ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

PROVOZNĚ EKONOMICKÁ FAKULTA

BAKALÁŘSKÁ PRÁCE

MS ACCESS V INFORMAČNÍM ZABEZPEČENÍ AUTOMOBILOVÝCH PARKŮ

VYPRACOVAL: ONDŘEJ WOLLNER

VEDOUCÍ BAKALÁŘSKÉ PRÁCE: ING. VÁCLAV VOSTROVSKÝ, PH.D.

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "MS ACCESS v informačním zabezpečení automobilových parků" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30.4.2009 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Václavu Vostrovskému, Ph.D. za ochotu a spolupráci při tvorbě mé bakalářské práce.

MS ACCESS v informačním zabezpečení automobilových parků

Automobile park management in MS ACCESS application

Souhrn:

Cílem práce je návrh informačního zabezpečení automobilových parků společnosti pomocí nástroje Microsoft ACCESS. Teoretická část práce pojednává o základních principech relačně databázových prostředků a popisuje nástroj pro tvorbu databázových aplikací Microsoft ACCESS. V druhé části práce specifikuje požadavky na informační systém evidence provozu vozových parků. Závěrečná část je praktickou ukázkou realizace databázové aplikace v prostředí MS ACCESS.

Klíčová slova: relační databáze, datový model, SQL, vozový park, MS ACCESS

Summary:

The aim of this thesis is proposal of company automobile park information system developed by Microsoft ACCESS tool. Theoretical part of the thesis discusses basic relational databases principles and describes database application development tool MS ACCESS. The second part specifies the information system requirements for automobile park management. The final part is presentation of practical MS ACCESS database application realization.

Key words: relational databáze, data model, SQL, automobile park, MS ACCESS

Obsah

1 Úvod

Pojem databáze je v dnešní době, kdy se s počítačovými systémy setkáváme denně, poměrně známý a téměř každý si pod tímto slovem představí soubor užitečných informací různého typu, jako například seznam kontaktů klientů společnosti, receptů či položek zboží v nabídce. Různé formy databázových aplikací tedy běžně používáme, aniž bychom si uvědomovali, jak jsou ve skutečnosti záznamy v databázi uspořádány.

V informačních systémech (IS) společností je většinou využíváno různých databázových aplikací. Hlavní IS slouží k uchovávání a práci se záznamy popisující obchodní nebo výrobní činnost firmy. Výchozí činnost společnosti je však spojena s dalšími aktivitami a procesy, které je také nezbytně nutné monitorovat a vést o nich evidenci. Mezi tyto aktivity může patřit marketingová činnost, evidence zaměstnanců nebo správa vozového parku. Ideálním případem z pohledu uživatelů i administrace dat je zahrnutí veškeré evidence do jednoho informačního systému. V praxi to však z různých důvodů nebývá vždy možné. Hlavní IS společnosti může být úzce specializovaný nebo provádění jeho změn je jak časově, tak i ekonomicky náročné, nezbývá tedy jiné řešení, než vytvořit pomocí jednoduchého, ale dostatečně flexibilního nástroje, databázovou aplikaci vlastními silami.

Vhodnou volbou může být program ACCESS společnosti Microsoft, který si autor této práce vybral, neboť je to jeden z nástrojů, který používá ve své praxi při tvorbě databázových aplikací.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem této práce je objasnění teoretických principů relačně databázových prostředků, vymezení zásad jejich uplatnění v problematice datových evidencí malých a středních podnikatelských subjektů.

V rámci tohoto bude dále představen nástroj pro tvorbu databázových aplikací MS ACCESS, formulace specifikací a požadavků kladených na informační zabezpečení provozu automobilových parků a naznačení tvorby vlastního návrhu databázové aplikace. V závěru práce budou hodnoceny klady a zápory zvoleného řešení.

2.2 Metodika

Pro práci byla zvolena metoda hromadného sběru dat z dostupných zdrojů tištěné literatury i internetových portálů. Popsané závěry byly učiněny na základě objektivní kritiky analýzou získaných informací.

Při samotném návrhu informačního zabezpečení byla zpracována analýza požadavků na informační systém formou interview s uživateli, na jehož základě byla vytvořena testovací verze programu. Po následném zapracování všech připomínek uživatelů byla dokončena finální verze databázové aplikace.

3 Teoretické principy relačně databázových prostředků

3.1 Definice databáze

V nejjednodušším slova smyslu je databáze množina záznamů a souborů, které jsou organizovány za určitým účelem. Neboli databáze je strukturovaná kolekce záznamů nebo dat uložených v počítačovém systému. Struktury je dosaženo organizováním dat dle databázového modelu (Viescas, 2005; www.wikipedia.org).

Při návrhu databázové aplikace je nezbytné vycházet nejen ze znalosti reálných vlastností sledovaných objektů a vazeb mezi nimi, ale také z teoretických principů relačně databázových prostředků. V řádně navržené databázi musejí být data uložena tak, aby se dala snadno udržovat a zároveň se neplýtvalo prostředky, například duplicitním uložením informací (Pokorný a Halaška, 2003).

3.2 Relačně datový model

Databázové řídicí systémy uspořádávají a strukturují data tak, aby k nim uživatelé a aplikační programy mohli získat přístup a dále s nimi pracovat. Strukturu dat a techniky zpřístupňování dat, které poskytují specifické databázové řídicí systémy, označujeme jako jejich *datový model*. Datový model vymezuje „charakter“ jak databázových řídicích systémů, tak i aplikací, pro něž je zvláště vhodný. Nejběžnějším typem modelu užívaným v dnešních databázových systémech je relační model (Groff, 2005).

Požadavky na relační datový model lze charakterizovat následujícím způsobem (Vostrovský, 2004):

1. Hodnoty v tabulkách musejí být *atomické* – nesmějí se tedy skládat z dalších hodnot.
2. Hodnoty musejí být *skalární* – nesmějí mít tedy více než jeden rozměr.

3. Hodnoty v tabulkách existují jako prvky jednotlivých domén. Všechny prvky dané domény musejí být mezi sebou porovnatelné a musejí náležet jednomu datovému typu.
4. Pro práci s tabulkami se používá operací výrokové logiky. V každé tabulce slouží hodnoty v jednom nebo více sloupcích (doménách) k jednoznačné identifikaci řádků mezi sebou. Tyto hodnoty jsou označovány jako primární klíče tabulky.
5. V některých tabulkách hodnoty v jednom nebo více sloupcích (doménách) mají vztah k hodnotám v jiných tabulkách (nebo ve zvláštním případě i k hodnotám vlastní tabulky). Tyto hodnoty jsou označovány jako cizí klíče.
6. V tabulkách lze definovat podmnožiny řádků anebo podmnožinu sloupců. Operace vybírající podmnožinu řádek jako výběr řádek z tabulky je označována jako selekce tabulky. Operace vybírají podmnožinu sloupců, jako výběr sloupců z tabulky je označována jako projekce tabulky.
7. Více tabulek lze kombinovat mezi sebou jako běžné množiny pomocí operací sjednocení, rozdílu, průniku množin a kartézského součinu množin. Kombinace kartézského součinu a selekce se nazývá spojení tabulek.

Relačně datový model byl v roce 1985 navrhnut Dr. Coddem jako pokus o zredukování a zjednodušení různých datových modelů vytvořených v sedmdesátých a osmdesátých letech. Dr. Codd namísto dosavadně používané struktury rodič-potomek navrhl model databáze, kde jsou všechna data reprezentována prostými tabulkami datových hodnot (Groff, 2005).

