VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ
Ústav automatizace a informatiky

ZPRACOVÁNÍ INFORMACÍ

(doplňující text ke konzultacím v 1. ročníku kombinovaného bakalářského studia)

Doc. RNDr. Ing. Miloš Šeda, Ph.D.

BRNO 2002

Obsah

1. Úvod

Zpracování informací patří mezi nejčastější aplikace počítačů. Každá prodejna či sklad má svou evidenci, evidenci osob mají lékaři (sledují pacienty), státní instituce uchovávají informace v evidenci obyvatel, seznamech oprávněných voličů, trestním rejstříku, katastru nemovitostí, podnikatelé registrují své příjmy a výdaje v účetnické evidenci, knihovny vedou záznamy o knižním fondu, čtenářích a jejich výpůjčkách apod. K tomu, abychom mohli zpracovávat tyto údaje na počítači, potřebujeme mít k dispozici nějaký databázový systém. Databázových systémů je na trhu velké množství od velkých (a drahých), které nejčastěji běží pod operačním systémem UNIX a jsou určeny pro velké množství záznamů (miliony a více) až po všeobecně dostupné, které lze provozovat na běžném kancelářském počítači s operačním systémem Windows a které jsou schopny zpracovávat desítky, nejvýše stovky tisíc záznamů. Do první skupiny patří např. systémy ORACLE, INFORMIX, PROGRESS, INGRESS, SYBASE, MAGIC II. Druhou skupinu reprezentují např. systémy MS Access, FoxPro, Paradox, WinBase602 a PC FAND.

Cílem tohoto kursu je ovládnout základy práce s databázovým systémem MS Access, který je společně s produkty MS Word, MS Excel a MS PowerPoint součástí kancelářského balíku MS Office a pro svou cenovou dostupnost (do 20.000 Kč) je běžnou softwarovou výbavou téměř každého počítače.

Teorie databázových systémů zde není probírána, protože k tomu je určen speciální předmět *Databázové systémy* zařazený do 3. ročníku oboru *Aplikovaná informatika a řízení*. Pro návrh datových struktur budeme používat spíše intuitivní přístupy. Co se týče popisu programovacích jazyků (SQL a Visual Basic), omezíme výklad na nejnutnější informace, protože na úrovni 1. ročníku, kam je předmět *Zpracování informací* zařazen, předpokládáme hlavně využívání nejrůznějším průvodců MS Accessu.

Poděkování:

Autor děkuje za podporu z Projektu rozvoje bakalářských programů na Fakultě strojního inženýrství VUT v Brně a za možnost začlenit do textu odborné poznatky získané při řešení výzkumného záměru CEZ J22/98: 261100009 „Netradiční metody studia komplexních a neurčitých systémů“.

2. MS Access

2.1. Definice struktury tabulky

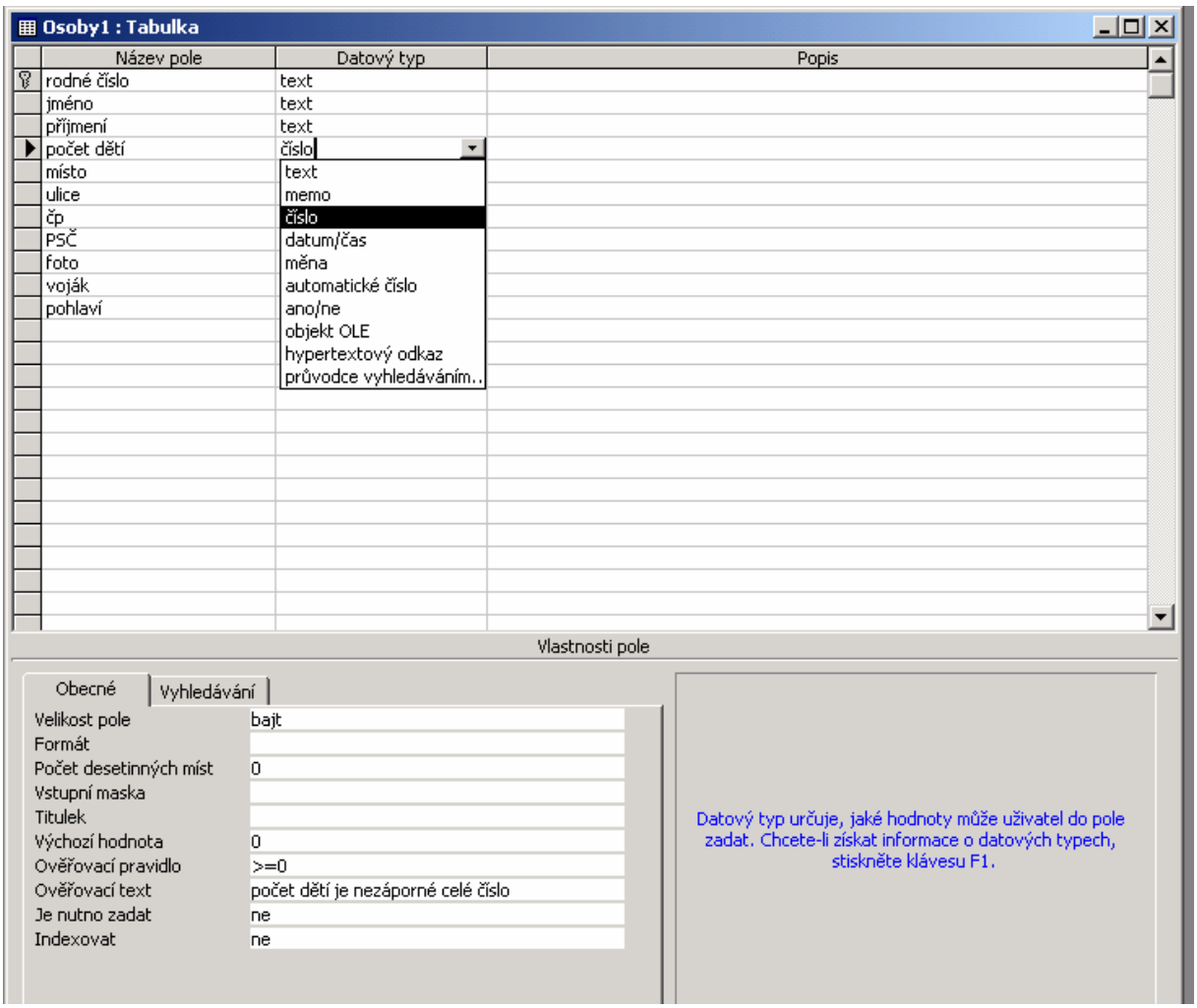
Příklad T1:

Předpokládejme tabulku

OSOBY1(rodné-číslo, jméno, příjmení, foto, místo, ulice, čp, PSČ, počet-dětí, pohlaví, voják)

V této tabulce určuje položka rodné číslo primární klíč, tj. položku, která jednoznačně určuje řádek tabulky. Primární klíč v předchozím zápisu je označen podtržením příslušné položky. Někdy může být primární klíč tvořen kombinací více položek.

V oknu Databáze zvolíme objekt Tabulky a pro něj volbu Vytvořit tabulku v návrhovém zobrazení. V následném dialogu zadáme názvy položek a jejich datový typ.



Obr. T1: Definice struktury tabulky

V obr. 1 je vidět definice jednotlivých položek tabulky. V definičním okně se uvede název položky, její datový typ a případně ještě komentář, který je nepovinný. Datové typy,

které jsou v MS Accessu k dispozici, jsou vidět z rozbalené rolety Datový typ pro položku počet dětí.

- Datový typ text zhruba odpovídá typu string známému v Pascalu.
- Typ memo je rovněž text, který však není omezen na jeden řádek tabulky.
- Typ číslo zahrnuje všechny číselné typy Pascalu (integer, real apod.). Ve spodní části jej lze v okně Vlastnosti pole a jeho vlastnosti Velikost pole upřesnit na jednu z možností bajt, celé číslo, dlouhé celé číslo, číslo s jednoduchou nebo dvojitou přesností atd.
- Typ datum/čas je v MS Accessu velmi důležitý, protože datum či čas se v databázích často zaznamenávají, např. datum narození, datum vystavení faktury a její splatnosti apod. S položkami tohoto typu je možné provádět také aritmetické operace, např. je možné k hodnotě typu datum přičíst celé číslo reprezentující počet dnů a dostat tak nové datum nebo dvě položky typu datum mezi sebou odečítat a zjistit, kolik dnů je dělí apod.
- Položku typu automatické číslo můžeme použít tehdy, když mezi datovými položkami tabulky žádná položka ani jejich kombinace nereprezentuje primární klíč a stačí řádky tabulky odlišit jejich prostým očíslováním posloupností přirozených čísel.
- Datový typ měna je vedle typu datum/čas dalším specifickým typem databázových aplikací. Jde vlastně o čísla s rozšířením o měnový symbol.
- Datový typ ano/ne odpovídá pascalovskému typu Boolean. Položky tohoto typu mohou nabývat pouze dvou hodnot: true nebo false.
- Datový typ objekt OLE umožňuje do tabulky vložit odkaz na externí objekty vytvořené v jiných aplikacích Windows, nejčastěji jde o obrázky, můžeme sem však začlenit odkaz i na vzorec, graf Excelu apod.
- Datový typ hypertextový odkaz umožňuje do položky tohoto typu vložit odkaz na www adresu.
- Poslední heslo ze seznamu datových typů průvodce vyhledáváním.. vlastně není žádným datovým typem, ale slouží k aktivaci průvodce, pomocí nějž můžeme buď definovat statický seznam hodnot, kterých příslušná položka může nabýt (odpovídá výčtovému datovému typu z Pascalu) nebo definovat SQL dotaz, jehož výsledek bude plnit tento seznam dynamicky. Při vstupu údaje do položky tohoto typu se v poli na jeho pravé straně objeví rozbalovací šipka a po kliknutí na ni se vybalí příslušný statický nebo dynamický seznam hodnot, z nějž jednu hodnotu vybereme.

V okně Vlastnosti pole jsou vidět kromě zmíněné vlastnosti velikost pole ještě další vlastnosti (jejich seznam závisí na datovém typu položky), z nichž je důležitá zejména vlastnost Ověřovací pravidlo, které umožňuje definovat podmínku, která musí být splněna, aby vstupující údaj byl přijat a bylo možné dané pole opustit. Chybovou zprávu o tom, proč hodnota není akceptována je možné uživatelem definovat ve vlastnosti Ověřovací text a zastínit tak velmi obecnou chybovou zprávu MS Accessu. Na obr. 1 je definováno ověřovací pravidlo zajišťující to, že vstupující informace o počtu dětí musí být nezáporným celým číslem. Aby nebylo možné pracovat s desetinnými čísly je nastavena pro tento údaj hodnota 0 ve vlastnosti Počet desetinných míst.

Vlastnost Formát umožní provést konverzi zadaného údaje (např. na velká písmena atd.), vlastnost Vstupní maska zase umožní předdefinovat určité pozice vstupujícího údaje. Kódy, které lze v posledních dvou vlastnostech využívat, lze najít, včetně ukázkových příkladů, v Helpu MS Accessu, který zpřístupníme po stisku klávesy F1 z pole příslušné vlastnosti.

Vlastnost Výchozí hodnota je užitečné definovat tehdy, jestliže ve většině případů bude položka nabývat jistou hodnotu, např. v našem příkladě předpokládáme, že osoby (např. studenti), jejíž údaje sledujeme, jsou bezdětní, a tedy má smysl hodnotu položky počet dětí předdefinovat jako 0.

Nastavení vlastnosti Je nutno zadat udává, zda údaj příslušné položky je povinný anebo může být prázdný.

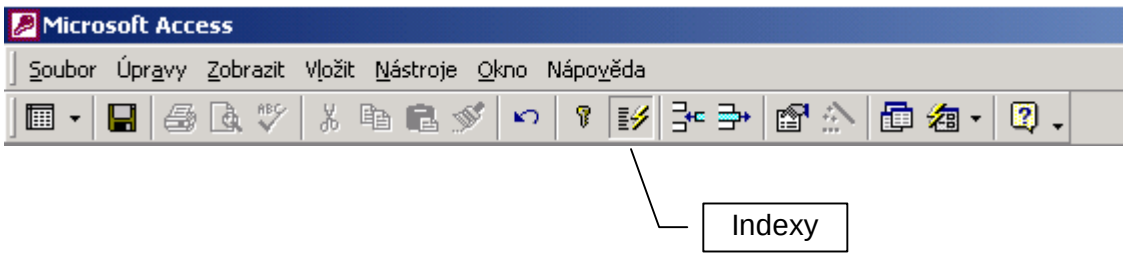
Vlastnost Indexovat specifikuje, zda příslušná položka je indexem, a umožňuje tedy seřazení řádků tabulky (ve výpisech při zapnutí tohoto indexu) podle vzestupného nebo sestupného pořadí hodnot položky.

Primární klíč je ve v definici struktury tabulky vyznačen obrázkem klíče, jak je vidět v obr. T1 vlevo položky rodné číslo.

Poznámka:

1. Pro jednu tabulku může být definováno více klíčů.
2. Klíč tabulky (jeden nebo i více) může být vícesegmentový (tvořený více položkami). Např. klíčem pro hledání v telefonním seznamu je příjmení, jméno a údaje o adrese. ♦

Na obr. T2 je vidět tvar panelu nástrojů v režimu definice struktury tabulky a je zde vyznačena ikona Indexy, která umožní definovat další klíče.



Obr. T2: Panel nástrojů při definici struktury tabulky

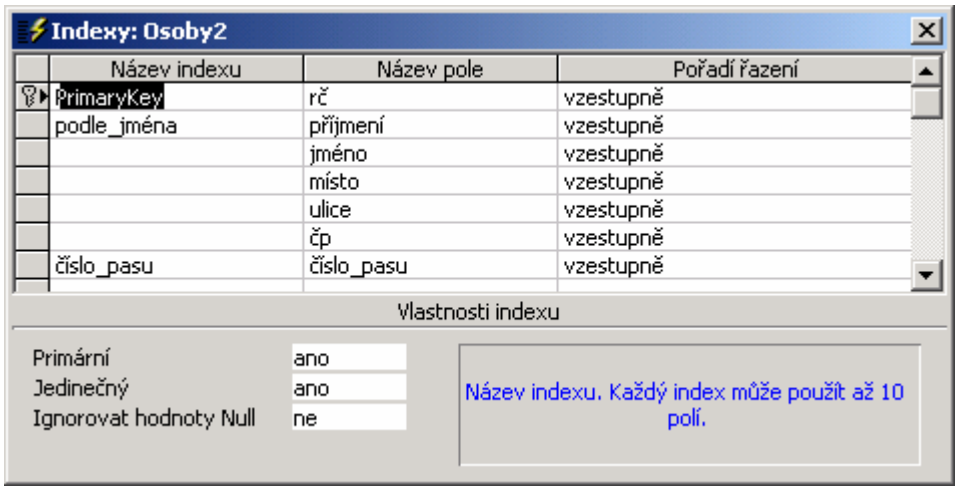
Jestliže klikneme na ikonu Indexy, otevře se okno, v němž můžeme definovat další klíče. která umožní definovat další klíče. Na obr. T3 jsou definovány tři klíče PrimaryKey (primární klíč) reprezentovaný položkou rč, dále pak vícesegmentový klíč podle_jména a konečně klíč číslo_pasu tvořený stejnojmennou položkou.

Při aktualizaci dat v tabulce (přidávání, rušení, modifikace záznamů) se průběžně aktualizují všechny „indexové soubory“ příslušející jednotlivým klíčům a z kódu Visual Basicu je možné docílit „zapnutím“ určitého indexu, že další akce, např. různé výpisy údajů, se budou provádět nikoliv podle primárního klíče, ale toho klíče, který odpovídá zapnutému indexu.

V okně Indexy je vícesegmentový klíč definován tak, že ve sloupci Název indexu je název klíče uveden pouze u prvního segmentu uvedeného ve sloupci Název pole a u všech dalších segmentů klíče již název indexu není uveden (příslušná pole ve sloupci Název indexu jsou prázdná).

Poznámka:

1. Vícesegmentovým klíčům se snažíme vyhýbat, protože je složitější je využívat k vyhledávání záznamů, snazší je určení záznamu jednoznačně určeným kódem, např. rodným číslem, číslem knihy (ISBN), kódovým označením zboží na skladě, pořadovým číslem objednávky, apod. U příkladu na obr. T3 navíc ani klíč podle_jména nemusí být jednoznačný pro určení záznamu v tabulce, protože nelze vyloučit případ, kdy dvě osoby mají všechny segmenty klíče (příjmení, jméno, místo, ulice, čp, PSČ) shodné. To může nastat v případě, kdy otec a jeho stejně se jmenující syn bydlí na stejné adrese.
2. Vždy však není možné se vícesegmentovým klíčům vyhnout, např. v tzv. *průnikových* tabulkách určeným k realizaci vztahu M:N. Zde je klíč tvořen tzv. *cizími klíči* vztahujících se tabulek. ♦



Obr. T3: Definice klíčů tabulky

Cvičení:

1. Definujte ověřovací pravidlo pro zadávání seznamu osob, které reprezentují oprávněné voliče (nesmí být mladší 18 let).
2. Definujte ověřovací pravidlo pro vkládání rodného čísla osob, víte-li, že musí jít o číslo dělitelné beze zbytku 11. Protože je výhodnější volit pro rodné číslo datový typ text (např. proto, že za skupinou rrmmdd v rodném čísle mohou být 3 nebo 4 cifry a při řazení podle rodného čísla a tím i podle věku by mohlo dojít ke zkreslení díky řádově jiným údajům), využijte k tomu účelu potřebné konverzní funkce uvedené v poslední kapitole textu.
3. Definujte klíče následujících tří tabulek

   ČTENÁŘI(IDčtenáře, příjmení, jméno, …)
   KNIHY(ISBN, název, autor, rok-vydání, nakladatelství, …)
   VÝPŮJČKY(IDčtenáře, ISBN, datum)

2.2. Definice jednoduchého dotazu

V tomto odstavci si ukážeme, jak vytvořit jednoduchý dotaz, aniž bychom znali dotazovací jazyk SQL.

Příklad D1:

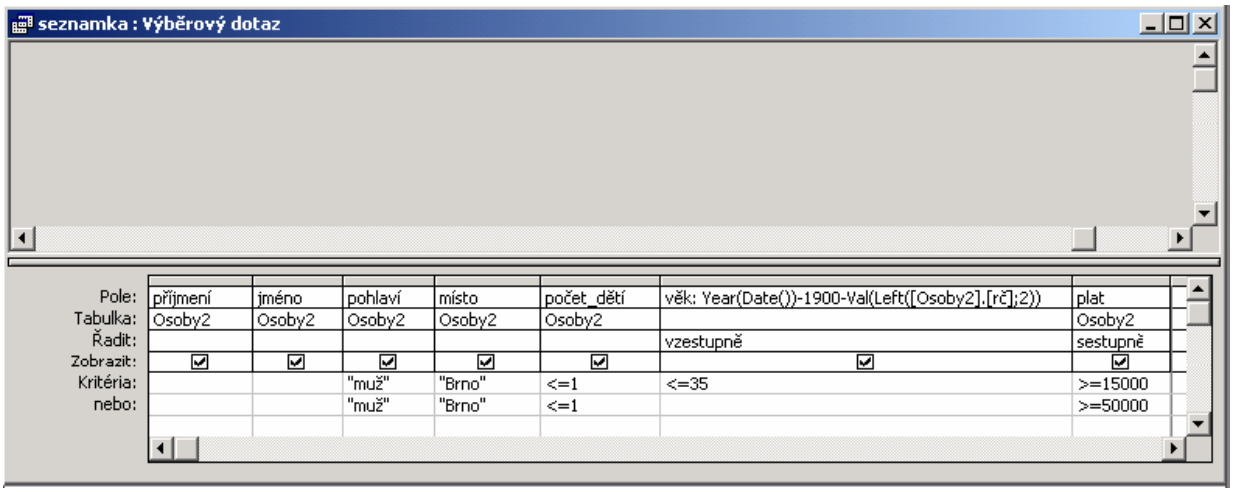
Předpokládejme, že máme k dispozici tabulku

OSOBY2(rodné-číslo, jméno, příjmení, foto, místo, ulice, čp, PSČ, počet-dětí, pohlaví, plat)

a chceme z ní vybrat všechny muže, kteří bydlí v Brně, mají nejvýše jedno dítě, jejich věk nepřevyšuje 35 let a plat mají minimálně 15000 Kč anebo na věku nezáleží a pak jejich plat musí převyšovat 50000 Kč a ostatní požadavky zůstávají stejné.

