce, který by nám při tvorbě formuláře pomohl např. vygenerováním kostry formuláře, jak to umí MS Access.

Formulář tedy vytváříme tak, že nejprve vytvoříme nový prázdný formulář, pro který nastavíme zdroj záznamů (viz předchozí podkapitola) na tabulku kterou chceme formulářem vyplnit. Následně z panelu nástrojů vybíráme jednotlivé prvky formuláře a nastavujeme jejich vlastnosti.

Každá položka, která má zobrazovat obsah některého pole připojené tabulky musí mít toto pole namapováno, ve svých vlastnostech, viz obr. 4.5.



Obrázek 4.5: Návrh formuláře - Kexi

I v tomto případě se mezi jednotlivými režimy práce s formuláře přepínáme pomocí tlačítek data-/design.

4.3 Dotazy

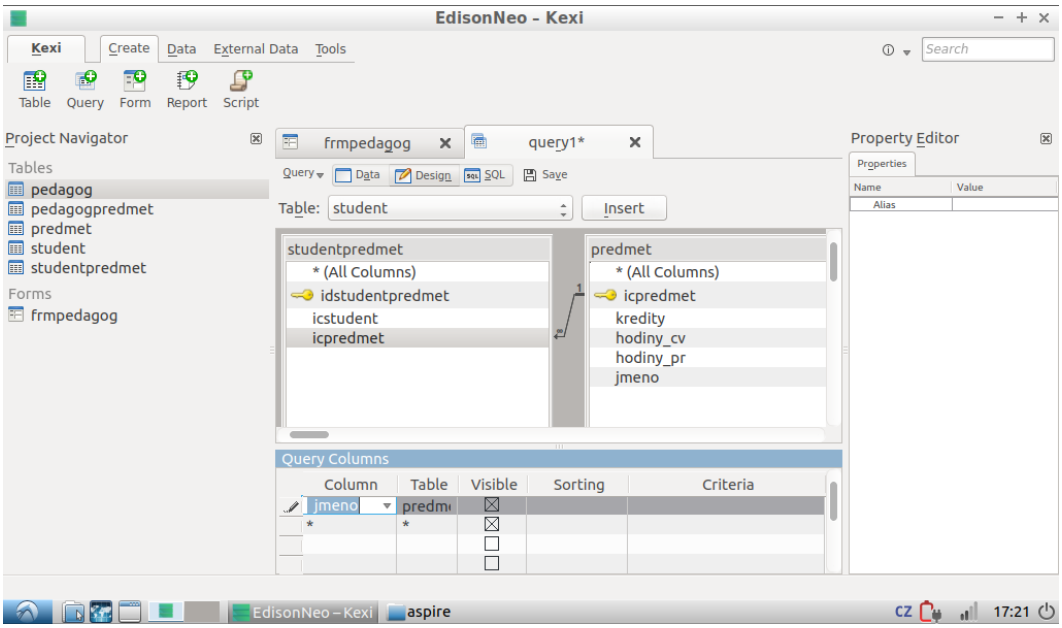
Dotazy jsou v případě databáze Kexi zvláštní - podívejte se na obr. 4.6.

Zvláštní ale v tomto případě neznamená omezené nebo špatné - ty pozornější napadne - není na obr. znázorněna relace? Odpověď je *ano i ne*. Relace jako takové nejsou podporovány, ale v případě dotazů je podporováno něco, z praktického pohledu vypadá a chová se jako relace.

Dotaz tedy konstruujeme tak, že se seznamu tabulek vybereme postupně tabulky, které mají být součástí dotazu a kliknutím na tlačítko insert je vložíme do dotazu.

Vybrané tabulky, ale nejsou propojené. Možná si vzpomenete na situaci, kdy jsme něco podobného dělali v MS Access - tam výběr tabulek do dotazu zároveň přenesl relace mezi nimi. Z předchozích podkapitol ale víme, že Kexi relace nepodporuje. Do dotazu se tedy nemá co přenést - relace pro Kexi dotazy je tedy potřeba definovat přímo v dotazu.

Relaci definujeme jednoduše přetažením souvisejících polí mezi tabulkami. Aby dotaz fungoval, musí být všechny tabulky použité v dotazu propojeny relacemi.



Obrázek 4.6: Návrh dotazu - Kexi

Teprve poté můžeme vybírat samotné pole, která má dotaz poskytnout. Vybíráme tabulku, ze které se se má pole použít, potom vybereme pole samotné. Lze nastavit, zda se pole má nebo nemá zobrazit. Podobně jako v případě MS Access nemá smysl do dotazu dávat pole, která nebudou viditelná a zároveň nemají nastaveno řazení (sorting) nebo filtry (criteria), jelikož takové pole k ničemu není :-).