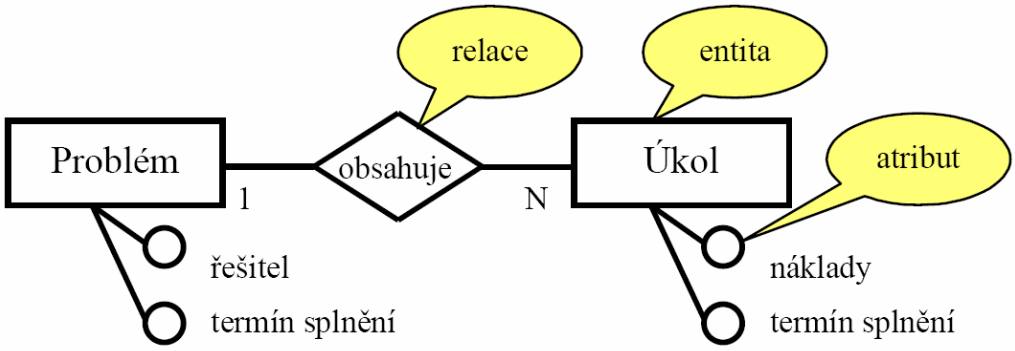
3.3 Struktura relační databáze

Všechny tabulky, které databáze obsahuje, jsou jednoznačně identifikovatelné svým názvem, určeným tvůrcem databáze při jejich vytvoření. Jednotlivé záznamy obsažené v tabulkách jsou uloženy v řádcích, rozdělených do sloupců, které jsou uspořádány v daném pořadí zleva doprava. V každém sloupci jsou uloženy různé *atributy subjektu*. Atributy uložené ve sloupcích musejí být stejného datového typu. *Doménou sloupce* nazýváme množinu dat, kterou může sloupec obsahovat. Každý sloupec musí mít jedinečný název v rámci jedné tabulky, zatímco sloupce v několika různých tabulkách jedné databáze mohou mít názvy shodné. Záznamy uložené v jednotlivých řádcích nejsou na rozdíl od sloupců nijak uspořádané, ale měli by být jednoznačně identifikovatelné. K identifikaci slouží tzv. *primární a cizí klíče*, kterými jsou na základě relací vzájemně propojeny tabulky. *Primární klíče* mohou být kombinací více sloupců tabulky nebo to mohou být jedinečné záznamy v rámci jednoho sloupce, kdy každá tabulka má pouze jeden primární klíč. *Cizí klíče* jsou hodnotou klíče primárního pro nějakou jinou tabulku, jedna tabulka tedy může obsahovat libovolné množství cizích klíčů, pomocí kterých může být spojena s ostatními tabulkami (Kruczek, 2007).

Termín *relační* pramení ze skutečnosti, že každý záznam v databázi obsahuje informace vztahující se k jedinému subjektu – a pouze k tomuto subjektu. Také data o dvou třídách informace (například zákazníci a objednávky) mohou být zpracovávána jako jediná entita založená na vztažených datových hodnotách. Například by bylo nadbytečné ukládat informaci o jménu zákazníka, která může být použita pro spojení každé objednávky s informacemi o zákazníkovi (Viescas, 2005).

3.4 E-R model

S relačně databázovým modelem jsou také spojeny pojmy *entita* a *typ entity*. *Entita* je popsatelný a jednoznačně určitelný objekt s vlastnostmi vyjádřenými parametry (atributy). *Typ entity* je skupina entit se stejnými parametry. Jde o nástroje modelování, používané při vytváření tzv. konceptuálních modelů databáze. Nejznámějším nástrojem je *E-R model* (z angl. Entity Relationship). V tomto systému modelujeme databázi pomoci entit a relací (vztahů, vazeb) mezi entitami (viz. Obrázek č. 1). Jak je zřejmé, pojem relace je zde použit ve zcela jiném významu, než v relačním datovém modelu. I přes vnesení řady problémů, je v databázových programech pojem relace běžně používán, a to ve významu vztahu mezi n-ticemi (řádky tabulky, záznamy) (Farana, 1999; Pokorný a Halaška, 2003).



Obrázek č. 1 E-R model

3.5 Coddova pravidla

Tvůrce relačně datového modelu Dr. Codd sestavil 12 pravidel, kterými se musí databáze řídit, aby mohla být považována za skutečně relační (Groff, 2005):

1. *Pravidlo informace.* Všechny informace v relační databázi se reprezentují explicitně na logické úrovni a jedním způsobem – hodnotami v tabulkách.

2. *Pravidlo zaručeného přístupu.* Musí být zajištěno, aby úplně každý údaj v relační databázi byl logicky přístupný použitím názvu tabulky, hodnoty primárního klíče a názvu sloupce.
3. *Systematické ošetření prázdných hodnot.* Prázdné hodnoty (odlišné od prázdného znakového řetězce nebo řetězce znaků mezer a odlišné od nulových nebo jakýchkoliv jiných čísel) jsou systematicky plně podporovány relačně databázovým řídícím systémem pro prezentaci chybějících informací a neplatných informací nezávisle na datovém typu.
4. *Dynamický online katalog založený na relačním modelu.* Popis databáze se reprezentuje na logické úrovni stejným způsobem jako běžná data tak, aby se oprávnění uživatelé mohli dotazovat na tato data pomocí stejného relačního jazyku, jehož pomocí se dotazují na normální data.
5. *Pravidlo komplexního datového podjazyku.* Relační systémy mohou podporovat několik jazyků a různé režimy použití terminálu (například režim doplňování). Nicméně musí existovat přinejmenším jeden jazyk, jehož příkazy jsou vyjádřitelné, na nějakou dobře definovanou syntaxi jako řetězce znaků a tento jazyk komplexně podporuje všechny následující položky:
    - *definici dat*
    - *definici pohledu*
    - *manipulaci s daty (interaktivně programově)*
    - *omezení integrity*
    - *autorizaci*
    - *vymezení transakce (begin, commit a rollback)*
6. *Pravidlo aktualizace pohledu.* Všechny pohledy, které je teoreticky možné aktualizovat, je rovněž možné aktualizovat systémově.
7. *Vysokoúrovňové vkládání, aktualizace a odstranění.* Schopnost zpracovávat základní relace nebo odvozené relace jako jediný operand se aplikuje nejenom na vyhledávání dat, ale rovněž na vložení, aktualizaci a odstranění dat.

8. *Fyzická datová nezávislost.* Aplikační programy a terminálové aktivity zůstávají logicky nedotčené, kdykoliv jsou provedeny nějaké změny buďto v reprezentacích úložiště nebo přístupových metodách.
9. *Logická datová nezávislost.* Aplikační programy a terminálové aktivity zůstávají logicky neporušené, pokud jsou v základních bulkách provedeny změny k uchovávání informací jakéhokoliv druhu.
10. *Nezávislost integrity.* Omezení integrity specifické pro jednotlivé relační databáze musí být možné definovat v relačním datovém podjazyku a uložit v katalogu nikoliv v aplikačních programech.
11. *Distribuční nezávislost.* Relační databázový řídicí systém má distribuční nezávislost.
12. *Pravidlo nenarušení.* Má-li relační systém nízkoúrovňový jazyk (zpracovávající záznamy jednotlivě), nelze pomocí tohoto jazyka rušit nebo obcházet pravidla integrity a omezení vyjádřená ve vysokoúrovňovém relačním jazyku.

3.6 Relace

Relace (vztahy) jsou základním nástrojem v každém moderním databázovém prostředí. Relace definují spojení mezi primárním klíčem jedné tabulky a cizím klíčem tabulky druhé. Jinak řečeno, přiřazuje záznamy tabulky záznamům druhé tabulky se shodným klíčem (Kruczek, 2007).

Relace mohou být tří typů:

- *Relace 1:1*
- *Relace 1:N*
- *Relace M:N*

3.6.1 Relace 1:1

Tento typ relace je používán asi nejméně často, protože nám nepřináší příliš výhod. Znamená to, že ke každému záznamu v první tabulce odpovídá maximálně jeden záznam se stejným klíčem v tabulce druhé.

Použití této relace najdeme zejména ve speciálních případech, kdy například:

- Potřebujeme zabezpečit některé části záznamů, proto je rozdělíme do více tabulek.
- Nestačí nám maximální počet sloupců tabulky nebo by byl pro nás počet sloupců příliš velký a nepřehledný.

3.6.2 Relace 1:N

Relace typu *1:N* je asi nejběžnější používanou relací. Každému záznamu v tabulce na straně 1 může odpovídat více záznamů se shodným klíčem na straně N. Jako příklad použijeme databázi knihovny s tabulkami Vydavatele a Knihy. Každý vydavatel může publikovat více knih, každá kniha má pouze jednoho vydavatele. Jedná se tedy o relaci 1:N.

3.6.3 Relaci M:N

Relace *M:N* znamená, že každému záznamu v tabulce na straně M může odpovídat více záznamů se shodným klíčem na straně N a naopak. Jako příklad použijeme databázi knihovny s tabulkami Autoři a Knihy. Každý autor může napsat více knih, každá kniha může mít také více autorů. Jedná se tedy o relaci M:N.