Poznámka:

Pokud bychom takový dotaz mínili jako informaci pro případné seznámení, pak by zde jistě nesměl chybět údaj o stavu osob. V tom se však problém komplikuje, protože označení stavu je závislé na pohlaví (ženatý resp. vdaná, svobodný – svobodná, rozvedený – rozvedená, vdovec – vdova). Jak z informace o pohlaví naplnit správnou čtveřici možných stavů rozbalovací seznam, z nějž stav osoby vybereme, ukážeme později v odstavci popisujícím formuláře a jeho řídící prvek Skupina voleb. Zde pro zjednodušení nebudeme stav osob uvažovat. ♦



Obr. D1: Návrhové zobrazení dotazu

Na obr. D1 je vidět konstrukci dotazu v návrhovém zobrazení. Návrh vlastně reprezentuje tabulku, která reprezentuje výstupní tvar údajů, které prováděný dotaz vrátí.

Řádek Pole: návrhu obsahuje buď (1) název položky některé tabulky nebo (2) výraz, v němž mohou být obsaženy položky tabulky, konstanty a také vestavěné funkce MS Accessu. V prvním případě je pak v řádku Tabulka: návrhu pod názvem položky uveden název tabulky, do níž položka patří; ve druhém případě je příslušná buňka prázdná.

V řádku s popisem Řadit: se uvádí, zda se má výsledek dotazu podle příslušné položky nebo výrazu uspořádávat vzestupně nebo sestupně. Volba vzestupně je implicitní, u číselných údajů to znamená řazení od nejmenších hodnot k největším, u údajů typu text pak obvyklé řazení podle abecedy (od „a“ do „z“). Jestliže specifikujeme řazení u několika sloupců, pak

se řazení provádí podle toho řadícího sloupce, který je nejvíce vlevo a až při rovnosti údajů podle dalšího řadícího sloupce, atd. V dotazu podle obr. D1 se tedy osoby, které splňují požadovaná kritéria, seřadí podle věku od nejmladšího k nejstaršímu, a ti, kteří jsou stejně staří, podle výše platu od nejvyššího k nejnižšímu. Jestliže bychom chtěli řadit výsledek dotazu nejdříve podle platu a pak podle věku, museli bychom pořadí příslušných sloupců mezi sebou vyměnit. To lze jednoduše docílit tím, že jeden ze sloupců vybereme kliknutím levým tlačítkem myši nad ním, a pak jej se stisknutým levým tlačítkem myši „přetáhneme“ na požadovanou pozici.

Protože v údajích tabulky přímo nemáme informaci o věku osob, ale jsou zde zaznamenána rodná čísla, určíme věk odtud. Je totiž známo, že v prvních dvou pozicích rodného čísla je poslední dvojčíslí roku narození. Jestliže od aktuálního roku, který zjistíme pomocí funkce Year(datum) z počítačového data Date() odečteme 1900 a dvojčíslí roku narození, dostaneme věk osoby. Dvojčíslí roku narození získáme z rodného čísla pomocí funkce Left(řetězec, počet znaků) aplikované na první 2 znaky. Tento podřetězec je funkcí Val(řetězec) konvertován na číslo, aby mohl vystupovat v aritmetických operacích.

V zápisu

věk: Year(Date())-1900-Val(Left([Osoby2].[rč];2))

vyjadřuje návěští věk: to, že ve výsledku dotazu bude mít příslušný sloupec záhlaví s tímto textem. Kdybychom žádné návěští nespecifikovali, MS Access by text záhlaví stanovil sám ve velmi obecném označení, např. jako Výraz1.

Poznámka:

Hranaté závorky v tomto zápisu vymezují začátek a konec identifikátoru. Jestliže identifikátory jsou tvořeny jedním slovem, čímž míníme i více slov spojených „podtržítkem“, např. identifikátor rodné_číslo, nemusí být hranaté závorky uvedeny. Jestliže však identifikátor je tvořen více slovy oddělených mezerami, např. rodné číslo, pak jsou hranaté závorky k vymezení identifikátoru nezbytné. ♦

Řádek Zobrazit: umožňuje při ladění dotazu některé sloupce z výstupu dotazu vypustit, pokud nás zrovna nezajímají. Zařídíme to tak, že v návrhu nebudou zaškrtnuta příslušná políčka.

Velmi důležité jsou řádky Kritéria: a nebo:. Umožňují specifikovat libovolně složitou filtrovací podmínku. V buňkách obou řádku jsou specifikovány dílčí podmínky, které jsou v rámci jednoho řádku ve vztahu konjunkce. to znamená, že všechny dílčí podmínky v jednou řádku musí být splněny, aby příslušný záznam byl výběrovým dotazem vrácen. Vztah řádků mezi sebou je disjunkcí. Jestliže definujeme nějaké podmínky v řádku nebo:, pak se automaticky v návrhu vytvoří další řádek nebo:, do nějž můžeme vkládat další podmínky atd. Záznam bude vybrán dotazem na výstup, jestliže jsou splněny všechny dílčí podmínky řádku Kritéria: anebo všechny dílčí podmínky řádku nebo: atd.

Poznámka:

1. Dílčí podmínky mají tvar porovnání. První operand této relace je v řádku Pole:, operátor porovnání následovaný druhým operandem se nachází (ve stejném sloupci jako první operand) v řádku Kritéria: popř. v některém z řádků nebo:.
2. Prázdné buňky v řádcích Kritéria: a nebo: vyjadřují, že v tomto údaji není definováno žádné omezení. ♦

Při vytváření dotazu v návrhovém zobrazení se na pozadí vytváří SQL dotaz, který si lze prohlédnout přes menu Zobrazit a volbu Zobrazení SQL. V našem příkladu zde bude výběrový dotaz ve tvaru:

```
SELECT Osoby2.příjmení, Osoby2.jméno, Osoby2.pohlaví, Osoby2.místo,
Osoby2.počet_dětí, Year(Date())-1900-Val(Left([Osoby2].[rč],2)) AS věk, Osoby2.plat
FROM Osoby2
WHERE (((Osoby2.pohlaví)="muž") AND ((Osoby2.místo)="Brno") AND
((Osoby2.počet_dětí)<=1) AND ((Year(Date())-1900-Val(Left([Osoby2].[rč],2)))<=35) AND
((Osoby2.plat)>=15000)) OR (((Osoby2.pohlaví)="muž") AND ((Osoby2.místo)="Brno") AND
((Osoby2.počet_dětí)<=1) AND ((Osoby2.plat)>=50000))
ORDER BY Year(Date())-1900-Val(Left([Osoby2].[rč],2)), Osoby2.plat DESC;
```

Jednotlivé klauzule tohoto dotazu nyní nebudeme rozebírat, protože SQL jazyku se věnujeme v samostatné kapitole.

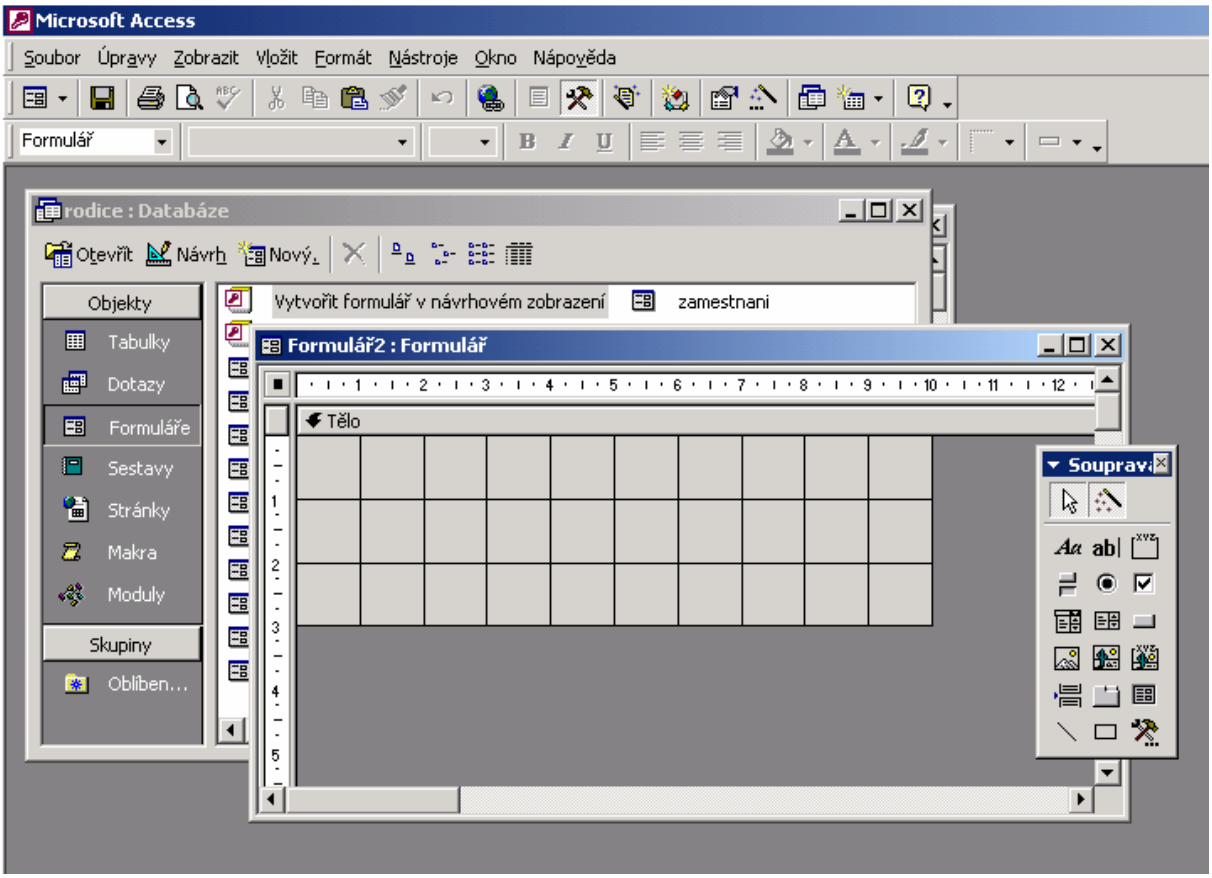
Cvičení:

1. Vytvořte dotaz, který z údajů v příkladu D1 vybere ty, kteří se narodili ve znamení Kozoroha, tj. v období od 22. prosince do 20. ledna. Návod: je třeba testovat, zda datum narození spadá do příslušné části prosince (od 22.12. do 31.12.) anebo do příslušné části ledna (od 1.1. do 20.1.). U žen musíme vzít úvahu to, že k měsíci jejich narození je v rodném čísle přičteno číslo 50.
2. Vytvořte dotaz, který seznam uchazečů v konkursním řízení zúží na ty, kteří buď mají vysokoškolské vzdělání anebo mají středoškolské vzdělání, ale v tomto případě je vyžadována praxe alespoň 5 let.

2.3. Vytváření formulářů

V tomto odstavci uvedeme pouze základní informace o formulářích, některé další zmíníme v dalších kapitolách, protože při tvorbě formulářů je třeba velmi často znát příkazy SQL jazyka nebo Visual Basicu, které budou stručně popsány později.

Jestliže v okně Databáze zvolíme volbu Formuláře a pro ně Vytvořit formulář v návrhovém zobrazení, objeví se prostředí návrhu formuláře následujícího tvaru:



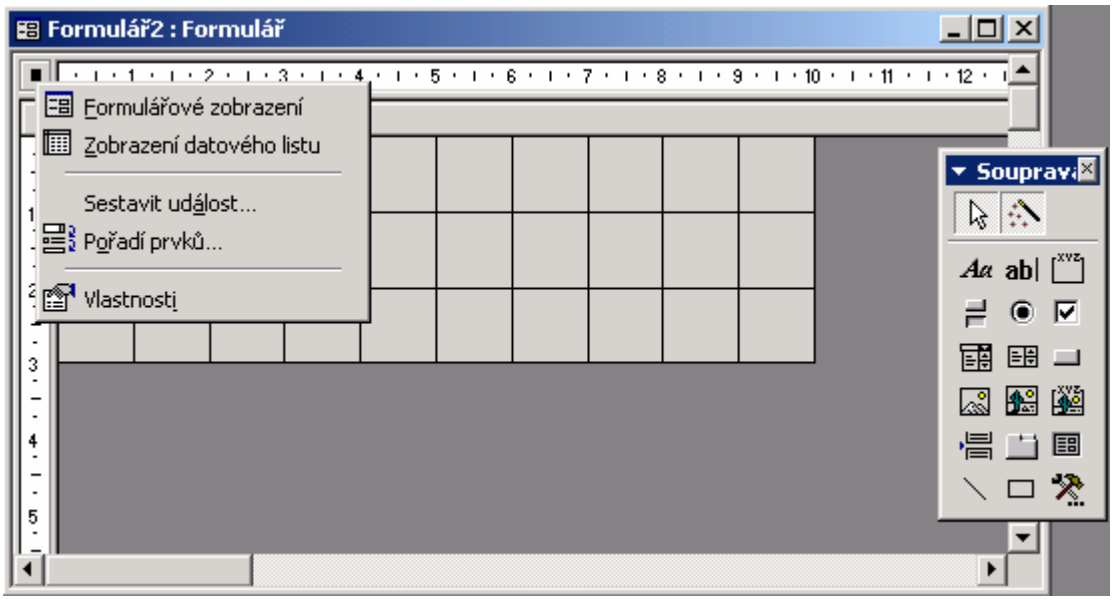
Obr. F1: Návrhové zobrazení formuláře

Zde vidíme vlastní okno formuláře, kam budeme vkládat jednotlivé prvky formuláře (popisky, editační políčka, tlačítka, seznamy apod.) a napravo okno s panelem nástrojů, který ke vkládání jednotlivých prvků slouží. Vlastní umístění oken, jejich tvar a velikost lze samozřejmě změnit, jak je to v prostředí Windows obvyklé.

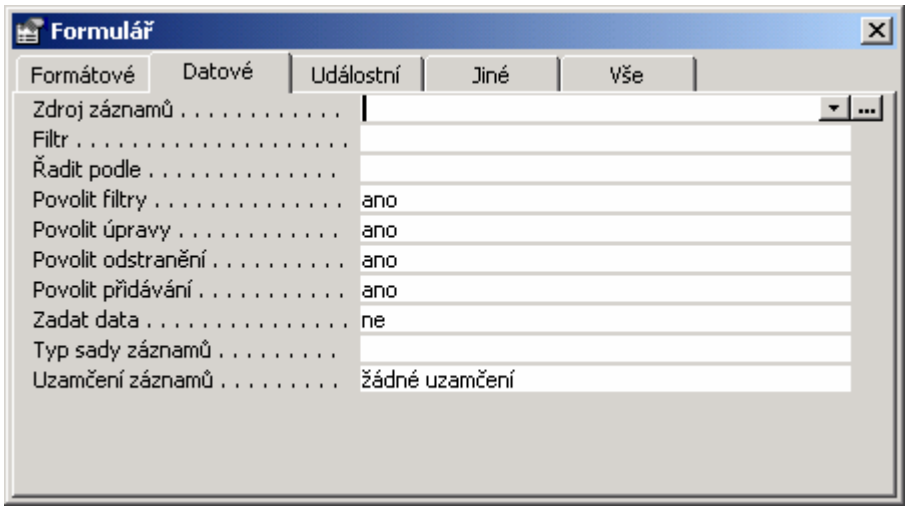
Okno s panelem nástrojů je k dispozici proto, že je zvýrazněna ikona  , v bublinové nápovědě označená jako Souprava nástrojů, která je umístěna zhruba pod heslem Nápověda. Zvýrazníme ji snadno kliknutím na ni, dalším kliknutím bychom ji „vypnuli“ a okno s panelem nástrojů by zmizelo.

Vidíme však, že není zvýrazněna ikona Seznam polí  první zleva vedle ikony Souprava nástrojů. Jestliže je ikona Seznam polí zvýrazněna, pak jsou k dispozici položky nějaké tabulky a je možné je do formuláře jednoduchým způsobem začleňovat. Abychom ji zvýraznili, nestačí zde na ni pouze kliknout, ale informaci o „podkladové“ tabulce formuláře definovat v jeho vlastnostech. Zpřístupnit je lze dvěma způsoby:

1. Klikneme-li pravým tlačítkem myši na „čtvereček“ kde se protínají vodorovné a svislé pravítko, rozbalí se lokální menu, jehož nejspodnější volbou jsou Vlastnosti. Situaci vidíme na obr. F2. Jestliže na tuto volbu klikneme, otevře se okno vlastností, viz obr. F3.

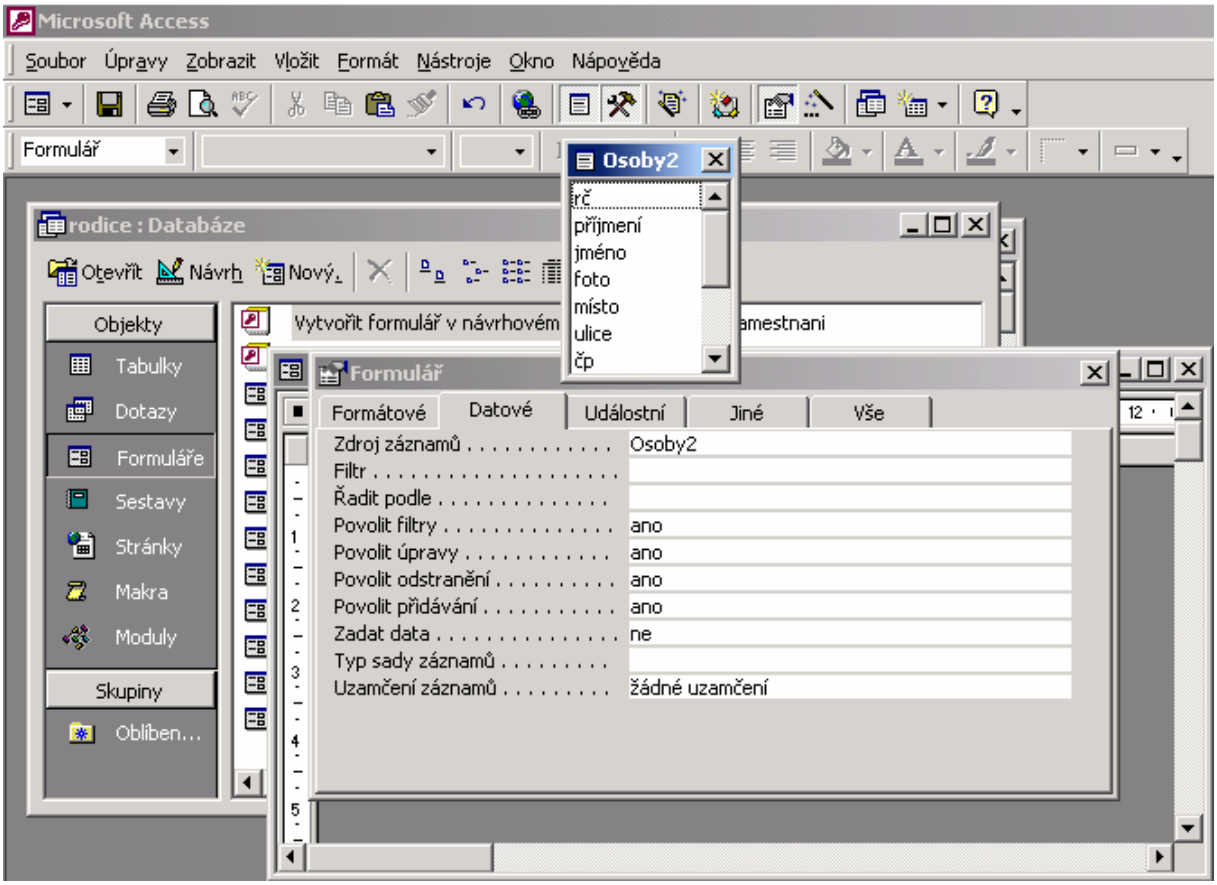


Obr. F2: Zpřístupnění vlastností formuláře



Obr. F3: Vlastnosti formuláře

Jestliže vybereme vlastnosti Datové a v řádku s vlastností Zdroj záznamů klikneme na šipku rozbalovacího seznamu, zobrazí se seznam tabulek a uložených dotazů, které jsou v aplikaci k dispozici. Když mezi nimi jednu tabulku (nebo dotaz) vybereme, stane se zdrojem záznamů. To znamená, že řádky tabulky (resp. výsledku dotazu) budou plnit pole formuláře svázaná s položkami tabulky (dotazu). Současně se zvýrazní předtím „zhasnutá“ ikona Seznam polí a otevře se malé okno se seznamem definovaných položek tabulky (dotazu), které je pak možné do prostředí návrhového zobrazení formuláře z tohoto okna jednoduše levým tlačítkem myši přetahovat. Předtím je však vhodné okno vlastností křížkem v pravém horním rohu zavřít, aby v ploše nebylo příliš mnoho překrývajících se oken.