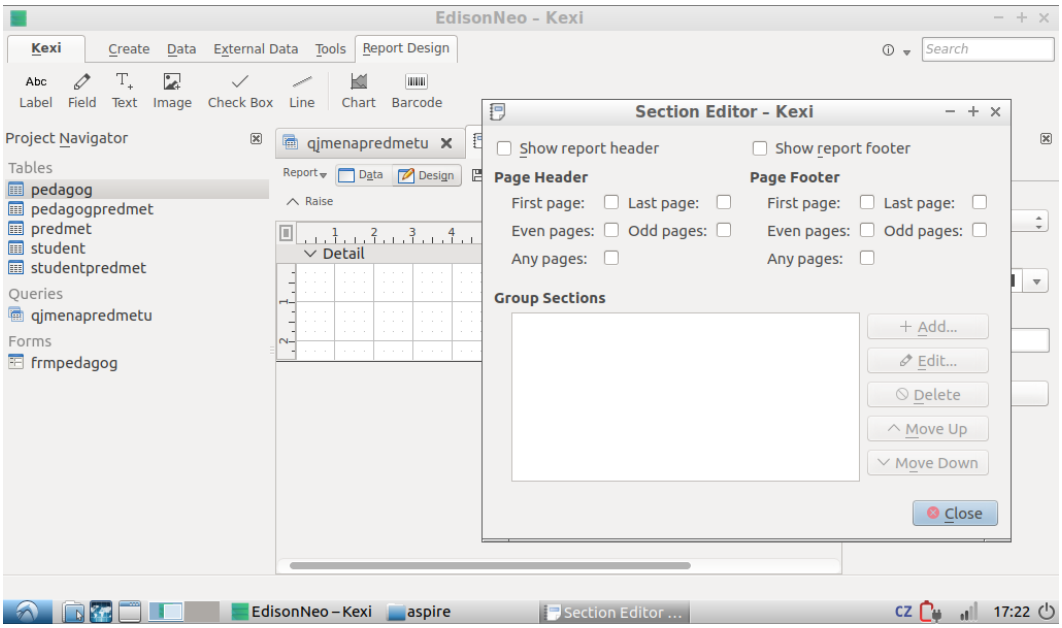
Filtrování a řazení funguje stejně jako v případě ostatních databázových systémů.

4.4 Sestavy

Sestavy se navrhují v Kexi buď na bázi tabulek nebo dotazů. Technicky je způsob podobný jako v případě návrhu formulářů. Tedy v nově vytvořené sestavě se provede napojení na zdroj - tabulku nebo dotaz. Následně se z panelu nástrojů vybírají drafické prvky jako jsou textová pole, popisky apod., které se pak mapují na jednotlivá políčka zdrojové tabulky/dotazu.

Oproti formulářům má ale některé další vlastnosti - má možnost nastavení záhlaví/zápatí, první/poslední stránku, sudé/liché stránky apod. Do jednotlivých částí je možno přidávat různé úrovně seskupení údajů.

Možností ovlivnit vzhled i funkci je tedy v případě sestav velké množství.



Obrázek 4.7: Návrh sestavu - Kexi

Literatura

[1] *Calligra Suite | The integrated work applications suite* [online]. [cit. 2017-01-13]. Dostupné z: https://www.calligra.org/

[2] *Office 365 přihlášení* [online]. [cit. 2017-01-10]. Dostupné z: https://portal.office.com/Home

[3] ASF. *Apache OpenOffice - Official Site - The Free and Open Productivity Suite* [online]. [cit. 2017-01-12]. Dostupné z: https://www.openoffice.org/

[4] DOCUMENT FOUNDATION. *Home | LibreOffice* [online]. [cit. 2017-01-12]. Dostupné z: https://www.libreoffice.org/

[5] MICROSOFT. *Microsoft Access 2016 Runtime* [online]. [cit. 2017-01-10]. Dostupné z: https://www.microsoft.com/cs-cz/download/details.aspx?id=50040

[6] ORACLE. *Download Free Java Software* [online]. [cit. 2017-01-12]. Dostupné z: https://java.com/en/download/

[7] W3C. *Extensible Markup Language (XML)* [online]. [cit. 2017-01-9]. Dostupné z: https://www.w3.org/XML/

[8] ŠENOVSKÝ, Pavel. *Počítače a ochrana dat - Návody do cvičení*. Ostrava: VŠB-TU Ostrava, 2006. 35 s.

Rejstřík