Relaci M:N nelze do databáze Microsoft ACCESS zapsat přímo, ale je potřeba ji převést na relace typu 1:N. Postup převodu (Viescas, 2005):

1. Naplánujeme si relaci M:N, zvolíme primární a cizí klíče. Nazveme si klíč na straně M jako KlicM a klíč na straně N jako KlicN.

2. Vytvoříme novou tabulku, která bude obsahovat dva sloupce – KlicM a KlicN. Jako primární klíč v této tabulce by mohl být vícepoložkový klíč složený z obou sloupců.
3. vytvoříme relaci z tabulky na straně N do nově vytvořené tabulky. Tato relace je již typu 1:N.
4. Vytvoříme relaci z tabulky na straně M do nově vytvořené tabulky. Tato relace je již typu 1:N.

3.7 Datová normalizace

Normalizace je postupný reverzibilní proces nahrazování dané množiny relací souhrnem relací, které mají jednodušší a současně i „regulérnější“ strukturu. Tento proces zjednodušování je založen na nestatickém kritériu. Reverzibilita zaručuje, že původní souhrn informací lze obnovit, a proto při použití této techniky nedochází ke ztrátě informace. Cíle normalizace jsou následující (Vostrovský, 2004):

1. Umožnit reprezentaci každé relace v databázi.
2. Získat účinné algoritmy vyhledávání, založené na jednodušší množině relačních operací.
3. Odstranit v relacích nežádoucí závislosti při operacích vkládání, aktualizace a rušení.
4. Redukovat potřebu restrukturalizace při zavedení nového typu dat.
5. Zajistit, aby souhrn relací byl neutrální vzhledem k četnosti dotazů, které mají tendenci měnit se v čase.

3.7.1 Normálové formy

Databázová komunita vyvinula několik pravidel pro normalizaci databází. Tyto pravidla jsou obecně známa jako normálové formy a jsou číslována od 1. normálové formy (nejnižší forma normalizace) až po 5. normálovou formu.

V praxi se však většinou setkáváme pouze s prvními třemi formami normalizace (Hillyer, 2005):

1. *normální forma – 1NF*
    - Data obsažená v jednotlivých sloupcích tabulky musejí být atomické. Pojmem atomické je myšleno, že neexistuje více skupin hodnot v rámci jednoho sloupce.
2. *normální forma – 2NF*
    - Splňuje všechny požadavky 1. normální formy.
    - Každý neklíčový sloupec musí být závislý na celém primárním klíči tabulky. V případě primárního klíče složeného z více sloupců, nesmí žádný z neklíčových sloupců záviset pouze na jedné části složeného primárního klíče.
3. *normální forma – 3NF*
    - Splňuje všechny požadavky 2. normální formy.
    - Všechny sloupce jsou přímo závislé na primárním klíči. Tabulka porušuje 3NF pokud je jeden sloupec závislý na jiném sloupci, který je závislý na primárním klíči, jedná se tady o takzvanou přechodnou závislost.

3.7.2 Pravidla tvorby databázových tabulek

Dle datové normalizace můžeme uvést několik zásadních pravidel pro tvorbu tabulek (Viescas, 2005):

1. Každé pole v tabulce by mělo představovat jedinečný typ informace.
2. Každá tabulka musí mít jednoznačný identifikátor neboli primární klíč, který je vytvořen z jednoho nebo více polí v této tabulce.
3. Pro každou jedinečnou hodnotu primárního klíče se musí hodnoty v datových sloupcích týkat předmětu tabulky a musí tento předmět úplně popisovat.

4. Musíme být schopni provést změnu dat v libovolném poli (jiném než pole v primárním klíči) bez toho, že by byla ovlivněna data v jakémkoliv jiném poli.

3.7.3 Denormalizace

Datová normalizace je proces zbavení se redundantních dat a ubezpečení se, že relace mezi tabulkami jsou smysluplná. Mohou však nastat situace, kdy je účelné pravidla normalizace porušit (Chapple, 2006).

Datovou normalizací dochází k nárůstu počtu tabulek a vazeb mezi nimi, normalizovaná data tak nejsou optimalizována k výkonnosti databázového systému z pohledu doby zpracování dotazů (Vostrovský, 2004):

Většina případů porušení pravidel je způsobena manipulací s databázovým návrhem kvůli dosažení vyšší rychlosti u určitých kritických úloh. Například, přestože mají moderní relační databázové systémy (jako je Microsoft ACCESS) velmi dobré výsledky při spojování mnoha vztažených tabulek u velmi složitých úloh,, můžeme narazit na určité situace, ve kterých není rychlost v případě spojených tabulek dostatečná. Někdy lze požadované rychlosti dosáhnout prostřednictvím určité *„denormalizace“* vybraných částí systémů. Například místo vytváření samostatné tabulky kódů kategorií produktů, která vyžaduje vazbu, lze umístit klasifikační popisy přímo do tabulky s produkty.

Dalším případem porušení pravidel je selektivní použití vypočítaných hodnot v databázi. Jestliže určitá sestava, například pro vedení podniku, vyžaduje u všech objednávek vypočítané celkové součty, avšak data se při sčítání jednotlivých hodnot z tisíců objednávek čtou jen velmi pomalu, bude nejspíš vhodné přidat do tabulek další pole pro celkový součet.

V případě požadavku na zachycení dat v časových bodech je též někdy nutno pravidla normalizace částečně porušit. Typickým příkladem jsou záznamy objednávek zboží a produktů, u kterých se cena může měnit. V tabulce objednávek je nezbytné umístit pole, které bude zaznamenávat cenu v časovém bodu, kdy byla objednávka uzavřena.

Jedním z dalších případů porušení pravidel je shromažďování dat určených pro tisk sestav. Pokud jsou základem sestavy poměrně komplikované dotazy a databáze obsahuje velké množství dat, může provedení příslušného dotazu zabrat nepřijatelně dlouhou dobu. V tomto případě je přijatelné vytvořit dočasné, „pravidla porušující“, tabulky, které se naplní výsledky složitého dotazu, nad nimiž pak lze spustit vlastní sestavu (Viescas, 2005).

3.8 Datové typy

Datové typy používané v systémech relačních databází (Groff, 2006):

- *Celá čísla* – sloupce obsahující tento typ dat typicky ukládají počet, množství, věk, atd. Celočíselné sloupce se rovněž často používají k ukládání identifikačních čísel, jako např. číselníků, zaměstnanců, objednávek.
- *Desetinná čísla* – sloupce s tímto datovým typem ukládají desetinná čísla, která mají desetinná místa a musí se vypočítat přesně, jako např. sazby a procenta. Rovněž se často používají k ukládání peněžních částek.
- *Čísla s plovoucí desetinnou čárkou* – sloupce s tímto datovým typem se používají k ukládání vědeckých čísel, která mohou být vypočítána přibližně, jako např. váhy a vzdálenosti. Čísla s plovoucí desetinnou čárkou mohou reprezentovat větší rozsah než desetinná čísla, ale ve výpočtu může dojít k chybám při zaokrouhlování.
- *Řetězce znaků s pevnou délkou* – ve sloupcích tohoto datového typu se zpravidla ukládají jména lidí a názvy společností, adresy, popisy atd.
- *Řetězce s proměnnou délkou* – tento datový typ umožňuje ukládat do sloupce řetězce znaků, jejichž délka se liší z řádku na řádek až po určitou maximální délku.