Obr. F4: Zpřístupnění podkladové tabulky formuláře

2. Jiným, rychlejším způsobem zpřístupnění vlastností formuláře je kliknout na ikonu Vlastnosti  . Tím se přeskočí krok s lokálním menu. Aby tento způsob správně fungoval, nesmí být ve formuláři vybrán žádný jeho prvek.

Obr. F5: Souprava nástrojů (ve dvou různých oknech)

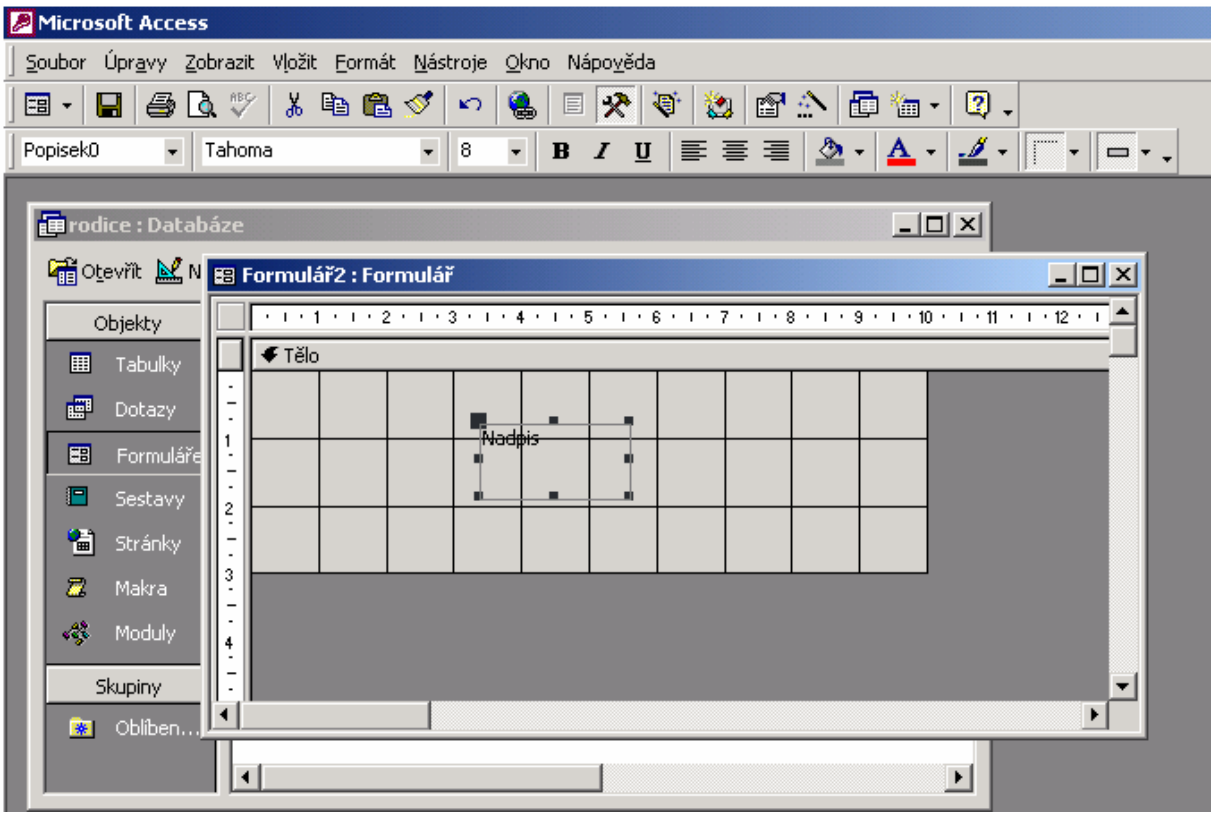
Poznámka:

Definováním podkladové tabulky formuláře jsme se zde zabývali tak podrobně proto, že je velmi smutné vidět, když mnozí studenti často ani po měsíci výuky ve cvičeních tento úkon s jistotou neovládají a opakovanými dotazy brzdí další výklad. ♦

2.3.1. Základní prvky formulářů

V předchozím textu jsme se již zmínili o soupravě nástrojů (viz. obr. F5 na předchozí straně) a jak ji zpřístupnit. Nyní probereme některé jednoduché nástroje. Složitější prvky, které souvisí s programováním, popíšeme později.

- Nejjednoduššími nástroji jsou předposlední dva nástroje  a  , které slouží k nakreslení čáry, resp. rámečku ve formuláři.
- Prvek  je určen k vložení statického textu do formuláře. Např. se může využít k vložení různých popisek a nadpisů. Po kliknutí na tuto ikonu nakreslíme ve formuláři obdélník, do něhož napíšeme vlastní text. Jestliže vymezený obdélník opustíme a vybereme jej tak, že se na něm objeví úchytné body, viz obr. F6, můžeme pak tažením za ně změnit jeho rozměry a hlavně jsou k dispozici všechny formátovací prostředky pro popisný text. Můžeme nastavit tvar písma (font), jeho velikost, styl (**B** - tučně, ***I*** - kurzíva, **U** - podtrženě), zarovnání textu uvnitř rámečku (vlevo, doprostřed, na pravý okraj), volit barvu pozadí, barvu písma a barvu ohraničujícího rámečku.



Obr. F6: Vkládání popisného textu do formuláře

- Prvek Textové pole  je nejčastěji svázán s položkou v podkladové tabulce formuláře a do formuláře jej vkládáme ze seznamu polí táhnutím myší, jak bylo uvedeno výše. Tím se automaticky do formuláře dostane i popisný prvek ***Aa***, jehož text je odvozen od názvu položky ve struktuře tabulky, lze jej ovšem z klávesnice změnit. Označení Textové pole je poněkud matoucí, protože tento prvek může být svázán s položkou tabulky libovolného datového typu.

  Někdy ovšem prvky  nemusí být se žádnou položkou tabulky svázány, např. tehdy, když do nich z klávesnice zadáváme výběrová kritéria.

  V prvku  se mohou také vyskytovat vypočítávané výrazy, jako je tomu v MS Excelu. Pak tento výraz je uveden znakem =. Např. jestliže v poli dat1 bude uloženo datum vystavení faktury, pak v jiném poli může uveden výraz =dat1+14, který automaticky vypočítává datum její splatnosti 14 dnů po vystavení faktury.
- 3 prvky  , které se zleva označují jako Přepínací tlačítko, Přepínač a Zaškrtávací políčko slouží k přepínání stavů 1/0, true/false, pravda/nepravda, ano/ne. Poznamenejme však, že prvky přepínače  vystupují i ve složitějším prvku Skupina voleb  , kde vytváří jakýsi polohový přepínač, v němž může být vybrána jen jedna poloha.

2.3.2. Tlačítka ve formuláři a jednoduché událostní procedury

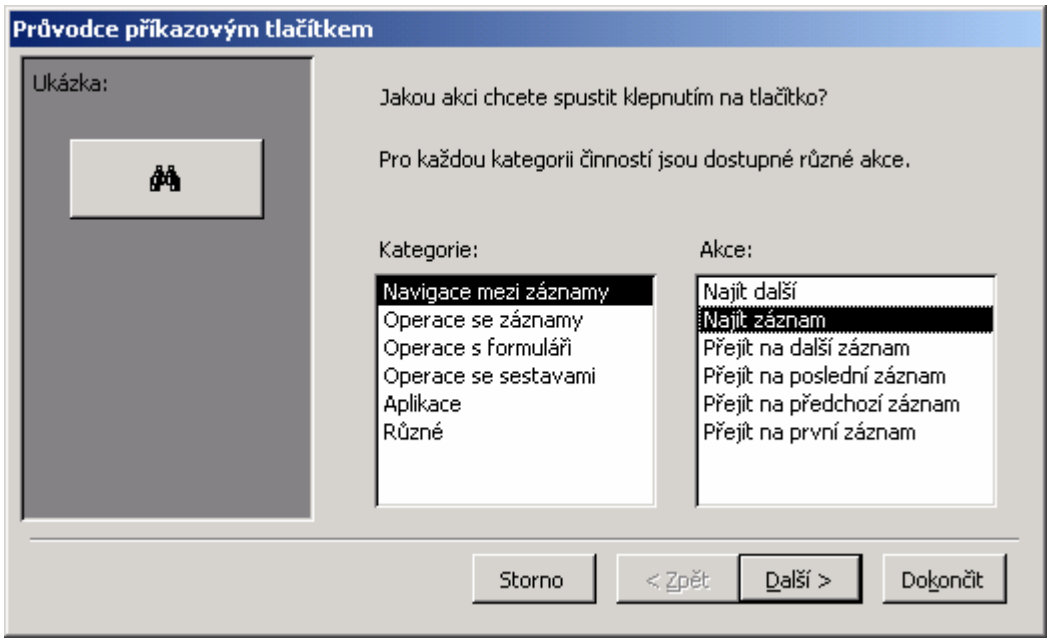
Součástí téměř všech formulářů jsou tlačítka, po jejichž stisku se zpravidla odehrají nějaké události, např. otevře se další formulář, provede se nějaká aktualizační operace, apod. Vzhledem k významu funkce tlačítek je pro ně předprogramována řada jednoduchých událostních procedur, které umožňují i začátečníkům řešit některé základní funkce aplikace, aniž by museli znát SQL jazyk či Visual Basic. Při pokročilejším vytváření aplikací většinou tyto procedury využíváme pouze jako základní řešení, které podle svých potřeb přizpůsobíme.

Jestliže v návrhovém zobrazení formuláře klikneme na prvek Příkazové tlačítko  v soupravě nástrojů a následně myší nakreslíme jeho obrys, pak se aktivuje Průvodce příkazovým tlačítkem, v jehož prvním kroku volíme kategorii předdefinovaných akcí a v ní vlastní akci. Postup ukážeme na příkladu akce Najít záznam, viz obr. F7. V dalším kroku zvolíme pro vkládané tlačítko textový popis nebo grafický symbol (obr. F8).

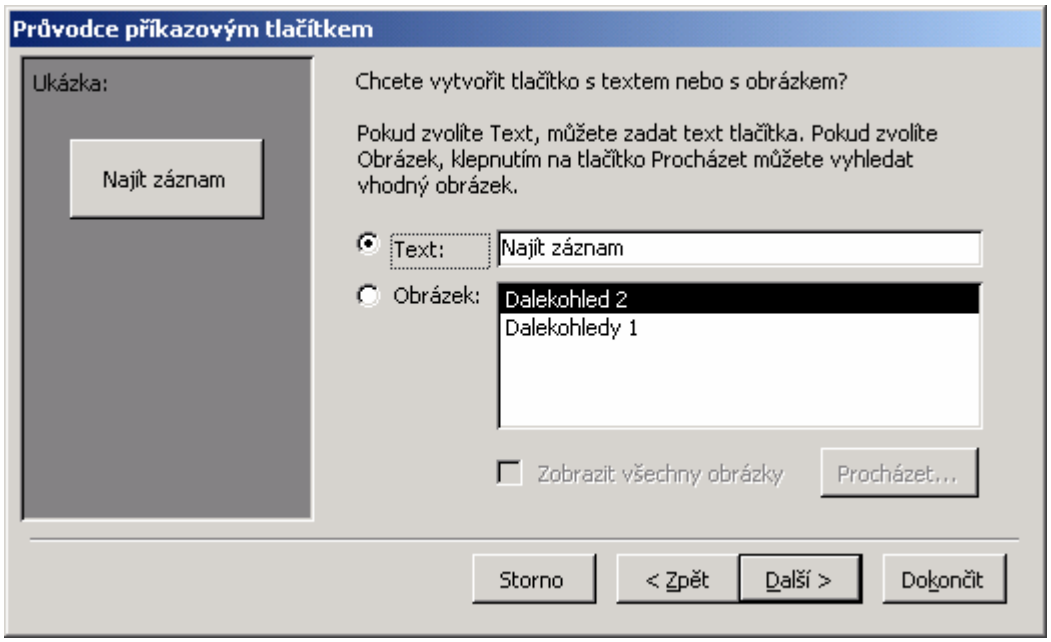
Poznámka:

Grafické symboly se snažíme používat v omezené míře, protože ikona nemusí být zcela vypovídající o vlastní akci a uživatel se tak nezatěžuje nutností si pamatovat její význam. ♦

V 3. kroku průvodce tlačítku přiřadíme interní jméno (obr. F9), na které se můžeme odkazovat ve zdrojovém kódu Visual Basicu. Zde je vhodné jméno volit mnemotechnicky a nepřijmout Accessem nabízené označení (složeno ze slova Příkaz a pořadového čísla prvku ve formuláři), protože při větším počtu tlačítek ve formuláři a jejich obecném označení se komplikuje orientace ve zdrojovém kódu událostních procedur. Po volbě označení dokončíme práci s průvodcem stiskem příslušného tlačítka, jak je patrné z obr. z obr. F9.

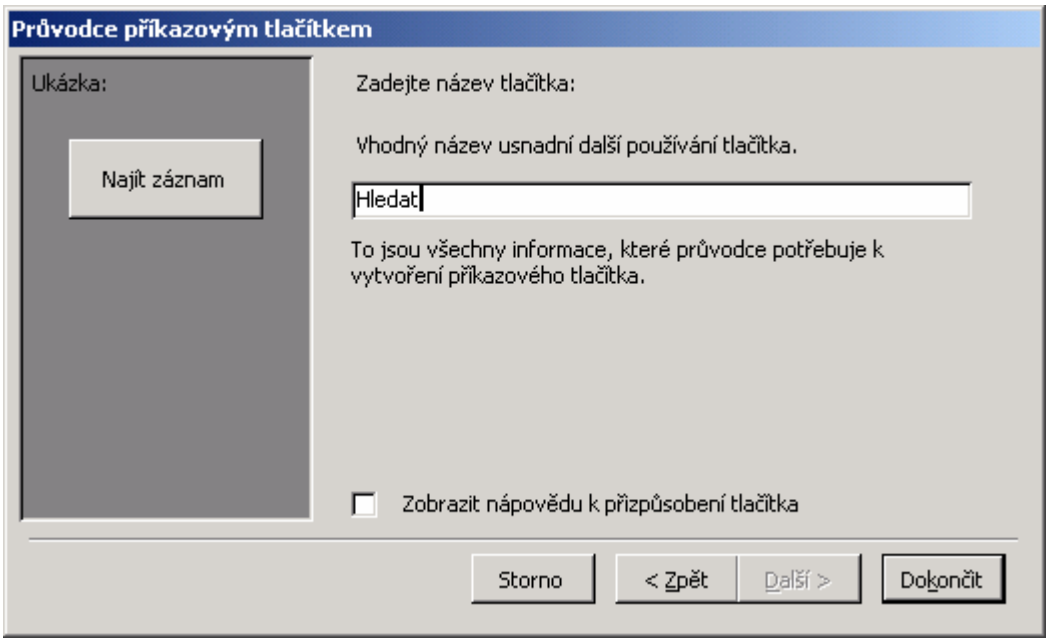


Obr. F7: Volba akce pro příkazové tlačítko



Obr. F8: Volba textového nebo grafického označení příkazového tlačítka

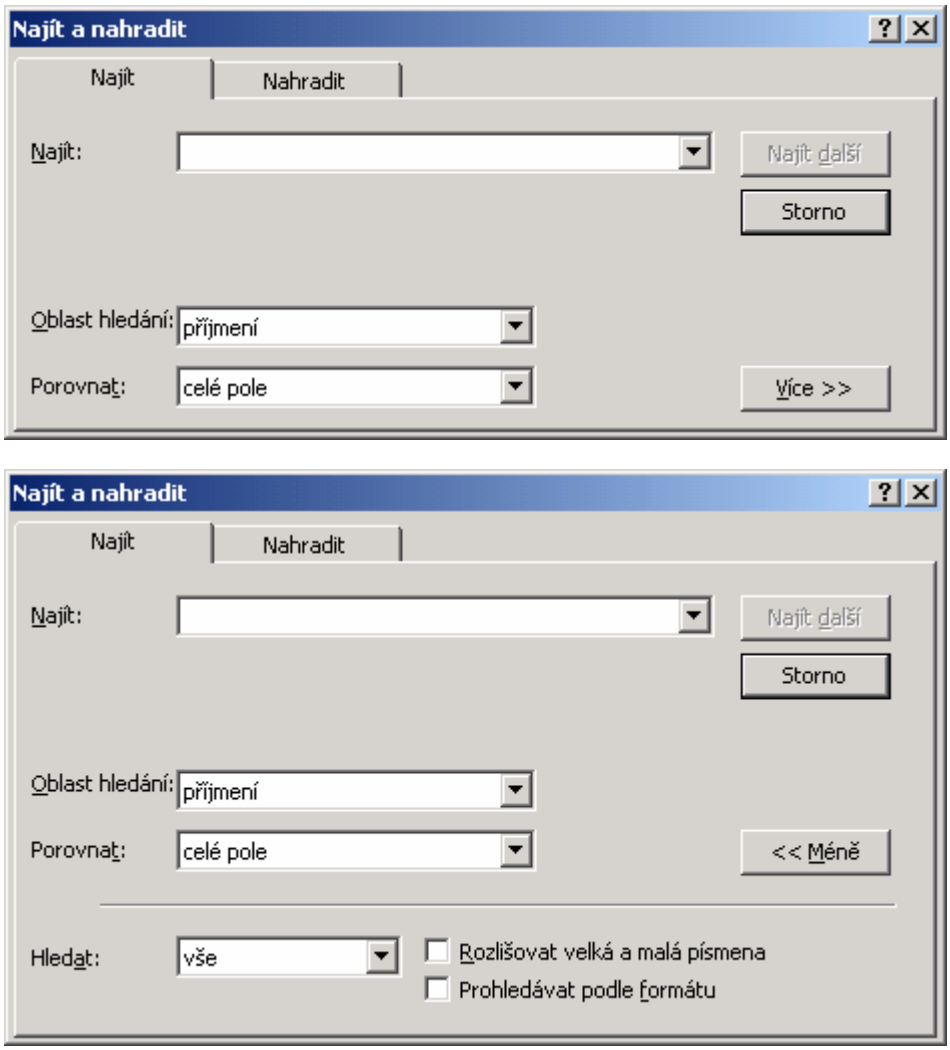
Jestliže nyní formulář převedeme z návrhového zobrazení do datového zobrazení a ve formátových vlastnostech formuláře máme vlastnost Výchozí zobrazení nastavenu na jednoduchý formulář, pak ve formuláři objeví hodnoty polí jednoho záznamu (tj. jednoho řádku podkladové tabulky). Po záznamech je možné se pohybovat pomocí navigačních tlačítek ve spodní části formuláře. Lze volit posun na 1. záznam, předchozí, následující a poslední záznam a dále na nový záznam, jehož hodnoty pak ve formuláři zadáme.