- *Peněžní částky* – většina databázových systémů podporuje typ MONEY nebo CURRENCY, jež se obyčejně ukládají jako desetinné číslo nebo číslo s plovoucí desetinnou čárkou. Existence různých peněžních typů umožňuje databázovému řídicímu systému, aby správně zformátoval peněžní částky při jejich zobrazení.
- *Data a časy* – databázové systémy rovněž běžně podporují hodnoty datum/čas, i když se podrobnosti mohou výrazně lišit z produktu na produkt. Obecně jsou podporovány různé kombinace data, času, časové známky, časového intervalu a aritmetiky data/času.
- *Logická data* – některé databázové systémy, jako např. Informix Dynamic Server podporují logické (TRUE nebo FALSE) hodnoty jako explicitní typ a jiné povolují logické operace (porovnání, logické operátory AND/OR atd.) na uložených datech v rámci příkazů SQL.
- *Dlouhý text* – několik databází založených na jazyce SQL podporuje sloupce, jež ukládají dlouhé textové řetězce (až do 32 000 nebo 65 000 znaků a v některých případech dokonce ještě delší). Tím nám umožní ukládat do databáze celé dokumenty, popisy produktů, technické zprávy, rekapitulace a podobná nestrukturovaná textová data. Databázové řídící systémy obvykle obyčejně omezují použití těchto sloupců na interaktivní dotazy a vyhledávání.
- *Nestrukturovaný tok bajtů* – několik databázových řídicích systémů povoluje ukládat a vyhledávat nestrukturované posloupnosti bajtů s proměnnou délkou. Sloupce obsahující takováto data se používají k ukládání komprimovaných video obrázků, proveditelného kódu a dalších typů nestrukturovaných dat. Například datový typ IMAGE v SQL Server může ukládat tok až do velikosti 2 GB dat.

- *Znaky mimo latinku* – jakmile databáze začaly výrazněji podporovat globální aplikace, výrobci databázových řídících systémů začlenili podporu pro řetězce 16bitových znaků s pevnou a proměnlivou délkou používané k reprezentaci Kanji nebo jiných asijských a arabských znaků. Zatímco nejmodernější databáze podporují ukládání a vyhledávání takových znaků (které většinou reprezentují s použitím konvencí UNICODE), v těchto typech GRAPHIC a VARGRAPHIC se výrazně liší podpora pro vyhledávání a třídění.

3.9 Dotazovací jazyk

Těžiště uživatelovy práce s databázovým prostředkem spočívá v možnosti tvorby uživatelských dotazů na data, která jsou uložena v bázi dat. Každý takovýto dotaz je ve své podstatě vlastní funkcí, která z dané množiny všech údajů báze dat vybírá takovou podmnožinu, která je pro uživatele v daném okamžiku důležitá. Tyto dotazy jsou předurčeny existujícím dotazovacím prostředkem neboli dotazovacím jazykem, který představuje souhrn prostředků umožňující snadné zadávání úloh pro relační databáze. Podle způsobu zadávání těchto dotazů rozlišujeme dotazovací prostředky na jazyky procedurální a neprocedurální. Procedurální jazyky se vyznačují tím, že je nutné zadat algoritmus pro získání požadované odpovědi. Neprocedurální jazyky jsou mnohem jednodušší, neboť vyžadují pouze zadat podmínky, které má požadovaná odpověď splňovat. V praxi se používají především tyto dva typy neprocedurálních jazyků (Vostrovský, 2004):

- *Jazyk SQL* (Structured Query Langure) – strukturovaný dotazovací jazyk, při jeho tvorbě byla dodržena zásada přiblížit specifikování dotazu principu kladení otázek v přirozeném jazyce, tj. v angličtině.
- *Jazyk QBE* (Query By Example), umožňuje zadávání dotazů v grafické podobě většinou na základě interaktivní komunikace

mezi databázovým systémem a uživatelem, přičemž vlastní dotaz je sestavován pomocí určitých symbolů zapisovaných do dotazovacích formulářů.

3.9.1 Jazyk SQL

Jazyk SQL je nástroj pro organizování, správu a získávání dat uložených v počítačové databázi. Zkratka SQL znamená *strukturovaný dotazovací jazyk* (Structured Query Language). Jak vyplývá z názvu, SQL je počítačový jazyk, který se používá pro komunikaci s relační databází (Groff, 2005).

Dotazovací jazyk SQL obsahuje příkazy, se kterými lze mimo získávaní uložených dat, řídit všechny funkce databázového systému. Lze s ním definovat strukturu a organizaci uložených dat a definovat jejich vzájemné vztahy. Pomocí SQL může uživatel nebo aplikační program přidávat nová data, aktualizovat nebo odstraňovat již uložená data. Řídit přístup jednotlivých uživatelů k databázi pomocí omezení jejich práv pro čtení, vkládání a modifikaci dat. Jazyk SQL také chrání databázi před porušením integrity dat, které by mohlo vzniknout neúplnou aktualizací nebo selháním systému. Jazyk SQL obsahuje asi 40 příkazů určených pro řízení databáze, není tedy klasickým programovacím jazykem jako například C++ nebo Visual Basic. Na rozdíl od těchto jazyků neobsahuje příkazy pro řízení běhu programu jako např. GOTO, FOR nebo DO, ani podmínkové příkazy typu IF.

Hlavní rysy a výhody jazyka SQL (Groff, 2005):

- Nezávislost na prodejci
- Přenositelnost mezi počítačovými systémy
- Standardy jazyka SQL
- Podpora a angažovanost firmy IBM (DB2)
- Angažovanost firmy Microsoft (SQL Server, ODBC a ADO)
- Relační základy
- Vysokoúrovňová struktura podobná angličtině
- Interaktivní dotazy, dotazy „ad hoc“

- Programový přístup k databázi
- Vícenásobné pohledy na data
- Kompletní databázový jazyk
- Dynamické definování dat
- Architektura klient/server
- Podpora podnikových aplikací
- Rozšiřitelnost a objektová technologie
- Internetový databázový přístup
- Integrace s jazykem Java (JDBC)
- Průmyslová infrastruktura

3.10 Microsoft ACCESS

Databázové programy pro osobní počítače jsou dostupné již dlouho. Bohužel mnoho z těchto programů bylo buď jednoduchými správci pro ukládání dat, které nejsou vhodné pro budování aplikací, nebo složitými systémy pro vývoj aplikací, které se obtížně učí a používají. Složitějším databázovým systémům se vyhýbalo dokonce i mnoho počítačově vzdělaných lidí, pokud nebylo potřeba vytvořit úplnou a uživatelsky přizpůsobenou databázovou aplikaci. *Microsoft ACCESS* však představuje významný obrat ve snadnosti používání a přitahuje mnoho lidí, kteří chtějí vytvářet jak jednoduché databáze, tak i složité databázové aplikace (Viescas, 2005).

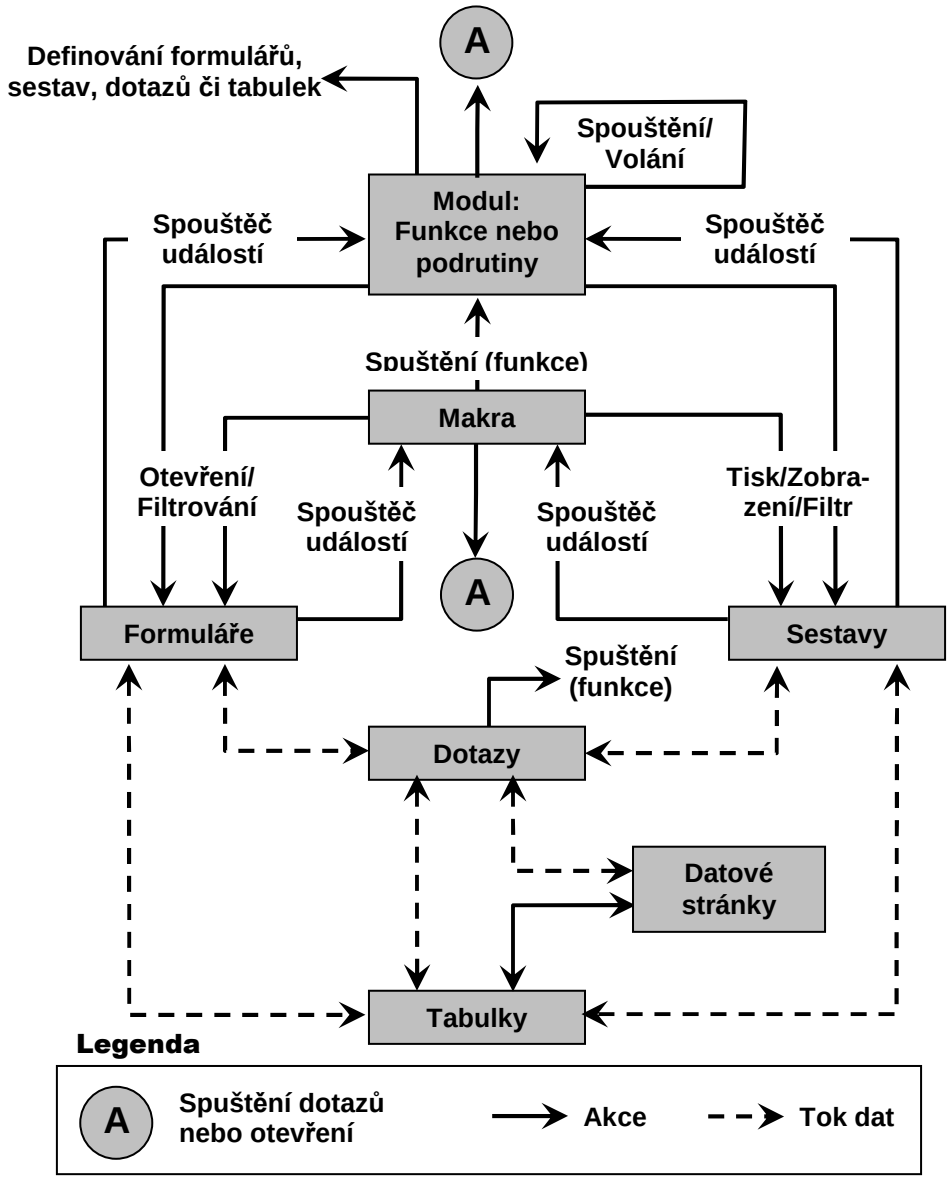
MS ACCESS je databázový prostředek vyvinutý společností Microsoft s myšlenkou poskytnout běžnému uživateli dostatečný prostředek, který od něho nebude vyžadovat speciální znalosti programovacích technik a umožní mu uspokojivě zpracovávat evidované údaje. Navíc jej tato firma obdařila totožným grafickým prostředím jako ve Windows (Vostrovský, 2004).

Pomocí aplikace Microsoft ACCESS můžeme (Kruczek, 2007):

- Uchovávat data (například kontakty klientů, účetní doklady, kartotéky, docházku zaměstnanců, jízdní řády, technické zprávy, smlouvy a mnoho dalších).

- Pracovat s daty (přepočítat, uspořádat, měnit, doplňovat).
- Zobrazovat data (na obrazovce, pro tiskárnu, prohlížet data z různých pohledů).
- Automatizovat práci s daty (vytvářet programové kódy).
- Sdílet data mezi uživateli.

3.10.1 Architektura MS ACCESS



Obrázek č. 2 Hlavní objekty a jejich relace v aplikaci ACCESS (Viescas, 2005)

3.10.1.1 Tabulky

Hlavní součástí databáze ACCESS jsou, tak jako u ostatních databází, *tabulky*, ve kterých jsou uložena všechna uživatelská data. Data jsou zapsána v řádcích neboli záznamech, rozdělených do sloupců dle datových typů a druhu informace. V tabulkách lze definovat primární klíče pro jednoznačnou identifikaci uložených záznamů a indexy pro zrychlení výběru dat (Viescas, 2005).

3.10.1.2 Dotazy

ACCESS nabízí celou řadu *dotazů*, které můžeme sestavit ručně nebo je naprogramovat. Dotazy použijeme ke zobrazení dat různými způsoby nebo ke změnám či analýze dat. Dotazy můžeme také využít jako zdrojů dat pro formuláře, sestavy a stránky.

V ACCESSu rozeznáváme následující typy dotazů (Held, 2006):

1. *Výběrové dotazy* – pomocí výběrového dotazu vyvoláme data z jedné nebo více tabulek a výsledky zobrazíme v datovém listu. V něm můžeme datové záznamy aktualizovat. Výběrovým dotazem je možno datové záznamy rovněž seskupovat a vypočítávat součty, počty záznamů, průměry a jiné hodnoty.
2. *Parametrické dotazy* – parametrický dotaz je takový, kdy se pro zadání informací zobrazí dialog. Informacemi mohou být kritéria pro vyvolání datových záznamů nebo hodnoty, které se mají přidat do nějakého pole. Dotaz můžete navrhnout i tak, že budete zadávat i více informací.
3. *Křížové dotazy* – křížových dotazů použijeme k výpočtům a nové strukturalizaci dat pro zjednodušení analýz. Křížové dotazy umějí vypočítat součet, průměr, počet záznamů nebo jinou funkci. Data jsou seskupována ve dva informační typy: podél levé strany datového listu a podél horní stany.

4. *Akční dotazy* – tyto dotazy provádějí pouze v jedné operaci změny na velkém množství datových záznamů nebo je posouvají. Akční dotazy jsou následujících typů:
   a) *Odstraňovací dotazy* – pomocí odstraňovacích dotazů odstraníme skupinu datových záznamů z jedné nebo více tabulek. Odstraňovací dotazy odstraňují vždy kompletní datové záznamy, nikoli pole označené v datových záznamech.
   b) *Aktualizační dotazy* – aktualizační dotaz provede globální změny na skupinách datových záznamů v jedné nebo více tabulkách. Aktualizačním dotazem můžeme data v disponibilních tabulkách měnit.
   c) *Přidávací dotaz* – přidávací dotaz přidá skupinu datových záznamů z jedné nebo více tabulek na konec jedné nebo více tabulek.
   d) *Vytvářecí dotazy* – vytvářecí dotaz vytvoří tabulku ze všech dat nebo z části dat v jedné nebo ve více tabulkách.
   e) *Dotazy SQL* – dotaz SQL je dotaz vytvořený použitím instrukce SQL. Pomocí SQL lze relační databáze, např. ACCESSu, dotazovat, aktualizovat a spravovat.

3.10.1.3 Formuláře

Formuláře jsou efektivním nástrojem pro práci s daty v tabulkách a celé databázi. Formuláře jsou databázové objekty, pomocí kterých můžeme zadávat, měnit nebo zobrazovat data z tabulek či dotazů. Formuláře však mohou zobrazovat také jen část tabulky a usnadnit tak uživateli orientaci při práci s databází. Pole tabulek se mohou při vkládání předvyplnit, na formulář můžeme přidat tlačítka pro další funkcionalitu nebo můžeme vložit obrázky a další grafické objekty a udělat tak databázové prostředí uživatelsky co nejpříjemnější. Formuláře si můžeme představit jako „okna“, přes která uživatel přistupuje k databázi. Efektivní využívání formulářů může uživatelům urychlit práci s databází, protože mají vždy vše důležité při ruce. Další výhodou formulářů je, že poskytují větší možnosti kontroly zadávaných dat, čímž

můžeme lépe zabránit vstupu nekorektních dat do tabulek. Každý formulář pracuje vždy nad jednou tabulkou či dotazem. Pro složitější práci je možné použít podformuláře (Kruczek, 2007).

3.10.1.4 Sestavy

Sestavy nám nabízejí možnost působivé prezentace dat v tištěné formě. Při jejich vytváření můžete sami stanovit velikost a zobrazení všech součástí sestav. ACCESS rozlišuje různé typy sestav. Všechny můžeme vytvořit s pomocí průvodce sestavou (Held, 2006):

1. *Detailní sestavy* – ke každému datovému záznamu existuje datový záznam v detailní sestavě. Tyto záznamy se potom podle stanovených hledisek seskupují a umožňují vytvoření úplného výkazu. S pomocí detailních sestav je možno zodpovědět například dotazy ohledně pohybů skladu, prodeje, objednávek atd. v daném časovém úseku.
2. *Sestavy se seskupením* – u těchto sestav se už všechna data netisknou. Data se seskupí v předběžném poli. Jako výsledek obdržíte jen několik málo řádků, tzn. zhuštěné zobrazení.
3. *Sestavy s diagramy* – u tohoto druhu sestav se data dodatečně zpracují do diagramu. Rovněž tuto úlohu můžeme velmi rychle vyřešit, aniž bychom napsali jediný řádek VBA.
4. *Formulářové sestavy* – průvodce sestavou podporuje rovněž vytváření sestav, které mohou vyhlížet jako formuláře.
5. *Sestavy se štítky* – pomocí Průvodce štítky můžete vytvořit štítky, přičemž určíte jejich velikost a rozměry stran.

3.10.1.5 Datové stránky

Datové stránky jsou obdobné jako formuláře, ale jsou uloženy v HTML kódu, je tedy možné je zobrazovat ve webovém prohlížeči a publikovat například na intranetu. Datové stránky tak umožňují sdílet data i s uživateli, aniž by museli mít nainstalovaný program pro ovládání databáze.