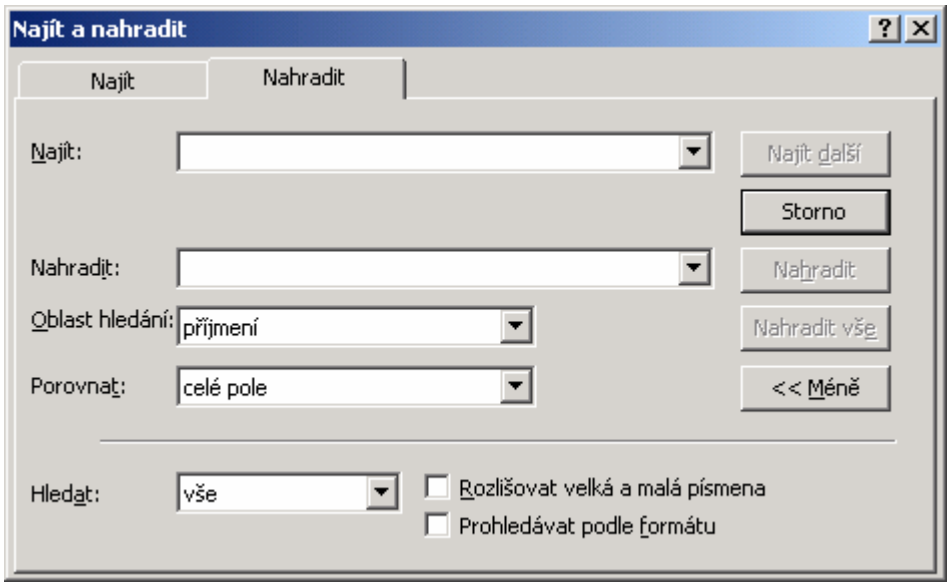
Obr. F9: Přiřazení interního jména tlačítku

Obr. F10: Zobrazení typu Jednoduchý formulář

Jestliže v našem formuláři z obr. F10 stiskneme průvodcem vytvořené příkazové tlačítko Najít záznam, otevře se další okno, v němž pak zadáváme, co chceme hledat. Všimněme si, že v obr. F10 stál kurzor v poli příjmení, a proto v oknech na obr. F11 se očekává prohledávání zadaného textu ve sloupci příjmení podkladové tabulky formuláře. Můžeme však v rozbalovacím seznamu Oblast hledání prohledávaný sloupec tabulky změnit., Druhé z dialogových oken na obr. F11 se otevřelo po stisku tlačítka Více >> v prvním z nich. Obsahuje možnost zadání podrobnějších specifikací režimu prohledávání, jak je z obrázku patrné. V obr. F11 je rovněž vidět, že dialogová okna obsahují záložku Nahradit. Jestliže ji stiskneme, otevře se stránka dialogu rozšířená o možnost zadání nových údajů pro nalezené hodnoty, které původní hodnoty nahradí.



Obr. F11: Hledání záznamu

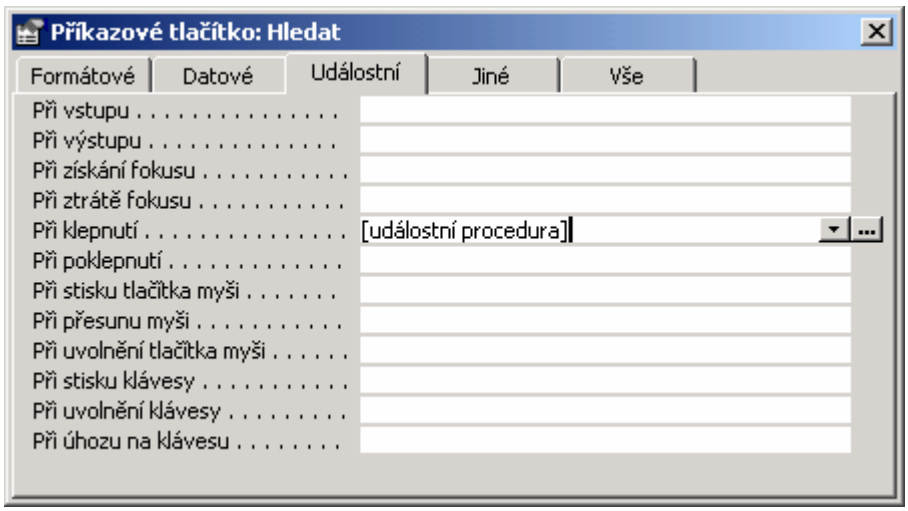


Obr. F12: Nahrazování nalezených údajů novými hodnotami

Nyní se podívejme na zdrojový kód událostní procedury, která se generuje po stisku tlačítka Najít záznam.

Zpřístupníme jej tak, že v návrhovém zobrazení formuláře příkazové tlačítko vybereme, pak stiskneme pravé tlačítko myši, to způsobí rozbalení lokálního menu a v něm volíme volbu Vlastnosti. (Vlastnosti můžeme zpřístupnit i bez použití lokálního menu, jestliže stiskneme v horním panelu nástrojů ikonu Vlastnosti.) Vybereme stránku událostních vlastností, na niž je vidět, že byla definována vlastnost Při klepnutí. Jestliže klikneme myší v řádku této vlastnosti, objeví se v něm na konci dvě značky, viz obr. F13, a po kliknutí na značce ... se objeví okno s kódem událostní procedury.

Kratším způsobem zpřístupnění kódu je kliknout na ikonu Kód v horním panelu nástrojů v návrhovém zobrazení formuláře. Je zde však rozdíl v tom, že v prvním případě se kurzor v okně kódu přesune na oblast událostní procedury zkoumaného tlačítka, zatímco v druhém případě na začátek okna se zdrojovými texty všech událostních procedur prvků formuláře, resp. na místo, kde v tomto okně kurzor naposled stál. Pak musíme příslušný text událostní procedury příkazového tlačítka v okně kódu nejdříve najít.



Obr. F13: Událostní vlastnosti příkazového tlačítka

I když jsme Visual Basic zatím neprobírali, dá se kód intuitivně pochopit. Vypadá následovně:

```
Private Sub Hledat_Click()
On Error GoTo Err_Hledat_Click

    Screen.PreviousControl.SetFocus
    DoCmd.DoMenuItem acFormBar, acEditMenu, 10, , acMenuVer70

Exit_Hledat_Click:
    Exit Sub

Err_Hledat_Click:
    MsgBox Err.Description
    Resume Exit_Hledat_Click

End Sub
```

Záhlaví událostní procedury Private Sub vyjadřuje, že jde o lokální proceduru, kterou lze volat pouze z formuláře, kde byla definována. (Globální procedury volatelné z libovolného místa musí být uloženy ve zdrojovém textu vytvořeném v prostředí Moduly.) Název procedury je automaticky vytvořen z názvu příkazového tlačítka uživatelem definovaným v předposledním kroku Průvodce příkazovým tlačítkem (obr. F9) a názvem událostní procedury v originálním anglickém vyjádření (Click = kliknutí, klepnutí, stisknutí).

Příkaz On Error GoTo *návěští* provede skok na specifikované návěští, dojde-li k nějaké chybě (v našem případě např. hledaný záznam nebyl nalezen), jinak se pokračuje následujícím příkazem. Při vyhodnocení chyby se příkazem MsgBox Err.Description generuje v samostatném okně zpráva s textovým popisem, k jaké chybě došlo, a po zavření tohoto okna tlačítkem OK, které zde slouží k potvrzení toho, že se uživatel se vzniklou chybou seznámil, se příkazem Resume *návěští* předá řízení na další návěští, za nímž následuje předčasné ukončení procedury příkazem Exit Sub.

Jestliže k chybě nedošlo, pokračuje provádění procedury příkazem Screen.PreviousControl.SetFocus, který zajistí návrat kurzoru na prvek, na němž stál před tím, než bylo stisknuto tlačítko Najít záznam.

Následující příkaz DoCmd.DoMenuItem acFormBar, acEditMenu, 10, , acMenuVer70 volá volbu z roletového menu Úpravy (v anglickém Accessu se nazývá Edit, proto se příslušný parametr nazývá acEditMenu), která je k dispozici při práci s formuláři (acFormBar), číslo 10 reprezentuje pořadí volby v roletě a poslední parametr acMenuVer70 skutečnost, že se jedná o soustavu roletových menu verze Access97 a vyšší.

2.3.3. Definování relace 1:N mezi 2 tabulkami

Mezi tabulkami se často vyskytuje vztah (relace) mezi některými položkami a tento vztah může být typu 1:1, 1:N a M:N.

V prvním případě to znamená, že jednomu záznamu v první tabulce přísluší (nejvýše) jeden záznam v druhé tabulce, např. muž u nás může mít v daný okamžik nejvýše jednu manželku.

Relace 1:N vyjadřuje to, že jednomu záznamu v jedné tabulce může příslušet více záznamů v tabulce druhé, např. čtenář může mít vypůjčeno více knih, objednávka může obsahovat více položek, osoba v databázi může ovládat více cizích jazyků apod.

Konečně relace M:N je vlastně relací 1:N a N:1 zároveň. Např. při vztahu mezi učiteli a vyučovanými předměty může každý učitel vyučovat více předmětu a současně každý předmět může být vyučován více učiteli. Podobně při rezervacích knih v knihovně může čtenář mít rezervováno více knih a současně tatáž kniha může být rezervována více čtenáři.

Je zřejmé, že relace 1:1 je speciálním případem relace 1:N. Relaci M:N můžeme modelovat pomocí 2 relací 1:N a N:1 přes tzv. *průnikovou tabulku*, jejíž klíč je nejčastěji vícesegmentový a jeho segmenty tvoří klíče tabulek ze vztahu M:N. V příkladu s učiteli a předměty bychom mohli tento vztah popsat trojicí tabulek:

UČITELÉ(IDuč, příjmením jméno, …)
PŘEDMĚTY(IDpř, název, ročník, semestr, …)
UČ-PŘ(IDuč, IDpř, úvazek)

Průnikovou tabulkou je zde tabulka UČ-PŘ.

Z uvedeného plyne, že nám stačí vědět, jak se v MS Accessu realizuje vztah 1:N.

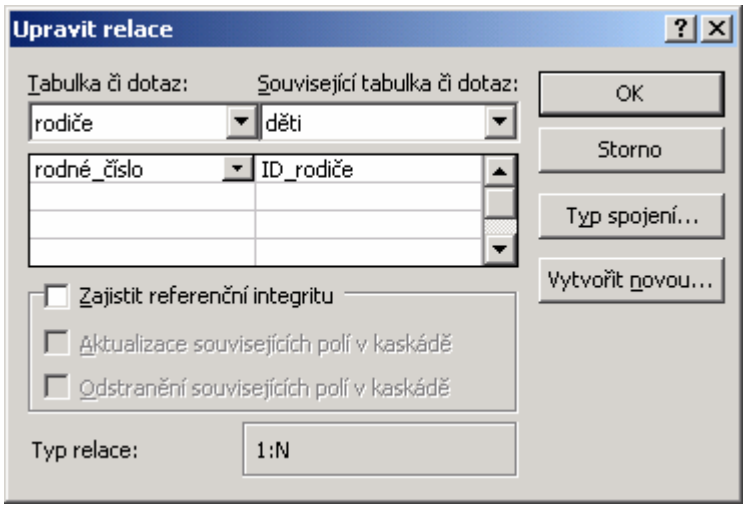
Příklad:

Uvažujme následující dvě tabulky

RODIČE(rodné-číslo, jméno, příjmení, …)
DĚTI(ID-rodiče, jméno-dítěte)

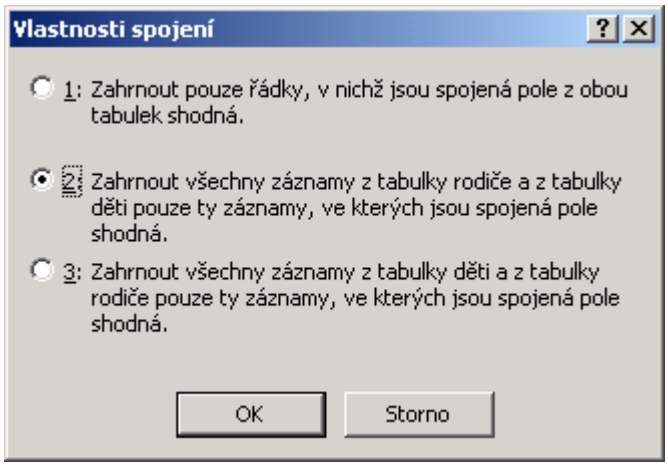
Budeme předpokládat, že v druhé tabulce jsou děti přiřazeny ke svému rodiči prostřednictvím informace o identifikačním čísle rodiče (jeho rodné číslo) a přitom předpokládáme, že v databázi máme pouze jednoho rodiče, např. uvažujeme vztah rodičovství svých kolegů, z nich žádný nemá na pracovišti svého partnera. V první tabulce je klíčem evidentně rodné číslo, v druhé tabulce tomu tak není, protože osoba může mít více dětí, a tedy její identifikační číslo se může vyskytovat na více řádcích. To znamená, že se jedná o vztah 1:N. Předpokládáme-li, že nikdo si nepojmenuje dvě nebo více svých dětí stejným jménem, je jednoznačně určující informací o řádku druhé tabulky dvojice obou jejich položek, která tak tvoří dvousegmentový klíč. (Jinou možností, jak volit klíč druhé tabulky, je i pro ni definovat identifikační údaj pomocí rodného čísla, zde tedy rodného čísla dítěte.)

Definici relace 1:N v MS Accessu zahájíme kliknutím na ikonu Relace  v horním panelu nástrojů nebo výběrem volby Relace… z roletového menu Nástroje. Objeví se okno se seznamem všech tabulek (a uložených dotazů) aktuální aplikace, z nichž tlačítkem Přidat do okna relace vložíme ty, pro které chceme relaci definovat. Relaci mezi souvisejícími položkami tabulek vytvoříme tak, že položku ze strany 1 první tabulky táhneme myší na související položku ze strany N druhé tabulky a potom levé tlačítko myši uvolníme. To způsobí, že se vyvolá dialog z obr. F14, v němž upřesňujeme vlastnosti relace. Jsou zde již vyplněny položky, které jsme v předchozím kroku dali do vztahu.



Obr. F14: Definice vlastností relace 1:N v MS Accessu

Jestliže stiskneme tlačítko Typ spojení… , zobrazí se dialog z obr. F15, v němž specifikujeme *povinné*, resp. *volitelné členství ve vztahu*, to znamená, zda se do vztahu zahrnou všechny záznamy tabulky, případně pouze ty, pro které existuje záznam se shodnou hodnotou související položky v druhé tabulce. V našem příkladu volíme prostřední možnost, kdy chceme zahrnout všechny osoby, i když někteří mohou být bezdětní. Po potvrzení této volby se vrátíme do dialogu na obr. F14.



Obr. F15: Definice typu spojení v relaci 1:N

Důležitou vlastností, kterou můžeme pro relaci 1:N definovat je *referenční integrita*. Jestliže zatrhneme zaškrtávací pole Zajistit referenční integritu, zpřístupní se další dvě zaškrtávací pole: Aktualizace souvisejících polí v kaskádě a Odstranění souvisejících polí v kaskádě. Znamená to, že v případě zatržení prvního pole se při opravě údaje na straně 1 automaticky opraví na novou hodnotu tento údaj i ve všech N výskytech související položky strany N. Při zatržení druhého pole se při zrušení záznamu v tabulce na straně 1 automaticky zruší všech N záznamů tabulky na straně N, které mají hodnotu související položky shodnou s údajem rušeného záznamu.

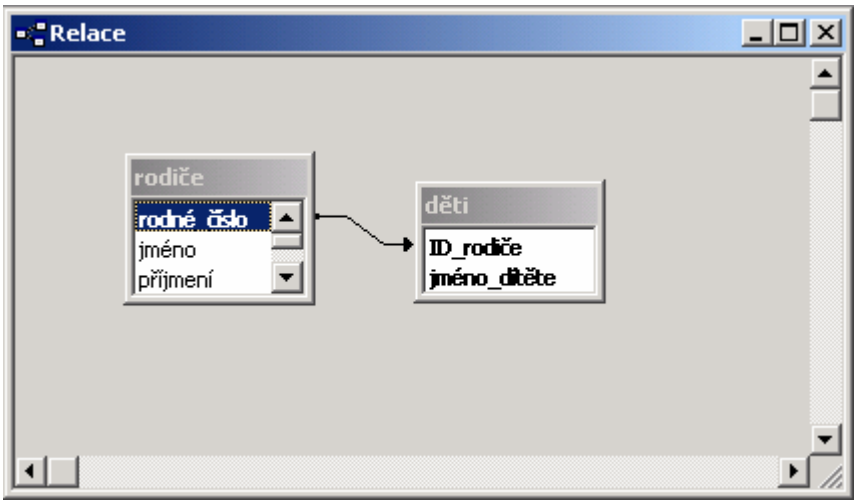
Poznámka:

To, zda některou z referenčních integrit použijeme, závisí na aplikaci.

Kdybychom měli např. relaci 1:N mezi tabulkami čtenářů a jejich výpůjčkami, pak by bylo nesprávné volit referenční integritu pro rušení. Kdyby totiž čtenář zrušil své členství v knihovně a my jsme o něm zrušili záznam v evidenci, pak by se automaticky zrušili všechny záznamy o jeho výpůjčkách a on by si vypůjčené knihy, které ještě nevrátil, mohl beztrestně nechat. Je zřejmé, že čtenáře můžeme v evidenci zrušit až poté, když všechny vypůjčené knihy vrátí. Na druhé straně, jestliže by došlo k přečíslování jeho členského čísla, pak je potřebné, aby se tento údaj opravil i ve všech výpůjčkách. Automaticky to zajistíme zatržením aktualizační referenční integrity. ♦

V našem příkladě rodičů a jejich dětí jsme nepoužili ani jednu referenční integritu. Protože zemře-li např. rodič dítěte a my jej z tabulky odstraníme, nechceme však s ním odstranit i jeho děti, protože ty žijí dál (pokud všichni nezemřeli při nějaké nehodě, což je čistě teoretická možnost, ne však obecný případ). Aktualizační referenční integrita zde rovněž nepřipadá v úvahu, protože rodná čísla, přes něž vážeme tabulky, se nemění.

Po definici všech potřebných vlastností relace dialog z obr. F14 ukončíme stiskem tlačítka OK. V okně relace se mezi oběma tabulkami objeví orientovaná hrana se směrem od strany 1 ke straně N. Pokud byla zvolena nějaká referenční integrita, je na straně N u šipky připsáno ohodnocení číslem ∞ (nekonečno).



Obr. F16: Relace 1:N bez referenční integrity

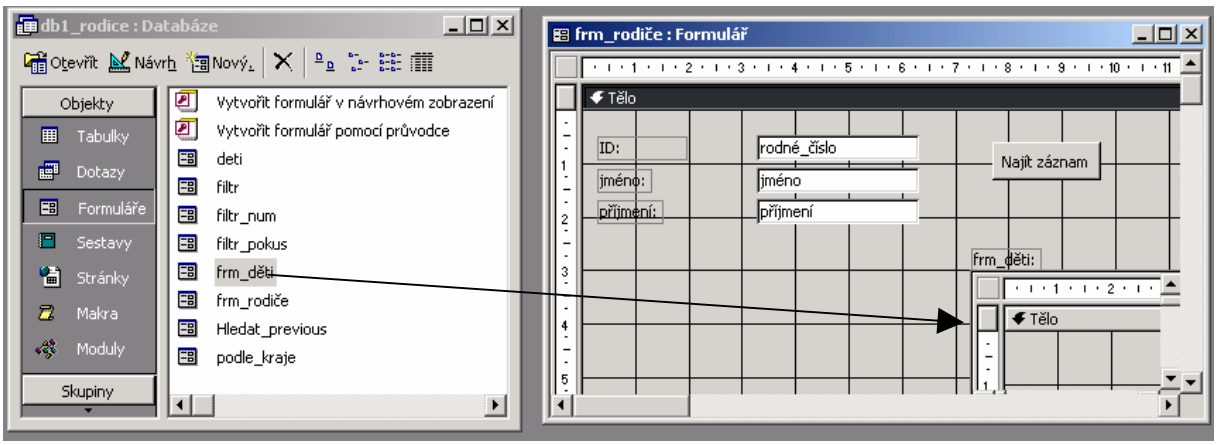
2.3.4. Formulář s podformulářem

Relaci 1:N využijeme ke konstrukci dvou formulářů, kdy druhý formulář je vnořen do prvního, a proto se mu také říká *podformulář*. Formulář obsahuje podkladovou tabulku ze strany 1 a podformulář podkladovou tabulku ze strany N.

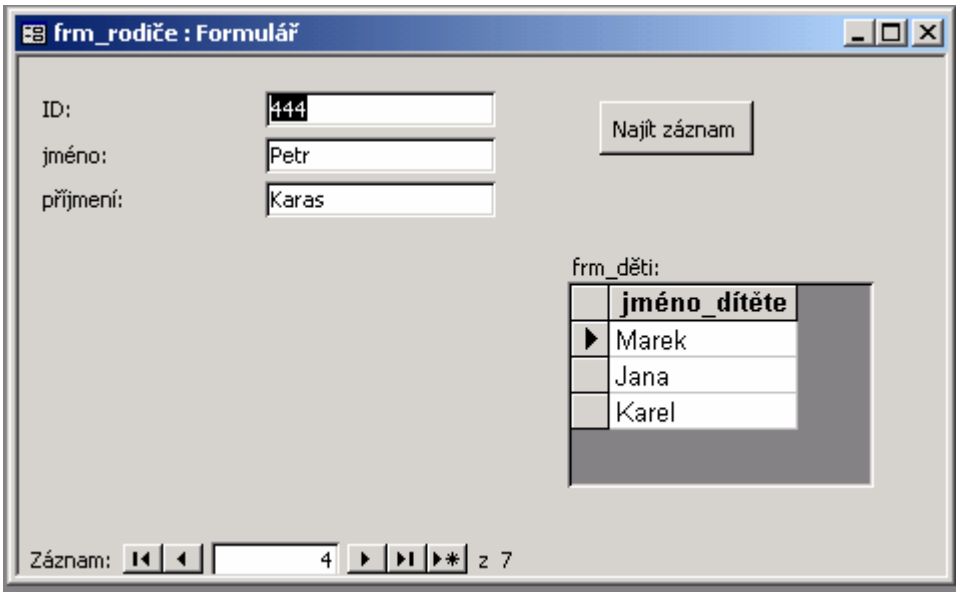
Podformulář lze tvořit uvnitř rodičovského formuláře prostřednictvím prvku  (Podformulář či podsestava) ze soupravy nástrojů formuláře nebo jej vytvoříme jako samostatný formulář, který pak do rodičovského formuláře vložíme. Popíšeme druhý způsob. Protože budeme chtít, aby při pohybu po záznamech osob v rodičovském formuláři se v podformuláři zobrazovaly všechny jejich děti bez toho, že bychom je postupně procházeli navigačními tlačítky, musíme ve formátových vlastnostech formuláře změnit implicitní výchozí zobrazení jednoduchý formulář na zobrazení datový list, kdy ve formuláři jsou jednotlivé řádky tabulky zobrazeny pod sebou, jako je tomu v tabulce.

Protože děti jedné osoby budeme chtít v podformuláři pouze zobrazovat, nepředpokládáme zde žádné přidávání ani aktualizace a diky zobrazení datový list nepotřebujeme ani žádná navigační tlačítka a můžeme nastavit formátové vlastnosti podformuláře takto: Volič záznamů: ne, Navigační tlačítka: ne. Dále pak nastavíme na hodnotu „ne“ také datové vlastnosti podformuláře Povolit přidávání a Povolit úpravy.

Při vlastním spojení obou formulářů do jednoho celku nejdříve otevřeme v návrhovém zobrazení rodičovský formulář, který na obrazovce posuneme tak, aby bylo viditelné okno Databáze se seznamem existujících formulářů aktuální aplikace (obr. F17) a z něj pak přetáhneme název podformuláře do návrhového zobrazení rodičovského formuláře, např. do jeho pravého dolního rohu. To způsobí, že se v rodičovském formuláři objeví na příslušném místě podformulář v jeho skutečné velikosti. Výsledek spojení obou formulářů je vidět při datovém zobrazení rodičovského formuláře na obr. F18.



Obr. F17: Vkládání podformuláře frm_děti do rodičovského formuláře frm_rodiče



Obr. F18: Formulář s podformulářem

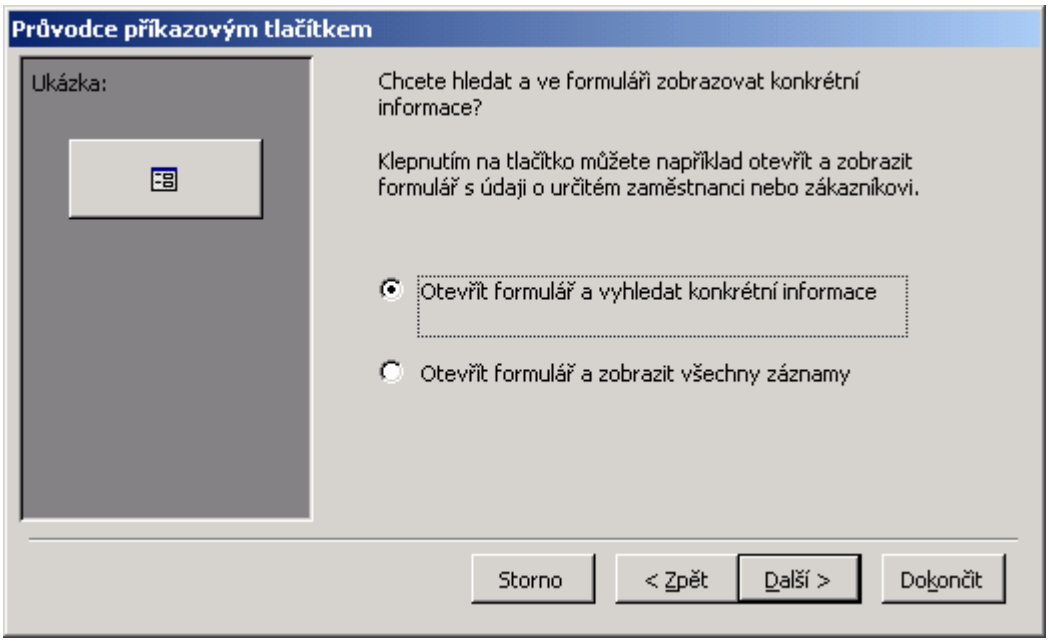
2.3.5. Otvírání dalších formulářů s filtrem

V následujícím příkladu ukážeme ještě jednu akci generovanou průvodcem, a sice otevření formuláře po stisku tlačítka s tím, že ve formuláři se objeví pouze ty záznamy, které budou splňovat specifikovanou podmínku. Opět využijeme Průvodce příkazovým tlačítkem, zvolíme kategorii Operace s formuláři a v ní akci Otevřít formulář.

Abychom demonstrovali tento přístup v kontrastu s konstrukcí formulář s podformulářem, odstraníme podformulář s rodičovského formuláře frm_rodiče a místo toho do něj zařadíme nové tlačítko, po jehož stisku budeme otvírat formulář frm_děti, který v předchozím výkladu plnil roli podformuláře. V tomto formuláři budeme pochopitelně opět žádat, aby se v něm zobrazili pouze děti, které patří k osobě z volajícího formuláře. Rozdíl

proti konstrukci formulář s podformulářem spočívá v tom, že seznam dětí zobrazíme v dalším formuláři až tehdy, když nás bude zajímat.

Po výběru otvíraného formuláře ze seznamu existujících formuláře aplikace průvodce zobrazí dialogové okno z obr. F19.



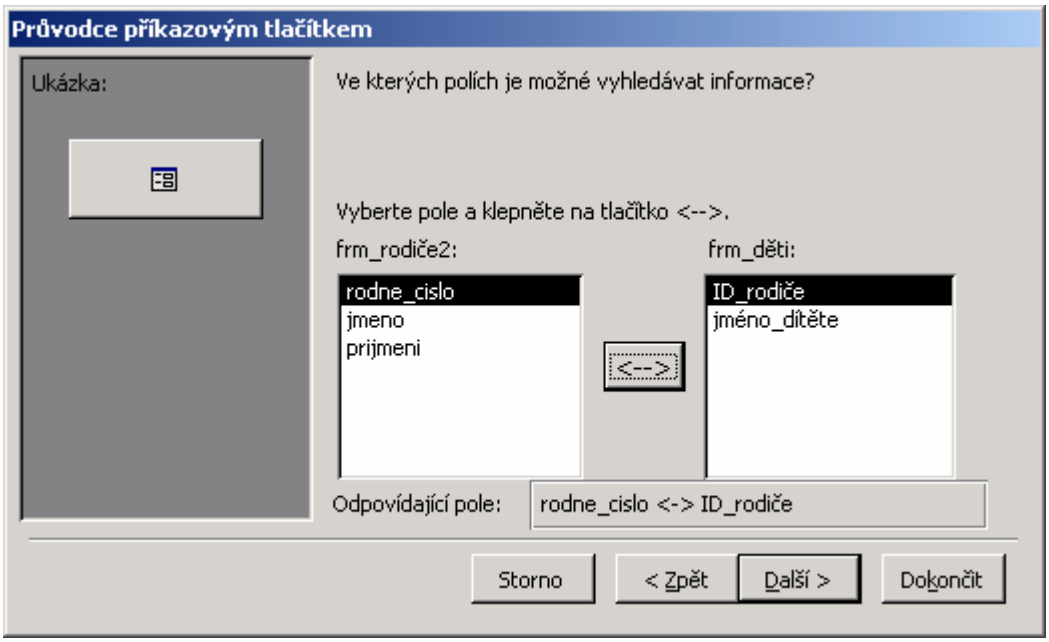
Obr. F19: Filtrovat záznamy v otvíraném formuláři?

V tomto dialogu se rozhodneme, zda ve volaném formuláři budou po jeho otevření dostupné všechny záznamy jeho podkladové tabulky anebo pouze některé z nich. V našem volíme variantu Otevřít formulář a vyhledat konkrétní informace, protože chceme zobrazit pouze děti zkoumané osoby. V následném dialogu na obr. F20 pak vybereme v seznamu podkladových tabulek volajícího a volaného formuláře položky, které se mají shodovat (filtr zajistí, že ve volaném formuláři se budou zobrazovat pouze ty záznamy, které se v odpovídajícím poli shodují s příslušnou hodnotou souvisejícího pole volajícího formuláře).

Vztah mezi oběma souvisejícími poli se definuje kliknutím na tlačítko 

, tím se vyvolá přepis příslušného vztahu vedle popisu Odpovídající pole: na obr. F20 dole.

Ve zbývajících dvou krocích průvodce se již jen (stejně jako v příkladě s hledáním záznamu po stisku tlačítka) zvolí označení tlačítka textem nebo grafickým symbolem a definuje se interní jméno tlačítka, jak bylo vidět z obr. F8 a F9.



Obr. F20: Definice souvisejících polí podkladových tabulek volajícího a volaného formuláře

Zdrojový kód událostní procedury po stisknutí tlačítka Otevřít formulář (tak jsme jej označili) a doplnění parametru acFormDS (proto, aby se formulář frm_děti otevřel v zobrazení datový list) je následující:

```
Private Sub Otevřít_frm_děti_Click()
On Error GoTo Err_Otevřít_frm_děti_Click

    Dim stDocName As String
    Dim stLinkCriteria As String

    stDocName = "frm_děti"

    stLinkCriteria = "[ID_rodiče]=" & "'" & Me![rodne_cislo] & "'"
    DoCmd.OpenForm stDocName, acFormDS, , stLinkCriteria

Exit_Otevřít_frm_děti_Click:
    Exit Sub

Err_Otevřít_frm_děti_Click:
    MsgBox Err.Description
    Resume Exit_Otevřít_frm_děti_Click

End Sub
```

Kód je opět pro studenty, kteří prošli programováním v předmětu *Informatika I* pochopitelný i bez znalosti Visual Basicu. Proti předchozímu kódu jsou zde navíc pouze deklarace dvou stringových (textových nebo také řetězcových) proměnných (v sekci Dim). Obě proměnné jsou naplněny přiřazovacími příkazy, do první z nich je uloženo jméno formuláře a do druhé filtrovací podmínka. Obě hodnoty přiřazované do těchto proměnných jsou rovněž typu string (text), a proto jsou výrazy na pravé straně obou přiřazovacích příkazů psány v uvozovkách.

Poznámka:

Pokud by uvnitř stringu měl být jiný string, pak se musí vymezit jinak než uvozovkami. K tomu účelu se používají apostrofy. To je použito v 2. přiřazovacím příkazu, kde apostrofy je ohraničena hodnota stringu Me![rodne_cislo]. ♦

Operátor & je určen ke zřetězení výrazů typu string. Hodnota Me![rodne_cislo] vyjadřuje, že jde o položku rodne_cislo v aktuálním formuláři (to je dáno označením Me!). Hranaté závorky zde plní pouze roli omezovače identifikátoru položky a zde, protože je tvořen jedním „slovem“ (dáno použitím podtržítka místo mezery), nejsou povinné. Vlastní otevření formuláře zajišťuje stěžejní příkaz procedury DoCmd.OpenForm …

Poznámka:

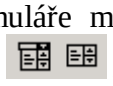
Filtrovací podmínka, s níž otvíráme formulář frm_děti, formálně odpovídá klauzuli **WHERE** výběrového dotazu **SELECT** v SQL jazyku. Jestliže bychom chtěli formulovat složitější filtrovací podmínku, museli bychom ji již dopsat ve zdrojovém kódu událostní procedury ručně. Průvodce totiž umožňuje definovat jen vztah rovnosti mezi jedinou dvojicí souvisejících polí podkladových tabulek obou formulářů. ♦

2.3.6. Seznamy

Častou součástí formulářů jsou prvky *seznam* (List Box) a *pole se seznamem* (Combo Box). První typ je rozbalený seznam, který je určen k přímému zobrazení všech (nebo většího počtu) hodnot prvků seznamu, druhý pak představuje rozbalovací seznam, který má tvar jednoho řádku se značkou napravo, na kterou když klikneme, tak se teprve seznam hodnot rozbalí. V obou případech, pokud seznam hodnot se nevejde do vyhrazené výšky okna pro seznam, to znamená, že seznam bude doplněn svislou rolovací lištou. Pole se seznamem má výhodu, že ve formuláři zabírá méně místa (jen jeden řádek) a rozbalí se až když to potřebujeme.

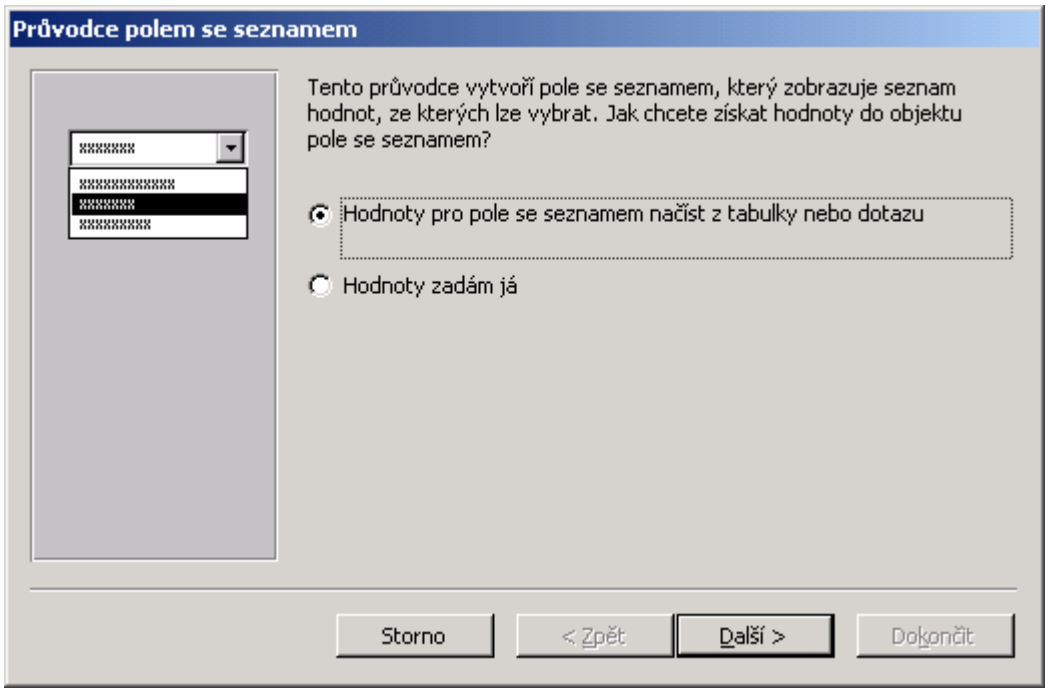
Prvky seznamu mohou mít přesně daný počet, např. u položky pohlaví máme možnosti muž, žena; vzdělání rozlišujeme na základní, středoškolské a vysokoškolské; kraje v České republice představují 14-ti prvkový seznam apod. V těchto případech je možné seznam plnit statickým výčtem hodnot.

Častěji ovšem délka seznamu může být proměnná, např. seznam oborů v knižním fondu knihovny se může lišit v závislosti na specializaci knihovny, její vybavenosti, místních podmínkách atd. V tomto případě je vhodnější pro obory, do nichž knihy patří, definovat samostatnou tabulku, tzv. *číselník*, která bude mít 2 položky: kód oboru a název oboru. (Pro vztahy 1:N atd. je vhodnější používat kód oboru, pro zobrazování pak jeho název.) Z číselníku naplníme seznam dynamicky načtením prostřednictvím SQL dotazu. I když SQL jazyk jsme zatím neprobírali, je možné příslušný výběrový dotaz, sloužící jako zdroj dat seznamu, vytvořit pomocí průvodce.

V soupravě nástrojů formuláře mají prvky Pole se seznamem a Seznam podobu následujících grafických symbolů  .