3.10.1.6 Makra

Objekt, který je strukturovanou definicí jedné nebo více akcí, které má ACCESS vykonat jako odezvu na nějakou definovanou událost. *Makra* mohou obsahovat jednoduché podmínky, které určují, kdy a jaké akce mají být provedeny. Makra mohou otevírat a provádět dotazy, otevírat tabulky nebo tisknout či prohlížet sestavy. Pomocí maker lze též spouštět další makra nebo procedury VBA (Viescas, 2005).

3.10.1.7 Moduly

Moduly jsou uživatelské procedury, které jsou napsány v jazyce Visual Basic ACCESS. Moduly mohou být přidruženy k libovolnému ovládacímu prvku ve formuláři nebo sestavě, nabízejí široké možnosti ovládání chodu databázové aplikace jako například spouštění dotazů, otevírání formulářů a sestav, filtrování, mazání a upravování záznamů. Pomocí kódu napsané ve Visual Basicu lze nejen vytvářet, mazat a upravovat objekty aplikace ACCESS, ale přes rozhraní Windows (API) můžeme spouštět i libovolné ostatní aplikace, se kterými může naše databázová aplikace dále komunikovat (Viescas, 2005).

4. Informační zabezpečení provozu automobilových parků

4.1 Automobilový park společnosti

Pojmem automobilový park společnosti rozumíme flotilu automobilů užívaných zaměstnanci společnosti k plnění pracovních povinností. Vozidla mohou být pracovníkům svěřovaná do dlouhodobého užívání nebo pouze k vykonání konkrétní služební cesty. Pokud má zaměstnanec svěřený vůz do dlouhodobého užívání je mu většinou umožněno používat vozidlo i k soukromým účelům.

Pořízení a provoz vozového parku představuje pro většinu společností nemalé náklady, proto se firmy snaží nalézt nejoptimálnější způsob financování. Vozidla mohou být pořízena na úvěr, finanční leasing nebo koupí za hotové, avšak stále více firem dává v dnešní době přednost takzvanému operativnímu leasingu. Operativní leasing je produkt, který začaly leasingové společnosti nabízet v České republice před necelými deseti lety. Na rozdíl od ostatních forem financování se u operativního leasingu vozidlo nestává po zvolené době majetkem společnosti, ale jedná se pouze o formu dlouhodobého pronájmu vozidla spojeného s poskytováním dalších služeb při zajišťování provozu vozového parku. Dle uzavřené smlouvy leasingová společnost zajistí nákup, přihlášení a pojištění vozidel, dále během provozu zajišťuje pravidelné servisní prohlídky, řeší pojistné události a opravy vozidel, sezónní výměnu pneumatik a jejich uskladnění, a v neposlední řadě také platbu veškerých poplatků a daní spojených s provozem vozidla. Společnosti využívající výhody operativního leasingu tak odpadá mnoho úkonů spojených s provozem služebních vozidel a platí pouze faktury leasingové společnosti za pronájem vozidel. Tento způsob pronájmu vozidel je pro společnost pohodlnější, avšak zajištěním provozu automobilového parku vlastními prostředky může společnost značně ušetřit, ale potřebuje k tomu dostatečně robustní databázový nástroj k evidenci všech důležitých údajů.

4.2 Požadavky na informační systém

Důležitým předpokladem úspěšné implementace informačního systému je jeho přehlednost, integrita uložených dat, dostatečná podrobnost a snadné a logické ovládání, které si jsou běžní uživatelé schopni rychle osvojit. Databázová aplikace správy vozového parku musí poskytovat ucelený obraz nejen o jednotlivých užívaných vozidlech, ale měla by obsahovat i informace spojené s úkony nákupu, provozu a vyřazení vozidel. Systém by neměl pouze zobrazovat vybraná data, ale dynamicky komunikovat s uživateli, upozorňovat na aktuální platby, nutnosti servisních prohlídek, vytváření statistik spojených s provozem vozidel jako jsou například přehledy nákladů na opravy a údržbu, počty ujetých kilometrů za vybraná časová období a podobně.

4.2.1. Seznam sledovaných událostí a procesů

Provoz automobilového parku řeší především následující typy událostí a procesů, které databázová aplikace musí umět evidovat:

- *Objednání vozu* – detailní specifikace objednaného vozu, kontaktní informace prodejce, forma financování.
- *Koupě vozu* – ověření všech technických údajů o voze, skutečná pořizovací cena, splátkový kalendář leasingu.
- *Převzetí vozu a jeho registrace* – údaje z technického průkazu.
- *Pojištění vozu* – pojišťovací společnost, číslo havarijní pojistky a zákonné odpovědnosti za škodu, výše pojistky.
- *Předání vozu uživateli* – údaje o zaměstnanci, datum a čas předání vozu, údaje o aktuálním stavu tachometru.
- *Platba leasingových splátek* – výše splátky, datum úhrady, číslo faktury.
- *Odebrání vozu uživateli* – údaje o zaměstnanci, datum a čas vrácení vozu, údaje o aktuálním stavu tachometru.
- *Pravidelná servisní prohlídka* – datum prohlídky, aktuální stav kilometrů, údaje o nákladech na prohlídku.

- *Škodná událost* – datum události, informace o viníkovi, způsob likvidace škody, výše škody, fotodokumentace škody.
- *Servisní oprava* – datum, popis závady, náklady na opravu.
- *Výměna registrační značky* – datum výměny, nová registrační značka.
- *Pravidelná evidence stavu najetých km* – datum, aktuální stav tachometru.
- *Vyřazení vozu z evidence a jeho prodej* – datum vyřazení, údaje o zůstatkové ceně při vyřazení vozu z majetku společnosti, prodejní cena.

5. Návrh informačního zabezpečení automobilových parků pomocí MS ACCESS

6.1. Strategie vlastního návrhu databáze

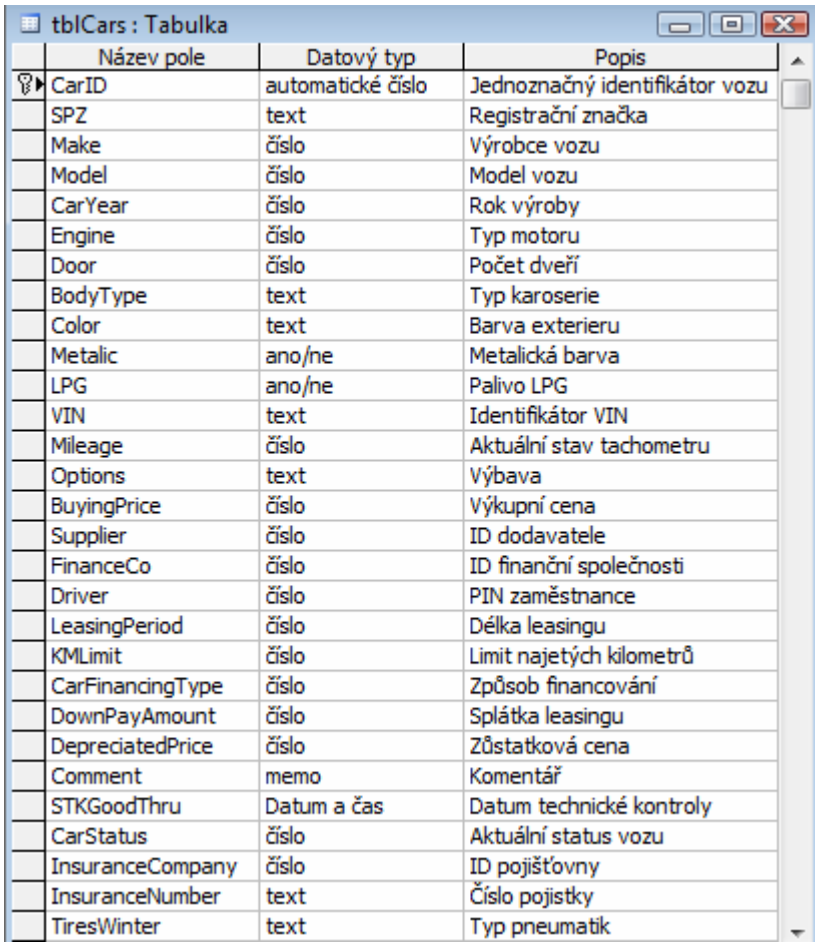
Tvorba databázového (informačního) systému začíná analýzou reálných procesů a toků dat, která vyústí do návrhů jednotlivých tabulek, názvů a datových typů položek, validačních podmínek, primárních (vlastních) klíčů, cizích klíčů a propojení tabulek (Kruczek, 2007).