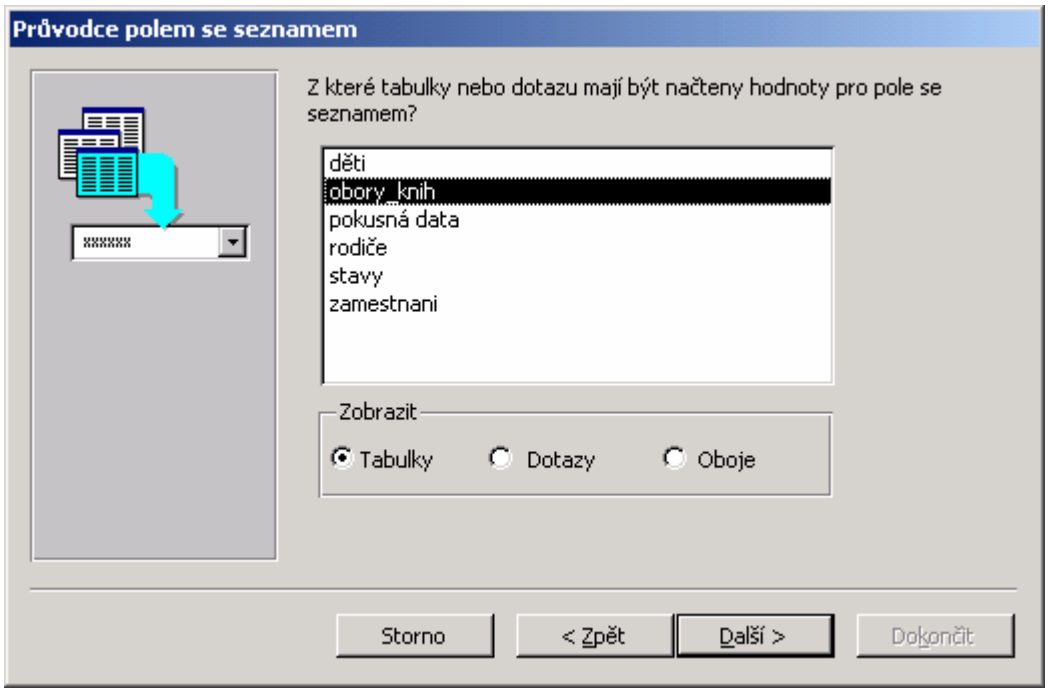
Protože všechny kroky průvodců pro oba typy seznamů jsou stejné, stačí se zabývat pouze jedním z nich. Zvolíme si třeba rozšířenější Pole se seznamem. Po kliknutí na jehi grafický symbol v soupravě nástrojů se aktivuje dialogové okno z obr. F21, které umožňuje

volit, zda budeme zadávat hodnoty statického seznamu (volba Hodnoty zadám já) anebo budou načítány výběrovým dotazem z nějaké tabulky nebo výsledku jiného dotazu.



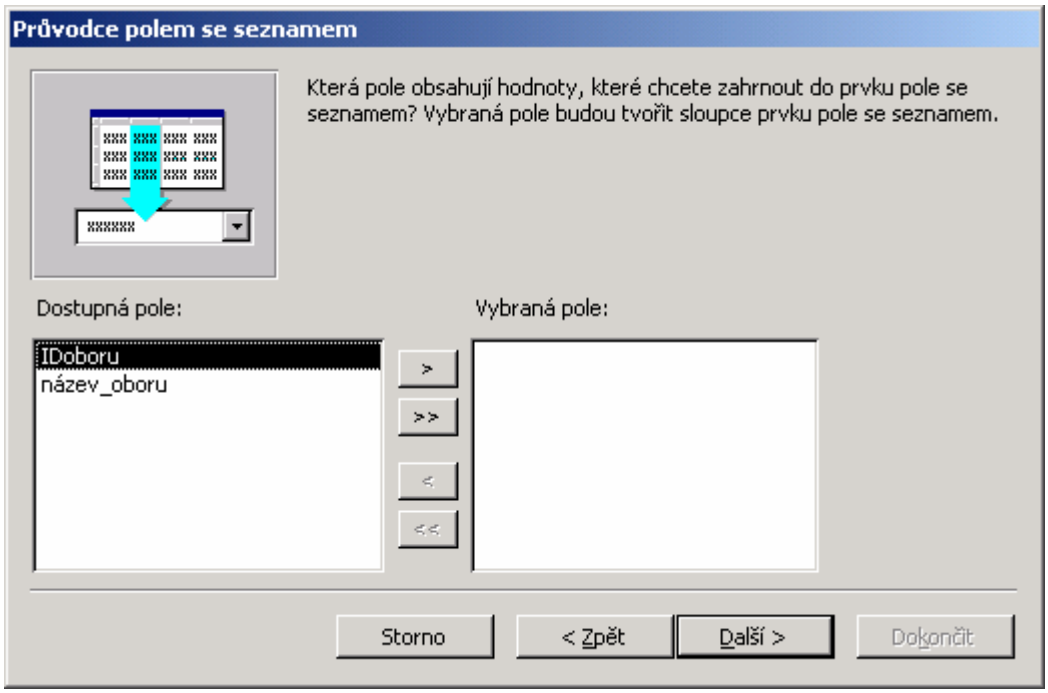
Obr. F21: Volba vytváření pole se seznamem dynamicky nebo staticky

Zvolme obecnější první variantu. Po stisku tlačítka Další > v dialogu podle obr. F21 se vyvolá dialogové okno, kde volíme zdroj dat pro pole se seznamem (viz obr. F22), to znamená sloupec (nebo i více sloupců) tabulky nebo některého dříve definovaného dotazu.

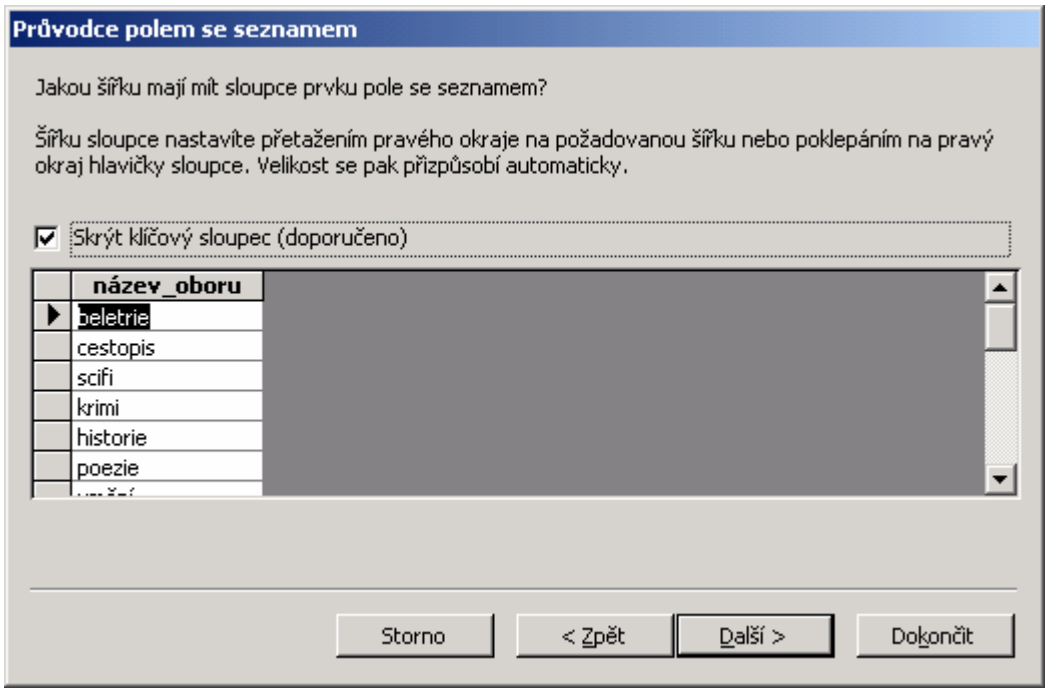


Obr. F22: Výběr zdroje dat pro pole se seznamem

Po výběru tabulky přejdeme do dalšího dialogu, kde vybíráme sloupec nebo sloupce, které se mají v seznamu objevit. Pro zvolenou tabulku obory_knih přesuneme do seznamu obě položky ID_oboru i název_oboru. Zajistíme to buď postupným použitím tlačítka > nebo naráz tlačítkem >> (obr. F23). Obě položky se tím přesunou do sloupce Vybraná pole:



Obr. F23: Výběr sloupců tabulky do pole se seznamem



Obr. F24: Zobrazení hodnot seznamu s možností odkrýt klíčový sloupec.

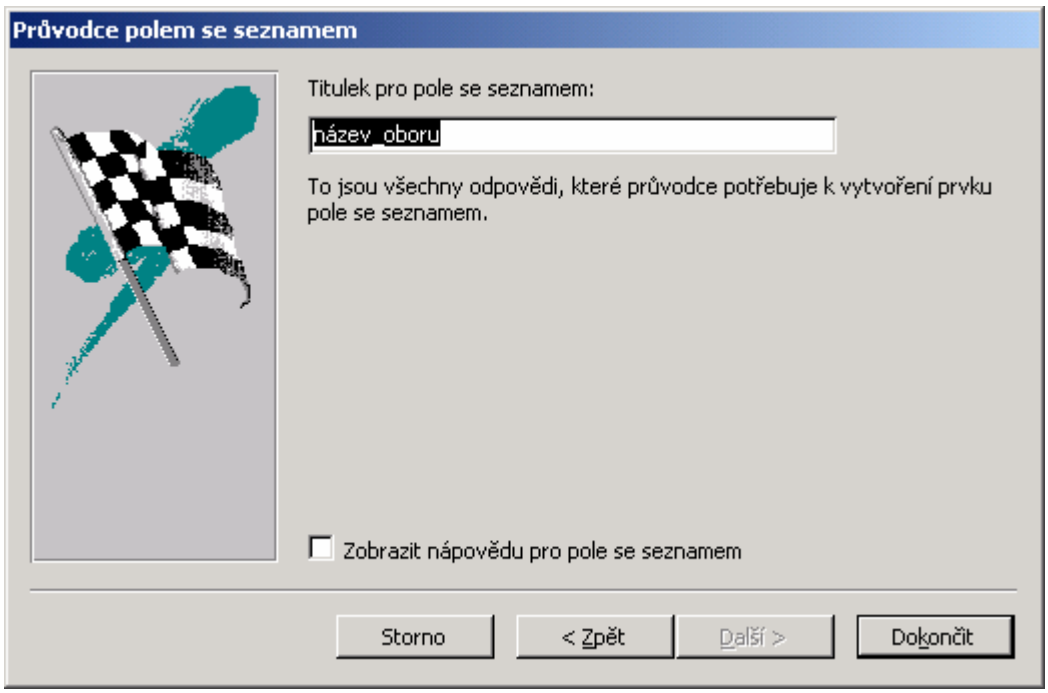
po ukončení dialogu z obr. F23 se dostaneme do situace na obr. F24, kde vidíme seznam hodnot, ale bez klíčového sloupce, který je skryt. Zrušíme-li zatržení zaškrtávacího pole, klíčový sloupec se odkryje. Většinou to není nutné. Někdy však v situaci, kdy průvodce používáme jen jako pomocný nástroj, kterým vytvoříme hrubý tvar SQL dotazu pro pole se seznamem, který pak ručně doladíme, je vhodné klíčový sloupec odkrýt, abychom jej mohli ve vlastnostech pole se seznamem zrušit.

Poznámka:

Jestliže je seznam tvořen více sloupci (tím myslíme i případ z obr. F24, kdy je vidět pouze jeden sloupec a druhý – klíčový sloupec – je skryt), pak návratovou hodnotou výběru v seznamu je hodnota v klíčovém sloupci příslušného řádku, který byl v seznamu vybrán.

Jestliže zdrojová tabulka seznamu nemá definován (primární) klíč, pak v dialogu z obr. F24 chybí zaškrtávací políčko Skrýt klíčový sloupec a po něm následuje ještě jeden dialog, kde určíme, který ze zahrnutých sloupců bude definovat návratovou hodnotu výběru v seznamu. ♦

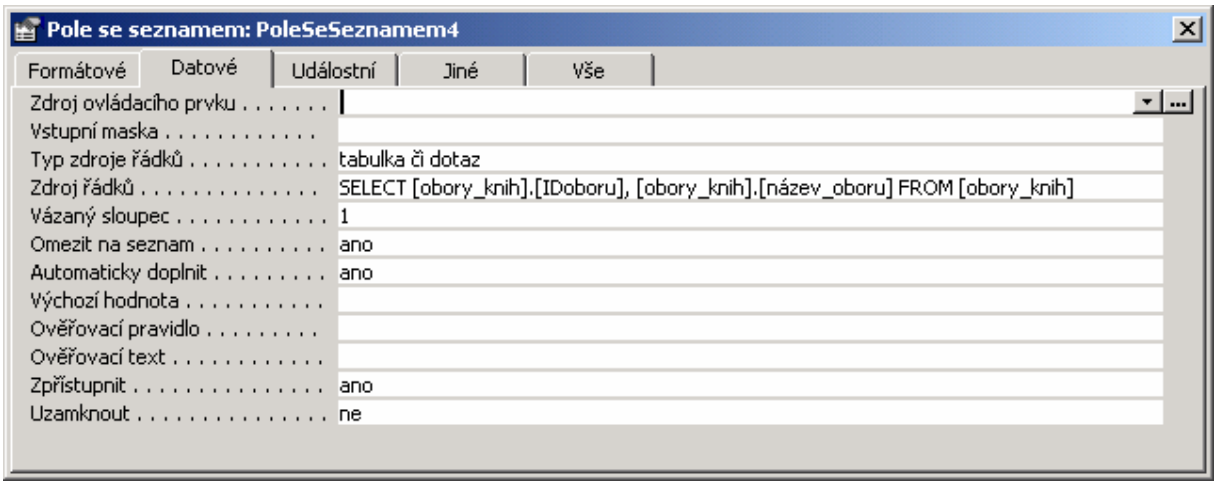
V posledním kroku (po určení sloupce s návratovou hodnotou) se již pouze určí titulek pro pole se seznamem a celý postup vedený průvodcem se ukončí.



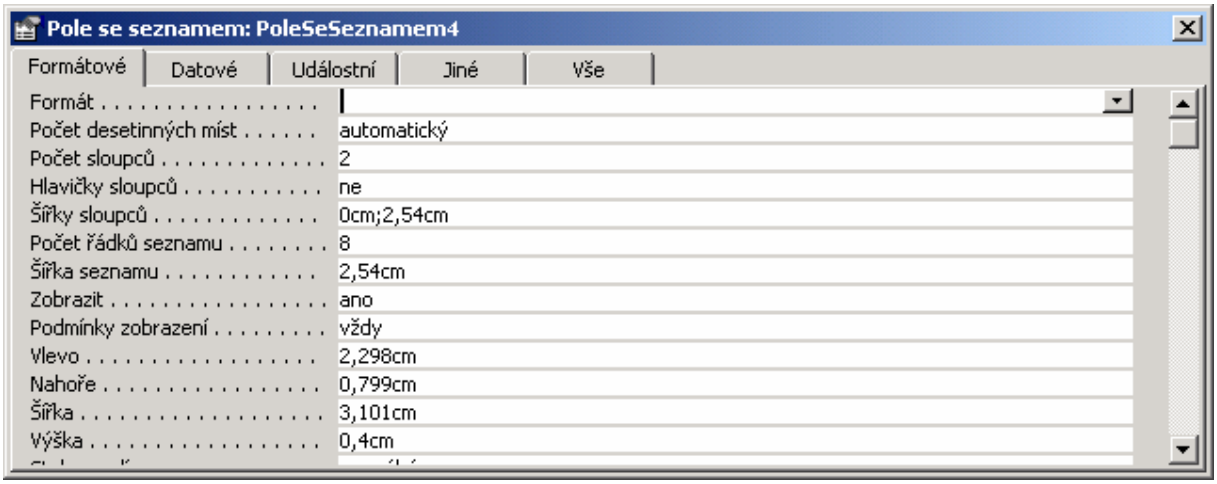
Obr. F25: Určení titulku pro pole se seznamem.

Zbývá ještě ukázat, že zdrojem řádků seznamu je výběrový dotaz jazyka SQL. Najdeme jej v datových vlastnostech pole se seznamem, viz obr. F26. Z jeho zápisu je patrné, že výsledkem dotazu jsou 2 sloupce z tabulky obory_knih. Vlastnost Vázaný sloupec s hodnotou 1 zde vyjadřuje, že 1. sloupec bude určovat návratovou hodnotu při výběru v seznamu.

Ve formátových vlastnostech pole se seznamem (obr. F27) je vidět např. počet sloupců v seznamu a jejich šířky. Skrytý klíčový sloupec má šířku 0.



Obr. F26: SQL dotaz ve zdroji řádků pole se seznamem.



Obr. F27: Formátové vlastnosti pole se seznamem.

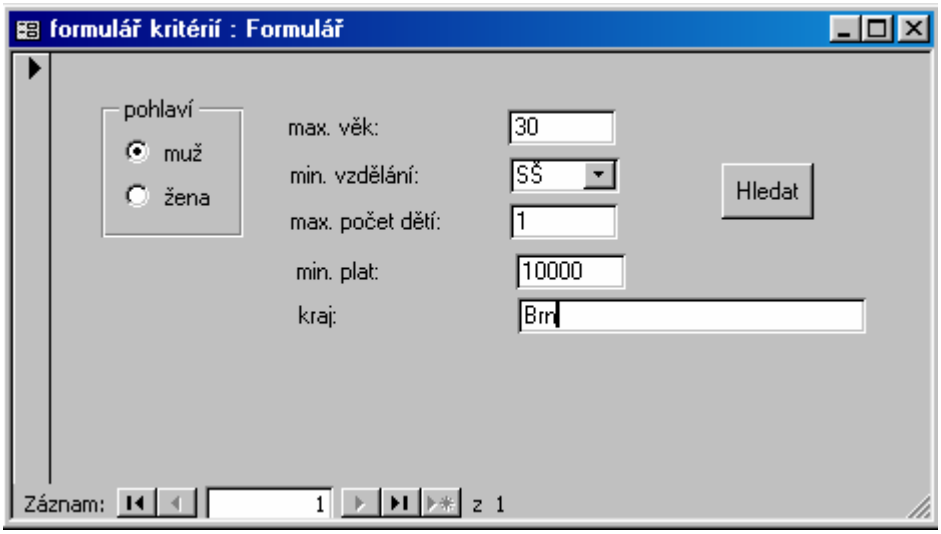
2.3.7. Formulář kritérií bez podkladové tabulky

Ve zdrojovém kódu za obr. F20 jsme řešili otevření nového formuláře s jednoduchým filtrem na jeho záznamy. Nyní uvedeme mnohem obecnější případ, kdy všechna výběrová kritéria jsou dána minimálními, resp. maximálními přípustnými hodnotami, popř. přesně požadovanými hodnotami sledovaných údajů. Tyto mezní hodnoty zadáváme do polí tzv. *kriteriálního formuláře*, který nemá podkladovou tabulku. Po stisku tlačítka se otevře nový formulář s podkladovou tabulkou, z níž budou k dispozici pouze ty řádky, které vyhovují podmínkám porovnání skutečných hodnot tabulky s hodnotami kritérii. Náš požadavek na filtraci dat ještě rozšíříme na případ, kdy některá kritéria nemusíme v kriteriálním formuláři ani vyplnit a pak to znamená, že na hodnotě odpovídajících položek nezáleží.

Příklad:

Vyhledejme z databáze osob možné partnery, kteří splňují požadované představy o maximálním věku, minimálním vzdělání, maximálním počtu dětí, minimálním platu a místě bydliště. U číselných údajů postupujeme tak, že nezáleží-li nám na nich, zvolíme velmi

velkou nebo naopak velmi malou hodnotu v závislosti na smyslu údaje. Např. pokud nám nezáleží na počtu dětí, zvolíme v tomto údaji velkou hodnotu, pokud není rozhodující výše platu, zadáme naopak hodnotu velmi malou, apod. Nezáleží-li nám na hodnotě znakového údaje, pak nezadáme nic a musíme zajistit, že se to bude interpretovat jako cokoliv.



Obr. F28: Formulář kritérií.

Kriteriální formulář vidíme na obr. F28. V návrhu byl použit prvek Skupina voleb, o kterém jsme zatím nemluvili. Je tvořen z kroužků, z nichž nejvýše jeden je vyplněn. Na obr. 28 slouží k přepínání muž nebo žena.

Poznámka:

Prvek Skupina voleb se skládá z několika částí – z vlastních kroužků, jejich popisek, rámečku, který je skupinu voleb ohraničuje a návěští v záhlaví rámečku. Je důležité si pamatovat, že návratová hodnota výběru je uchována v identifikátoru ohraničujícího rámečku skupiny voleb. Hodnoty výběru jsou 1, 2, atd. podle toho, zda byla nastavena 1. nebo 2. poloha atd. ♦

Postup řešení bude stejný jako v odstavci 2.3.5. Pomocí průvodce vytvoříme událostní proceduru při stisknutí tlačítka na otevření dalšího formuláře s jednoduchým filtrem jeho podkladové tabulky, který později předefinujeme. Výsledek je uveden na následujících řádcích. Pro zkrácení výpisu, zde nejsou uvedeny příkazy ošetřující vznik chyby.

O uvozovkách a apostrofech platí to, co již bylo zmíněno v poznámce na str. 27. Mezera a podtržítko na konci řádku signalizují, že příkaz pokračuje na dalším řádku. Mezereea před podtržítkem nesmi chybět, jinak by Access vyhodnotil chybu. Při srovnání s textovou informací místa bydliště je místo rovnosti použit operátor **LIKE**, který umožňuje použít i tzv. hvězdičkovou konvenci.

```
Private Sub Hledat_Click()
    ...
    Dim stDocName As String, stLinkCriteria As String
    stDocName = "osoby"
```

```
    stLinkCriteria ="[pohlaví] =" & "'" & IIf(Me![Frame0] = 1, "muž", "žena") & "'" _
                                  & " AND [věk] <=" & Me![maxvek] _
                                  & " AND [vzdělání] =" & "'" & Me![Combo11] & "'" _
                                  & " AND [počet dětí] <=" & Me![maxdeti] _
                                  & " AND [místo] LIKE " & "'" & Me![místo] & "*'" _
                                  & " AND [plat] >=" & Me![minplat]
    DoCmd.OpenForm stDocName, , , stLinkCriteria
    …
End Sub
```

Poznámka:

Po určité praxi budeme schopni psát zřetězení dílčích podmínek s menším počtem operandů, např. nebudeme apostrofy vyčleňovat do samostatných řetězců.
Zjednodušení zápisu filtrovací podmínky je vidět z následující ukázky.

```
stLinkCriteria = "[pohlaví] ='"   & IIf(Me![Frame0] = 1, "muž", "žena") & "'" _
                                  & " AND [věk] <=" & Me![maxvek] _
                                  & " AND [vzdělání] ='" & Me![Combo11] & "'" _
                                  & " AND [počet dětí] <=" & Me![maxdeti] _
                                  & " AND [místo] LIKE '" & Me![místo] & "*'" _
                                  & " AND [plat] >=" & Me![minplat]
```

Cvičení:

1. Nechť ve formuláři jsou obsažena tři pole typu 

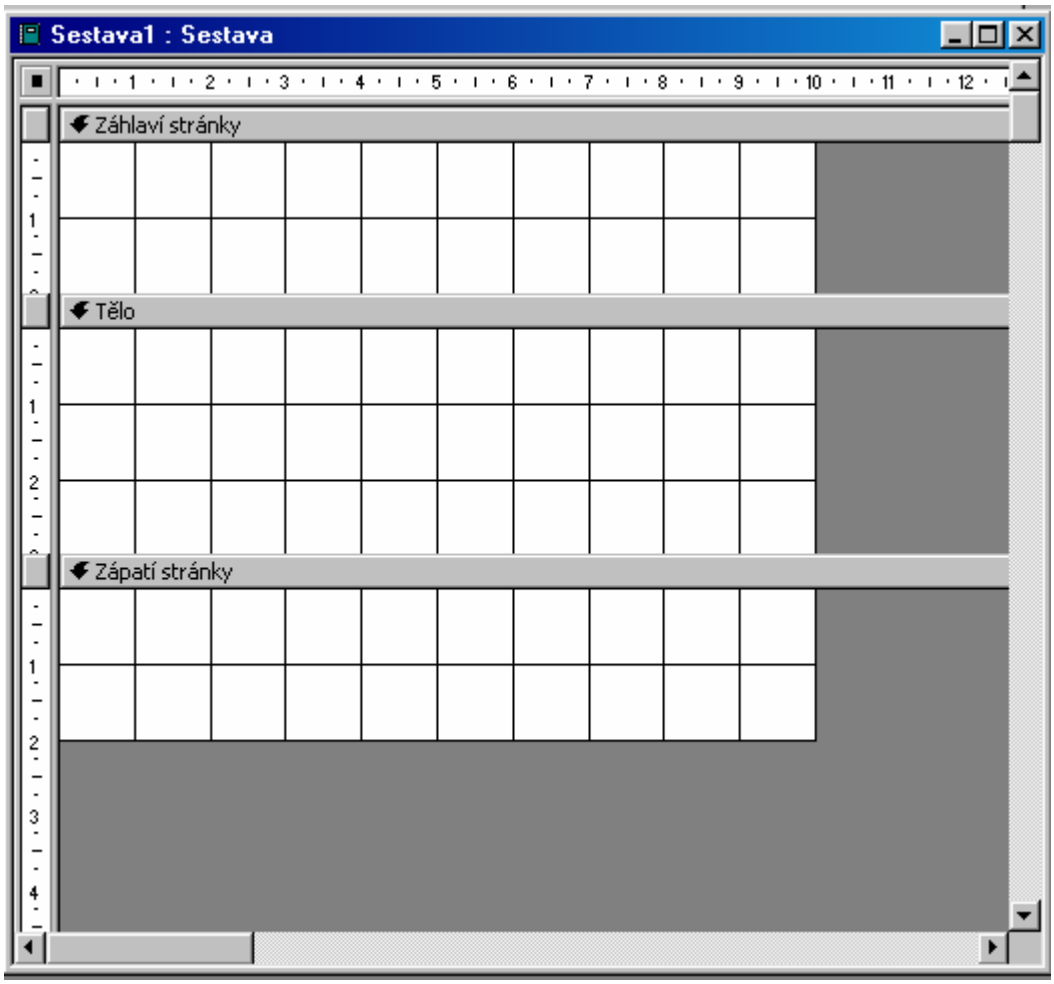
, přičemž první dvě jsou svázána s tabulkovými položkami uchovávajícími informaci o ceně bez DPH a vlastní DPH (v procentech). Definujte výraz pro 3. pole tak, aby vyhodnocoval z prvních dvou polí cenu s DPH, kterou platí zákazník.
2. Vyzkoušejte Průvodce příkazovým tlačítkem pro některé další akce a prostudujte s pomocí helpu význam vygenerovaných příkazů událostních procedur.

2.4. Vytváření sestav

Poznámka:

K tomu, abychom mohli sestavy v MS Accessu vytvářet, musí být v prostředí Windows nainstalována nějaká tiskárna, protože sestavy jsou primárně určeny k tisku přehledů na papír. Jestliže tomu tak není, pak MS Access při pokusu vytvářet sestavu v návrhovém zobrazení nahlásí chybu a akci ukončí. ♦

Abychom mohli sestavu vytvářet, zvolíme v okně Databáze volbu Sestavy a jednu z voleb Vytvořit sestavu v návrhovém zobrazení anebo Vytvořit sestavu pomocí průvodce. Zde je možné s výhodou využívat průvodce, protože nejsložitější na sestavě je dobře rozložit na tiskové straně datové položky a jejich popisné texty. Přesto z pedagogických důvodů doporučujeme alespoň pro začátek zkusit sestavu navrhnout „od nuly“ bez použití průvodce. Jestliže budeme takto postupovat, objeví se prostředí návrhu sestavy s plochou rozčleněnou na tři pásy. Ještě než je budeme nějakým způsobem zaplňovat, přiřadíme sestavě podkladovou tabulku, což zajistíme zcela stejně jako u formulářů.



Obr. S1: Návrh sestavy.

Z pásů Záhlaví stránky, Tělo a Zápatí stránky je nejdůležitější Tělo, do něhož umísťujeme datové položky záznamů. Do Záhlaví stránky zpravidla píšeme nadpisy (výraznějším písmem, často i barevně), může jím být i pojmenování položek ze záhlaví tabulky. Do Zápatí stránky vkládáme nejčastěji číslování stránek, infomaci o tom, kolik má sestava celkem stran a aktuální datum, které je důležité, abychom věděli, kdy sestava byla vytisknuta. Stejně jako u formulářů, je i při práci se sestavou k dispozici souprava nástrojů. Informace datum a číslování stránek zařadíme do Zápatí stránky prostřednictvím nástroje **ab|** a jejich obsah definujeme výrazy:

```
=Date()
="str. " & [Page] & " z " & [Pages]
```

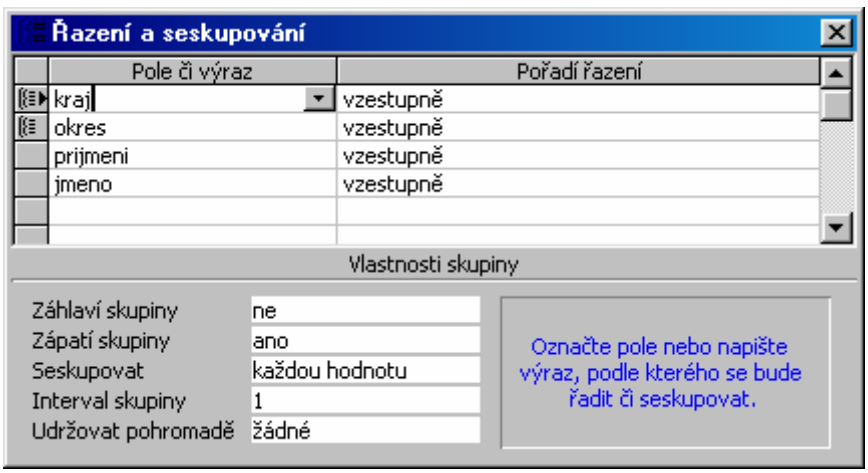
Ve druhém výrazu Page znamená číslo stránky, která se právě tiskne (zobrazuje) a Pages celkový počet stran. Jak již bylo uvedeno u formulářů, operátor & je určen ke zřetězení stringů a hranaté závorky pouze ohraničují identifikátory systémových proměnných (u jednoslovních identifikátorů mohou být vynechány, to znamená i zde).

Kdybychom do záhlaví stránky vložili záhlaví tabulky (pomocí prvků ***Aa***) a do těla položky tabulky (pomocí prvků **ab|**), sestava by vlastně kopírovala zobrazení datového listu.

Silnou stránkou sestav je však možnost vytvářet skupiny záznamů se stejnou hodnotou nějaké položky a pro tyto skupiny vypočítávat součty, průměry, maximum, minimum apod. prostřednictvím tzv. *agregačních funkcí*. Jde o úplně stejnou skupinu agregačních funkcí jako je tomu ve výběrovém dotazu SQL jazyka, a proto pouze odkážeme na tabulku 1 v odstavci 3.1.2.

K vytváření skupin je třeba kliknout na ikonu  Řazení a seskupování v horním panelu nástrojů nebo aktivovat stejnojmennou volbu y z roletového menu Zobrazit. V menu Zobrazit se navíc nachází ještě volba pro Záhlaví či zápatí sestavy, která způsobí, že se v návrhu sestavy objeví ještě jeden pás úplně nahoře a jeden úplně dole. Do záhlaví sestavy můžeme napsat nějaký velký nadpis s případnými dalšími informacemi obdobnými titulnímu listu nějaké publikace a do zápatí sestavy např. informaci o autoru aplikace apod.

Obr. S2 znázorňuje dialog po stisku volby Řazení a seskupování.

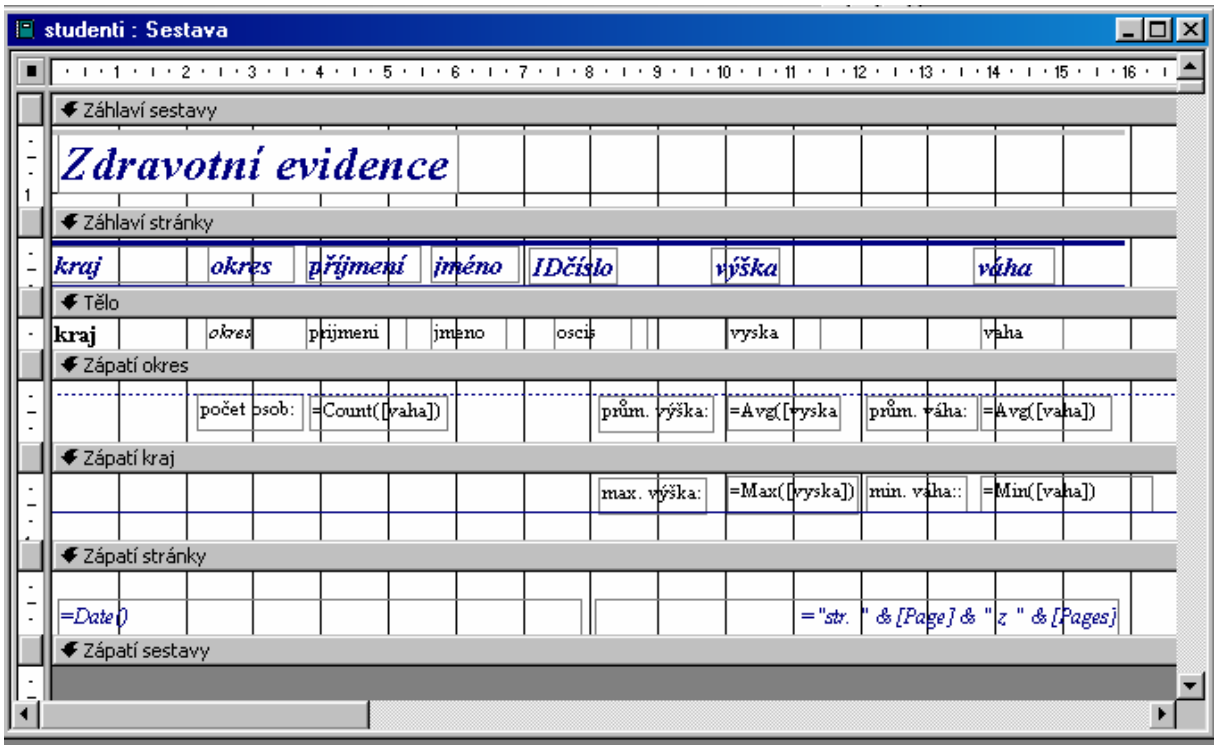


Obr. S2: Definice skupin a pořadí řazení.

Ve schématické ukázce uvažujeme údaje o osobách s informací z jakého regionu (kraje a okresu) pocházejí a jaké mají tělesné proporce (výšku a váhu). Z těchto informací budeme zjišťovat některé agregující údaje, např. průměrnou výšku či váhu v rámci okresu atd., což by

mohlo být užitečné zdravotníkům při zkoumání, jaký vliv má ekologická zátěž krajiny na tělesný růst apod. Pro zjednodušení budeme uvažovat starší označení krajů zkratkami JM (Jihomoravský) atd.

Položky, které jsou určeny k vytváření skupin, se od položek, které pouze zajišťují seřazení podle abecedy (či velikosti), odlišují přítomností ikony Řazení a seskupování před názvem položky, viz obr. S2. Zde se vytváří skupiny podle kraje a podle okresu a v nich se osoby řadí podle jejich příjmení a jména. Řazení se přitom vztahuje i na názvy krajů a okresů, tj. i skupiny jsou řazeny. Aby některá z položek se stala položkou, podle jejíž hodnot se vytváří skupiny, musíme alespoň jednu z vlastností Záhlaví skupiny nebo Zápatí skupiny nastavit do polohy ano. V opačném případě se pro příslušnou položku žádné skupiny nevytváří. Specifikací vlastnosti Seskupovat můžeme zvolit, zda prvky skupiny textového typu musí mít stejné hodnoty (implicitní, Seskupovat = 0 = každou hodnotu) nebo se mohou shodovat jen v *n* prvních znacích (1). U položky typu datum můžeme volit seskupování pro každou hodnotu (0), podle roku (2), kvartálu (3), měsíce (4), týdne (5), dne (6), hodiny (7) či minuty (8) V posledních dvou vlastnostech je možné zvolit, že seskupování se nebude týkat každé hodnoty, ale např. jen každé desáté, resp. na výstupu se nebude vypisovat celá skupina, ale jen několik úvodních vzorků., protože nás zajímají hlavně výsledky agregací a ne přímo dílčí údaje záznamů skupiny.



Obr. S3: Návrh sestavy s agregacemi v zápatích skupin.

Na obr. S3 jsou vidět rozvržení položek, oblasti záhlaví a zápatí skupin, stránky i sestavy, včetně definice agregujících výrazů z funkcí AVG, MAX, MIN.

Konečně na obr. S4 je vidět tvar výstupu tiskové sestavy pro cvičná data.

Zdravotní evidence

kraj	okres	příjmení	jméno	IDčíslo		výška		váha
JČ	CB	Sojka	Přemek	15		183		97
		Sova	Pavel	5		172		66
	počet osob:	2			prům. výška:	177,5	prům. váha:	81,5
JČ	ST	Srovnal	Jaroslav	8		177		71
	počet osob:	1			prům. výška:	177	prům. váha:	71
					max. výška:	183	min. váha::	66
JM	BV	Novotná	Marta	10		168		59
	počet osob:	1			prům. výška:	168	prům. váha:	59
JM	BZ	Karas	Miroslav	7		184		80
		Žídek	Milan	17		189		95
	počet osob:	2			prům. výška:	186,5	prům. váha:	87,5
JM	PV	Janků	Marie	30		166		54
	počet osob:	1			prům. výška:	166	prům. váha:	54
JM	UH	Klos	Petr	22		179		82
		Konečný	Petr	2		174		78
	počet osob:	2			prům. výška:	176,5	prům. váha:	80
					max. výška:	189	min. váha::	54
SČ	AB	Marek	Jiří	6		179		75
		Mošna	Jindřich	12		187		82

Obr. S4: Ukázka výstupu sestavy pro cvičná data.