Dvěma hlavními přístupy k návrhu databází jsou návrh řízený procesy (známý také jako návrh shora dolů), který se zaměřuje na funkce nebo úlohy, které je potřeba vykonávat, a návrh řízený daty (známý také jako návrh zdola nahoru), který se soustřeďuje na identifikaci a uspořádání všech jednotlivých dat, která potřebujeme. Při vlastním návrhu použijeme kombinaci myšlenek obou těchto metod (Viescas, 2005).

6.1.1. Tvorba tabulek a relací

Jak již bylo uvedeno dříve v textu, základním kamenem relačních databází jsou tabulky, pro ukládání záznamů. Při návrhu databázové aplikace správy automobilového parku společnosti budeme vycházet z tabulky dat popisujících vlastnosti jednotlivých vozidel. Této tabulce dáme název „tblCars“ a bude obsahovat mimo jiné název výrobce vozidla, model, rok výroby, typ motoru, typ karoserie, barvu exteriéru, typ převodovky, registrační značku, VIN (viz. Tabulka č. 1). Ve vytvořené tabulce je nezbytné určit primární klíč. Primární klíč slouží k jednoznačné identifikaci jednotlivých záznamů a propojení se záznamy z ostatních tabulek, v našem případě do tabulky přidáme pole s názvem CarID, které bude mít datový typ takzvané Automatické číslo, a toto pole určíme primárním klíčem tabulky. Vzhledem k úspoře objemu dat je vhodné pole výrobce, model a typ motoru definovat jako číselné hodnoty a k nim ekvivalentní slovní názvy evidovat v samostatných tabulkách, se kterými vytvoříme vazby.

Tabulka č. 1. Atributy hlavní tabulky



tblCars : Tabulka

Název pole	Datový typ	Popis
CarID	automatické číslo	Jednoznačný identifikátor vozu
SPZ	text	Registrační značka
Make	číslo	Výrobce vozu
Model	číslo	Model vozu
CarYear	číslo	Rok výroby
Engine	číslo	Typ motoru
Door	číslo	Počet dveří
BodyType	text	Typ karoserie
Color	text	Barva exterieru
Metalic	ano/ne	Metalická barva
LPG	ano/ne	Palivo LPG
VIN	text	Identifikátor VIN
Mileage	číslo	Aktuální stav tachometru
Options	text	Výbava
BuyingPrice	číslo	Výkupní cena
Supplier	číslo	ID dodavatele
FinanceCo	číslo	ID finanční společnosti
Driver	číslo	PIN zaměstnance
LeasingPeriod	číslo	Délka leasingu
KMLimit	číslo	Limit najetých kilometrů
CarFinancingType	číslo	Způsob financování
DownPayAmount	číslo	Splátka leasingu
DepreciatedPrice	číslo	Zůstatková cena
Comment	memo	Komentář
STKGoodThru	Datum a čas	Datum technické kontroly
CarStatus	číslo	Aktuální status vozu
InsuranceCompany	číslo	ID pojišťovny
InsuranceNumber	text	Číslo pojistky
TiresWinter	text	Typ pneumatik

Další důležité záznamy se týkají zaměstnanců společnosti, kterým může být svěřeno služební vozidlo. Tyto záznamy budeme evidovat v tabulce pod názvem „tblEmployee“, která bude obsahovat mimo jiné PIN zaměstnance (Personal Identification Number), jméno, příjmení, pozice, oddělení, kontaktní údaje a informaci o tom zdali je jedinec stále v aktivním pracovním poměru. Primárním klíčem této tabulky bude pole obsahující PIN zaměstnance, u kterého je zaručena jeho jedinečnost. Definujeme vazbu s tabulkou „tblCars“, do které přidáme další pole pod názvem „Driver“, jenž bude obsahovat PIN zaměstnance, kterému byl vůz svěřen. Tímto jsme vytvořili základ databáze a můžeme přejít k tvorbě dalších tabulek určených pro evidenci historických záznamů a ostatních dílčích informací.

Pro celkový přehled o vozovém parku je nutno v databázi evidovat stavy, ve kterých se jednotlivá auta nacházejí. V tabulce „tblCars“ vytvoříme číselné pole CarStatus, které bude nabývat hodnot odpovídajících jednotlivým slovním názvům statusům uložených v tabulce „tblCarStatus“. Při koupi nového vozu bychom měli dostupné informace co nejdříve zanést do databáze, dokud však výkup není zcela dokončen a vůz není připraven pro provoz je nezbytné záznam odlišit od ostatních vozidel. Prvním stavem, ve kterém se může vůz nacházet je tedy „Proces výkupu“. Když jsou dokončeny všechny náležitosti koupě a uvedení vozidla do provozu, uvedeme záznam v databázi do stavu „Čeká na alokaci“. Dalšími stavy mohou být „Přiděleno uživateli“, „V opravě“, „Odcizeno“ a v případě vyřazení z vozového parku „Na prodej“ a „Prodáno“. Pro tvorbu reportů historického přehledu počtu užívaných vozidel je nezbytné evidovat historii stavů, ve kterých se vozidla nacházela. Toho dosáhneme vytvořením tabulky „tblCarStatusHistory“, se kterou vytvoříme relaci pomocí cizího klíče „CarID“. V tabulce bude dále uložen datum, kdy byl stav vozu přidělen, číselná hodnota odpovídající statusu, a login uživatele, který status změnil. Jelikož status vozu je povinné pole a není tedy možné mít v databázi uložen záznam o vozidle, který by neobsahoval informaci, ve kterém stavu se vůz nachází, není nutné zaznamenávat datum, kdy byl status změněn. Tuto informaci vyčteme z následujícího záznamu tabulky historie statusů.

Obdobně vytvoříme tabulku historie uživatelů vozidla s tím rozdílem, že bude obsahovat i datum, kdy zaměstnanec vozidlo vrátil, protože každý vůz může být ve stavu, kdy jej nemá přidělený žádný zaměstnanec.

V tvorbě databáze budeme pokračovat tabulkami pro ukládání záznamů historie najetých kilometrů, kde budou uloženy informace o počtu ujetých kilometrů ke konkrétnímu datu. Tato informace je nezbytná pro dodržování intervalů servisních prohlídek vozů, proto je nutné, aby zaměstnanci v pravidelných intervalech zadávaly informace o aktuálním stavu najetých kilometrů dle palubních přístrojů vozidla.

Databáze musí být připravena i na škodné události způsobené provozem vozů. Další tabulka bude tedy obsahovat záznamy o pojistných událostech, ve

které bychom měli ukládat všechny informace nezbytné pro komunikaci s pojišťovnou. Měla by tedy mimo jiné obsahovat záznamy o datu, místě a popisu nehody, viníkovi, přepokládaných nákladech na opravu, čísla pojistné události, popřípadě výše spoluúčasti platby za způsobenou škodu.

V případě pořízení vozů na leasing či úvěr je vhodné mít v databázi evidované došlé faktury a jejich platby. V tabulce „tblInvoices“ vytvoříme pole číslo faktury, datum přijetí, částka, splatnost, dodavatel atd. Tuto tabulku opět propojíme s hlavní tabulkou vozů pomocí cizího klíče CarID.

Pro evidenci vedlejších nákladů spojených s provozem vozidel vytvoříme tabulku „tblExpensis“. V tabulce budou vedeny záznamy například o nákladech na pravidelné servisní prohlídky, opravy vozidel nehrazené pojišťovnou, pořízení dálničních známek, sezónních pneumatik, poplatky za rádio, silniční daň atd. v tabulce vytvoříme mimo jiné tato pole: datum vzniku nákladu, popis druhu nákladu a výše platby.

Při provozu vozidel může dojít k výměně registrační značky, proto bychom měli též evidovat jejich historii. V nově vytvořené tabulce „tblSPZHistory“ vytvoříme pole registrační značky a data přidělení. Opět vytvoříme relaci pomocí cizího klíče s tabulkou vozidel.