3. Dotazovací jazyk SQL

SQL (Structured Query Language)

3.1. Výběrový dotaz

SELECT [**ALL** | **DISTINCT** | **DISTINCTROW** | [**TOP** n [**PERCENT**]]]
{* | *tabulka*.* | [*tabulka*.] *položka*1 [**AS** *alias*1]
[,*tabulka*.] *položka*2 [**AS** *alias*2]
[,...] }
FROM {[1]*tabulka*1 [**AS** *alias*1*t*] |
[2]*výběrový-dotaz*1 [**AS** *alias*1*d*] |
[3]=1|2} [**LEFT** | **RIGHT** | **INNER**] **JOIN**
*tabulka*2 [**AS** *alias*2*t*] **ON** *spojovací-podmínka*
[**IN** *externí-databáze*] }
[,{...}]
[**WHERE** *vyhledávací-podmínka*}]
[**GROUP BY** *klíč-agregace*]
[**HAVING** *vyhledávací-podmínka-skupin*]
[**ORDER BY** {*název-sloupce* | *číslo-sloupce* [**ASC** | **DESC**] }
[,{...}]
[**WITH OWNERACCESS OPTION**]

spojovací-podmínka

- *tabulka*1.*položka*1{= |<> | < | <= | > | >= } *tabulka*2.*položka*2

vyhledávací-podmínka

- *výraz*1 {= |<> | < | <= | > | >= } *výraz*2
- *výraz* [**NOT**] **BETWEEN** *dolní* **AND** *horní*
- *výraz* [**NOT**] **IN** *množina-hodnot*
- *výraz* [**NOT**] **LIKE** *vzorový-řetězec*
- *výraz* {= |<> | < | <= | > | >= } **ALL** (*poddotaz*)
- *výraz* {= |<> | < | <= | > | >= } **ANY** | **SOME**
- (*poddotaz*)
- *výraz* [**NOT**] **IN** (*poddotaz*)
- [**NOT**] **EXISTS** (*poddotaz*)

Příklad

ODDĚLENÍ(č-oddělení, název-oddělení)
PRACOVNÍCI(rodné-číslo, jméno, příjmení, č-oddělení)

1. **INNER JOIN** - kombinuje všechny záznamy, které splňují operaci spojení, tj. vyberou se všechna oddělení, na nichž jsou nějací pracovníci
2. **LEFT JOIN** - vybrat všechna oddělení, i když na některých nemusí být žádní pracovníci
3. **RIGHT JOIN** - vybrat všechny pracovníky včetně těch, kteří nejsou přiřazeni na žádná oddělení (např. externisté).

3.1.1 Agregační funkce

název agregační funkce	význam
COUNT(*)	počet záznamů včetně těch, které obsahují **Null** hodnoty
COUNT(*položka*)	počet záznamů, neuvažují se položky s hodnotou **Null**
AVG(*výraz*)	aritmetický průměr
SUM(*výraz*)	součet
MIN(*výraz*)	minimální hodnota
MAX(*výraz*)	maximální hodnota
STDEV(*výraz*)	směrodatná odchylka základního souboru číselných výrazů
STDEVP(*výraz*)	odhad směrodatné odchylky základního souboru ze směrodatných odchylek skupin vytvořených pomocí **GROUP BY**
VAR(*výraz*)	rozptyl základního souboru číselných výrazů
VARP(*výraz*)	odhad rozptylu základního souboru z rozptylů skupin vytvořených pomocí **GROUP BY**

Tab. 1. Agregační funkce

Ve vícevrstvé agregaci záleží na pořadí složek agregačního klíče. Jestliže toto pořadí v předchozím příkladu obrátíme, pak dostaneme odlišný výstup.

3.1.2 SQL s poddotazy

Příklad:

Jména čtenářů, kteří mají rezervovanou *nějakou* knihu.

```
SELECT čtenáři.jméno, čtenáři.příjmení
    FROM čtenáři
    WHERE čtenáři.ID-čtenáře IN
            (SELECT rezervace.ID-čtenáře
                FROM rezervace)
```

Zadání předchozího příkladu lze vyjádřit jinými slovy tak, že chceme určit jména čtenářů takových, že *existuje* kniha, kterou mají rezervovánu. Při tomto vyjádření se nabízí použití klauzule **EXISTS**.

```
SELECT čtenáři.jméno, čtenáři.příjmení
    FROM čtenáři
    WHERE EXISTS
        (SELECT *
```

FROM rezervace
WHERE čtenáři.ID-čtenáře = rezervace.ID-čtenáře)

3.2. Křížový dotaz

TRANSFORM *agregační-funkce* (*výraz*1)
výběrový-dotaz
PIVOT *výraz*2

3.3. Akční (aktualizační) dotazy

Vytvoření nové tabulky.

SELECT [**ALL** | **DISTINCT** | ...]
seznam-polí
INTO *nová-tabulka* [**IN** *externí-databáze*]
FROM *zdrojová-tabulka*

Přidání jednoho záznamu do existující tabulky nebo dotazu.

INSERT INTO { *tabulka* | *dotaz* }
[(*seznam polí, do nichž se data přidávají*)]
VALUES (*seznam hodnot*)

Výběr záznamů z jedné tabulky a jejich přidání do nových záznamů jiné tabulky.

INSERT INTO { *tabulka* | *dotaz* }
[(*seznam polí, do nichž se data přidávají*)]
SELECT ...
FROM ...
WHERE ...

Změna obsahu položek v tabulce.
UPDATE { *tabulka* | *dotaz* }
[**IN** *externí-databáze*]
SET *název-sloupce* = { *výraz* | **NULL** }
[, ...]
WHERE *vyhledávací-podmínka*

Vymazání (zrušení) záznamů v tabulce.

DELETE [∗ | *tabulka*. ∗ | *seznam-sloupců*]
FROM ...
[**IN** *externí-databáze*]
WHERE *vyhledávací-podmínka*

3.4. Definiční dotazy

Vytvoření nové tabulky, definice položek (název, datový typ, velikost).a indexů

CREATE TABLE *tabulka*
(*položka*1 *datový-typ*[(*velikost*)]
[**NOT NULL**] [**CONSTRAINT** *index*1…]
[, *položka*2 *datový-typ*[(*velikost*)]
[**NOT NULL**] [**CONSTRAINT** *index*2…]
[, ...]]
[,**CONSTRAINT** *vícesložkový-index* …
[, ...]])

Klauzule CONSTRAINT
- *definice indexů* (*a vytvoření relace s jinou tabulkou*)
v příkazech **CREATE TABLE a ALTER TABLE**

- *Jednoduchý index*

CONSTRAINT *jméno-indexu*
{ **PRIMARY KEY** | **UNIQUE** | **NOT NULL** |
REFERENCES *cizí-tabulka* [(*cizí-pole*1, *cizí-pole*2)]}

- *Vícesložkový index* (lze jej definovat pouze pro primární klíč.)

CONSTRAINT *jméno-indexu*
{ **PRIMARY KEY** (*primární-segm*1[,*primární-segm*2 [, ...]]) |
UNIQUE} (*jedinečný-segm*1 *jedinečný-segm*2 [, ...]]) |
NOT NULL (*nenulový-segm*1[,*nenulový-segm*2 [, ...]]) |
FOREIGN KEY (*ref*1[,*ref*2 [, ...]]) **REFERENCES** *cizí-tabulka*
[(*cizí-pole*1 [,*cizí-pole*2
[, ...]])]}

- *Modifikace struktury existující tabulky (přidávání nových položek, resp. odstraňování existujících položek.*

ALTER TABLE *tabulka*
{ **ADD** { **COLUMN** *položka datový-typ* [(*velikost*)]
[**NOT NULL**}] [**CONSTRAINT** *index* …] |
CONSTRAINT *vícesložkový-index* …} |
DROP { **COLUMN** *položka* | **CONSTRAINT** *jméno-vícesložkového indexu* } }

- *Zrušení existující tabulky z databáze, resp. zrušení existujícího indexu z tabulky.*

DROP {**TABLE** *tabulka* | **INDEX** *index* **ON** *tabulka*}

- *Vytvoření nového indexu pro existující položku v tabulce.*

CREATE [**UNIQUE**] **INDEX** *index*
ON *tabulka* (*položka* [**ASC** | **DESC**]
[,*položka* [**ASC** | **DESC**], ...])
[**WITH** { **PRIMARY** | **DISALLOW NULL** | **IGNORE NULL**}]

4. Visual Basic pro aplikace MS Access

Definice konstant

[**Public** | **Private**] **Const** *název-konstanty* [**As** *datový_typ*] = *konstantní_výraz*

Private (implicitní) – konstanta přístupná pouze v modulu, kde byla definována
Public v deklarační sekci (modulu, formuláře nebo sestavy) – globální platnost
datový_typ: **Byte**, **Boolean**, **Integer**, **Long**, **Currency**, **Single**, **Date**, **String**, **Variant**

4.1. Řídící struktury Visual Basicu

4.1.1 Přiřazovací příkazy

proměnná = výraz
Set *proměnná* ={ **New** *objektový-výraz* | **Nothing**}

4.1.2 Příkazy větvení

If *podmínka* **Then** *příkaz*1 [**Else** *příkaz*2]

```
If podmínka Then
   příkazy1
[Else If podmínka2 Then
          příkazy2
       [Else
          příkazy3]]
End If
```

```
Select  Case testovaný-výraz
        Case seznam-hodnot-1
              příkazy1
        Case seznam-hodnot-2
              příkazy2
        ...
        [Case Else
              příkazy-n]
End Select
```

4.1.3 Příkazy cyklu

For *řp = poč* **To** *konc* [**Step** *krok*]
    *příkazy*1
[**Exit For**]
    *příkazy*2
Next *řp*

Jestliže se tělo cyklu provádí pro všechny prvky nějaké kolekce, je výhodné použít následující verzi cyklu **For**.

For Each *prvek* **In** *skupina*
    *příkazy*1
[**Exit For**]
    *příkazy*2
Next *prvek*

Do {**While** | **Until**} *podmínka*
    *příkazy*1
    [**Exit Do**]
    *příkazy*2
Loop

Do
    *příkazy*1
    [**Exit Do**]
    *příkazy*2
Loop {**While** | **Until**} *podmínka*

While *podmínka*
    příkazy
Wend

4.1.4 Příkazy skoku

- *Nepodmíněný skok* na jiný příkaz v proceduře nebo funkci
  GoTo {*návěští* | *číslo-řádku*}

- *Podmíněný skok při vyhodnocení chyby*
  On Error {**GoTo** *identifikátor-řádku* | **Resume** [**Next**] | **GoTo** 0}

- Je-li uvedeno pouze **Resume**, provede se skok na příkaz, který způsobil chybu a Access se pokusí jej znovu provést.

- Konstrukce **Resume Next** zachycuje chyby, přitom se však pokračuje v provádění následujícího příkazu.
- Jestliže je použito **GoTo** 0, pak je zachycování chyb v aktuální proceduře vypnuto a chyba je předána chybové rutině ve volající proceduře. Pokud žádná předcházející chybová rutina neexistuje, otevře se chybové dialogové okno.

Poznámka:
V chybové rutině (podprogramu ošetření chyby) lze testovat:

1. Hodnotu vestavěné proměnné **Err**(číslo chyby). Jestliže **Err**=0, pak žádná chyba nenastala.
2. Chybové hlášení pomocí funkce **Error**.
3. Lze využít i objekt **Err** a jeho vlastnosti **Err**.**Description**, v níž je uložena textová informace o vzniklé chybě, a **Err**.**Number** obsahující číslo chyby.

4.1.5 Příkaz With

With *objekt*
    příkazy
End With

Příklad:

Odkazy na prvky kolekce ve Visual Basicu

Forms … kolekce otevřených formulářů
Forms(*i*) … (*i*+1)-ní otevřený formulář
Forms[*objednávky*] … formulář `objednávky`
Forms("objednávky") … formulář `objednávky`

Jestliže je jméno prvku kolekce uloženo v proměnné, např. je parametrem procedury nebo funkce, pak musíme použít poslední zápis s kulatými závorkami, uvozovky se ovšem nepoužijí.

4.1.6 Procedury a funkce

[**Public** | **Private**] [**Static**] **Sub** *název-procedury* ([*seznam-parametrů*])
    [*příkazy*1]
    [**Exit Sub**]
    [*příkazy*2]
End Sub

	Rozsah platnosti procedury/funkce
Public	Ve všech procedurách/funkcích *všech* modulů.
Private	V dalších procedurách/funkcích *stejného* modulu.
Static	Všechny proměnné deklarované (implicitně či explicitně) po celou dobu otevření modulu obsahující tuto proceduru budou uchovány. **Příklad:** *Čítač* uvnitř procedury, jehož hodnota se zvyšuje o 1 při každém volání procedury.

Tab. 2. Rozsah platnosti procedury/funkce

4.1.6.1 Parametry procedur a funkcí

[**Optional**] [**ByVal** | **ByRef**] [**ParamArray**] *název-parametru* [**As** *datový-typ*]

- **Optional** označuje nepovinný parametr. Umožňuje deklarovat parametr typu **Variant**. Za nepovinným parametrem musí být všechny další parametry rovněž nepovinné. Test nepřítomnosti nepovinných parametrů lze provést pomocí funkce **IsMissing**().
- **ByVal** - volání hodnotou. Jestliže je skutečným parametrem výraz, Visual Basic s ním zachází jako by byl deklarován pomocí **ByVal**.
- **ByRef** - volání odkazem. Pole se vždy předávají odkazem.
- **ParamArray** musí být v seznamu parametrů poslední.

4.1.6.2 Volání procedur a funkcí

Proceduru můžeme volat dvěma způsoby:

Call *název-procedury*(*seznam-skutečných-parametrů*)

nebo

název-procedury seznam-skutečných-parametrů

Výsledkem funkce je hodnota, a tedy je možné ji použít všude tam, kde je přípustný výraz, např. na pravé straně přiřazovacího příkazu:

proměnná = *název-funkce*(*seznam-skutečných-parametrů*)

výchozí hodnoty:(Pokorný/Kopp, str. 40-41)

číslo	0
řetězec	prázdný řetězec
typ **Variant**	**Empty**
objektový typ	**Nothing**

4.2. Objekt RecordSet

Příklad:
Cyklus modifikace ve všech řádcích tabulky (např. zvýšení poplatku za elektřinu o 10 procent).

```
    ...
    Dim ws As Workspace
    Dim db As Database
    Dim rst As RecordSet
'1.
    Set ws = DBEngine.Workspaces(0)
    Set db = ws.Databases(0)

'2  Set db = DBEngine(0)(0)

'3. Set ws = DBEngine.Workspaces(0)
    'Set db = ws.OpenDatabase("název souboru .mdb")

'4. Set db  = CurrentDb()

    Set rst = db.OpenRecordset("název tabulky")
    rst.MoveFirst
    While Not(rst.EOF)                  '   Do Until rst.EOF
        rst.Edit                        '       rst.Edit
        rst![sazba] = rst![sazba] *1.1  '       rst![sazba] = rst![sazba] *1.1
        rst.Update                      '       rst.Update
        rst.MoveNext                    '       rst.MoveNext
    Wend                                '   Loop
    rst.Close
    ...
```

4.3. Volání SQL dotazu z VBA

1. Uložený dotaz (výběrový, křížový, …)
DoCmd.**OpenQuery** *název-dotazu* [, *pohled*][, *datový-mód*]

pohled: `acViewDesign`, `acViewNormal`, `acViewPreview`
datový-mód: `acAdd`, `acEdit`, `acReadOnly`

2. Výběrový dotaz
Set *rst* = *db*.**OpenRecordset**(*výběrový_dotaz*,`dbOpenDynaset`)

3. Akční a definiční dotazy

DoCmd.**RunSQL** *dotaz* [, *použít-transakci*]

nebo
db.Execute *dotaz*

V příkazu **DoCmd**.**RunSQL** lze jako *dotaz* použít pouze *akční* nebo *definiční* dotaz, to znamená dotazy **INSERT INTO**, **DELETE**, **UPDATE**, **SELECT INTO**, resp. dotazy **CREATE TABLE**, **DROP TABLE**, **ALTER TABLE**,**CREATE INDEX**, **DROP INDEX**.

5. Kontrolní otázky

I. SQL

1. Je dána tabulka

   PACIENTI(rodné-č, jméno, příjmení, diagnóza, den-nástupu, č-oddělení, název-oddělení)

   Dotazem SQL zjistit všechna oddělení a všechny diagnózy nemocí, s nimiž pacienti na odděleních v nemocnici leží a kolik pacientů tu kterou nemoc má.

   Příklad výstupu:

oddělení	diagnóza	počet
interna	slepé střevo	3
	zápal plic	4
onkologie	leukémie	7
	rakovina plic	5

2. Je dána tabulka

   BYDLIŠTĚ(rodné-č, jméno, příjmení, místo, ulice, ČP, PSČ, městská-čtvrt, okres, kraj)

   Dotazem SQL zjistit všechny kraje a okresy, v nichž evidované osoby bydlí, s uvedením jejich počtu.

   Příklad výstupu:

kraj	okres	počet
jihomoravský	Uh. Hradiště	3
	Hodonín	4
severomoravský	Opava	7
	Frýdek Místek	5

3. Je dána tabulka

   TRESTNÍ-REJSTŘÍK(rodné-č, jméno, příjmení, trestný-čin, paragraf, sazba-od, sazba-do, výše-trestu, nápravná-skupina)

   Dotazem SQL zjistit všechny potrestané s uvedením celkového počtu trestů a součtu let za udělené tresty a výstupní informaci přitom setřídit sestupně podle počtu trestů.

4. Je dána tabulka

   AUTA(číslo, typ, barva, obsah-válců, rok-výroby, prodejní-cena, dovezené{typ logical}, rodné-č-majitele, jméno, příjmení)

Dotazem SQL zjistit všechny dovezené typy aut s rokem výroby 1996, jejichž průměrná pořizovací cena překročila 500 tisíc a ve výstupu uvést typ, počet a průměrnou pořizovací cenu.

5. Je dána tabulka

   VÝPŮJČKA(rodné-č, jméno, příjmení, název-knihy, nakladatelství, autor, ISBN, tématické-zaměření)

   Dotazem SQL zjistit všechny čtenáře, kteří mají vypůjčené nejméně dvě knihy.

II. Visual Basic pro aplikace Accessu

1. Napsat funkci ve Visual Basicu, jejíž návratovou hodnotou je řetězec obsahující názvy všech tabulek definovaných v aktuální databázi. Tabulky tvoří kolekci TableDefs, jež je součástí objektu Database. Vlastnost název se ve VBA označuje Name.
2. Napsat proceduru ve Visual Basicu, která ve všech řádcích tabulky ZBOŽÍ(ID, název, cena) provede sezónní snížení cen o 30%. Úlohu vyřešte bez příkazu SQL s využitím objektu typu recordset.
3. Napsat proceduru ve Visual Basicu, která v tabulce UCHAZEČI(rodné-č, jméno, příjmení, vzdělání) zruší všechny osoby, které nemají VŠ vzdělání. Úlohu vyřešte bez použití SQL s pomocí objektu typu recordset.
4. Napsat funkci ve Visual Basicu, jejíž návratovou hodnotou je minimální cena zboží v tabulce ZBOŽÍ(ID, název, cena). Úlohu vyřešte bez použití SQL s pomocí objektu typu recordset.
5. Napsat funkci ve Visual Basicu, jejíž návratovou hodnotou je průměrná cena zboží v tabulce ZBOŽÍ(ID, název, cena). Úlohu vyřešte bez použití SQL s pomocí objektu typu recordset.

Literatura

[1] Farana, R.: *Tvorba relačních databázových systémů*. VŠB TU, Ostrava, 1999.

[2] Farana, R.: *Aplikace počítačů v řízení. Relační databáze.* VŠB TU, Ostrava, 1995.

[3] Fortier, P.J.: *Database Systems Handbook*. McGraw-Hill, 1997, ISBN 0-07-021626-6.

[4] Havlát, T., Benešovský, M.: *Úvod do databázových systémů*. Skriptum UJEP PřF, Brno, 1984.

[5] Pokorný, J.: *Visual Basic pro aplikace Accessu 2000*. Kopp, České Budějovice, 2000.

[6] Pokorný J.: *Učíme se SQL*. PLUS, Praha, 1993.

[7] Pokorný, J., Halaška, I.: *Databázové systémy*. *Vybrané kapitoly a cvičení*. Karolinum – nakladatelství Univerzity Karlovy, Praha, 1998.

[8] Šimůnek, M.: *SQL - kompletní kapesní průvodce*. Grada, Praha, 1999.

[9] Viescas, J.: *Mistrovství v Microsoft Access 2000*. Computer Press, Praha, 2000.