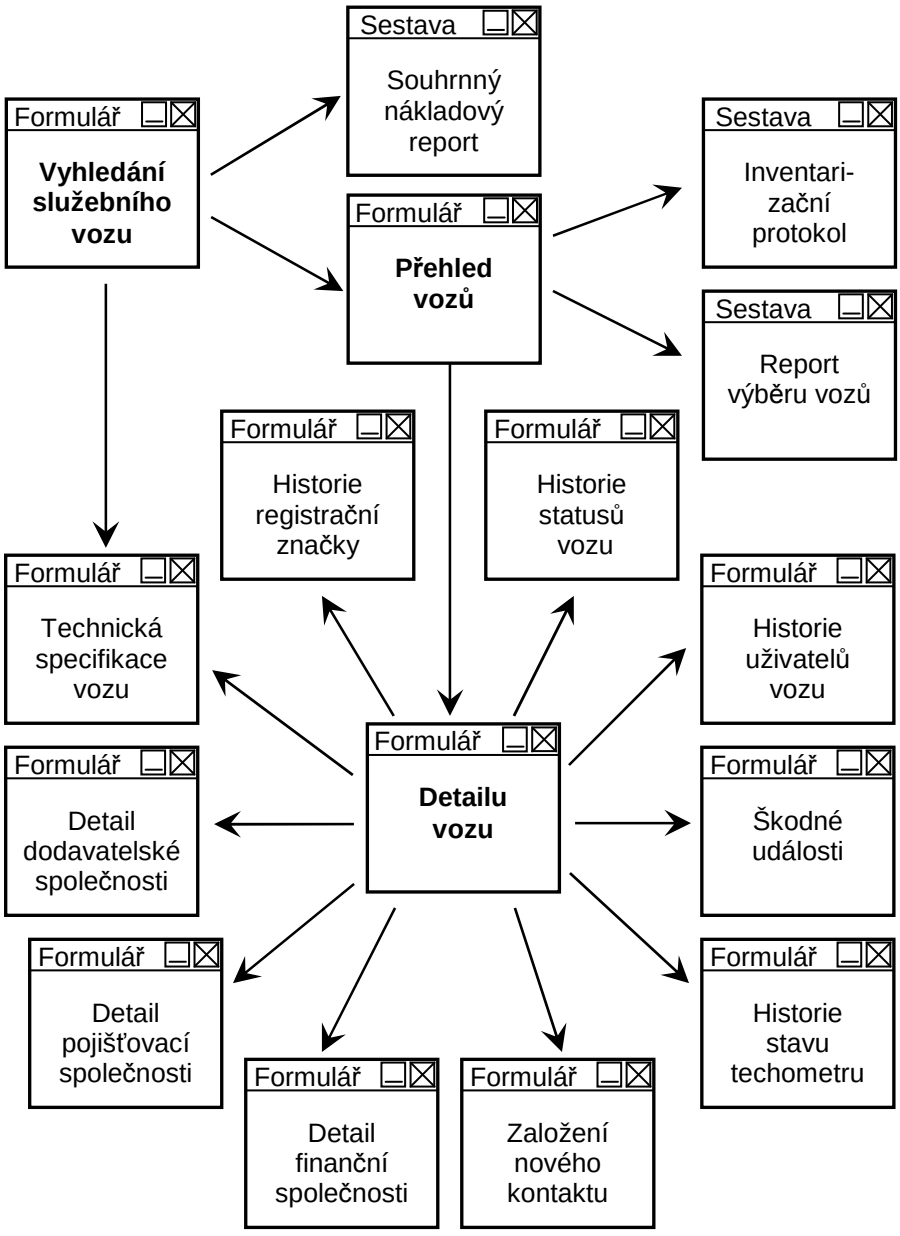
Ve většině případů nákupu vozidel je společnost, která nám vůz dodala různá od společnosti, která zajišťuje financování nákupu vozu. Lze předpokládat, že při zajišťování servisních prohlídek a oprav vozů budeme komunikovat s dodavatelem vozu, zatímco při placení faktur, ukončení leasingu a změnách ve financování bude komunikace probíhat s leasingovou společností. Proto je vhodné v tabulce vozidel založit pole „Dodavatel“ a pole „Finanční společnost“, které propojíme s dvěmi instancemi nově vytvořené tabulky „tblSuppliers“, která bude obsahovat kontaktní informace firem a pole umožňující rozlišení druhu kontaktu, zdali se jedná o dodavatele či leasingovou společnost.

Posední tabulka, která bude v přímé relaci s tabulkou vozidel, bude obsahovat informace o uložených rastrových souborech souvisejících s vozidly. Tabulka bude obsahovat pouze dvě pole, jedním bude cizí klíč tabulky „CarID“

a druhým bude označení názvu souboru, ve které je uložen příslušný dokument. Může se jednat například o elektronickou kopii technického průkazu, fotografie vozidla, případného poškození, elektronické kopie faktur, záznamy o provedených servisních prohlídkách a opravách.

Jednotlivé relace mezi tabulkami jsou vyznačeny v schematickém diagramu databáze (viz. Příloha č. 1).

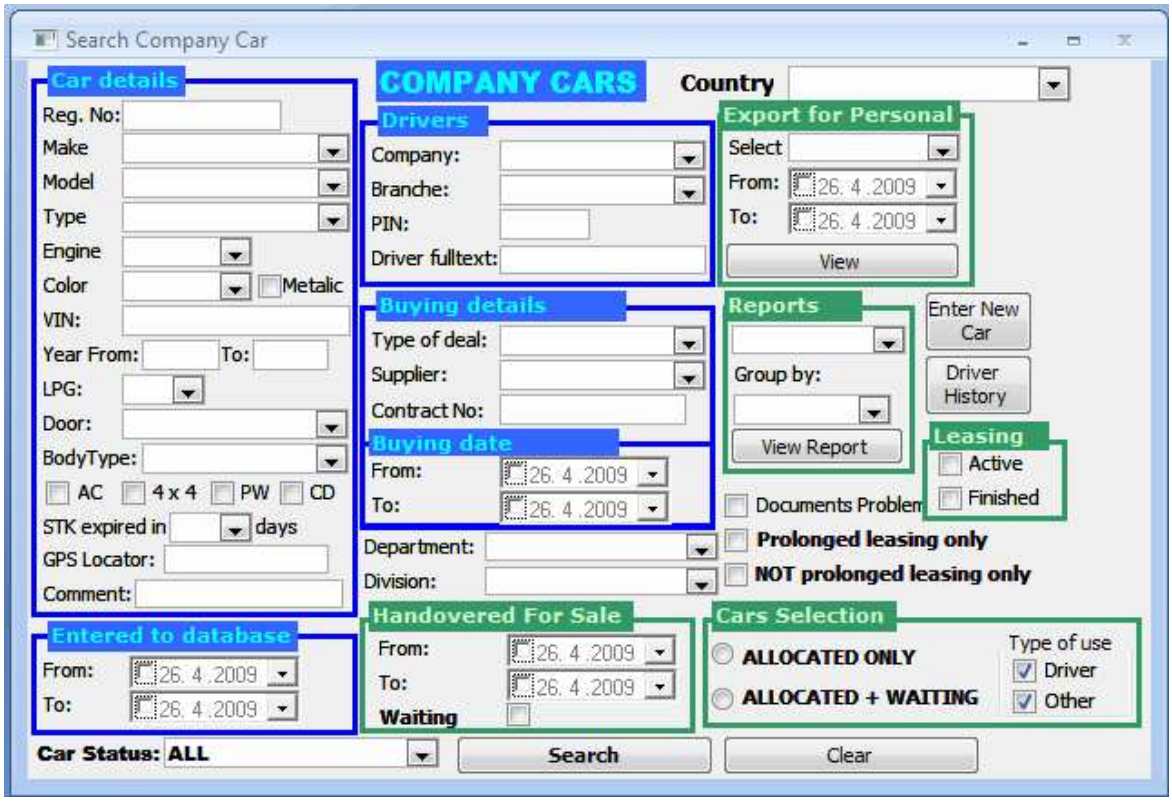
6.1.2. Tvorba uživatelského rozhraní aplikace



Obrázek č. 3 Schéma uživatelského rozhraní

6.1.2.1. Formuláře

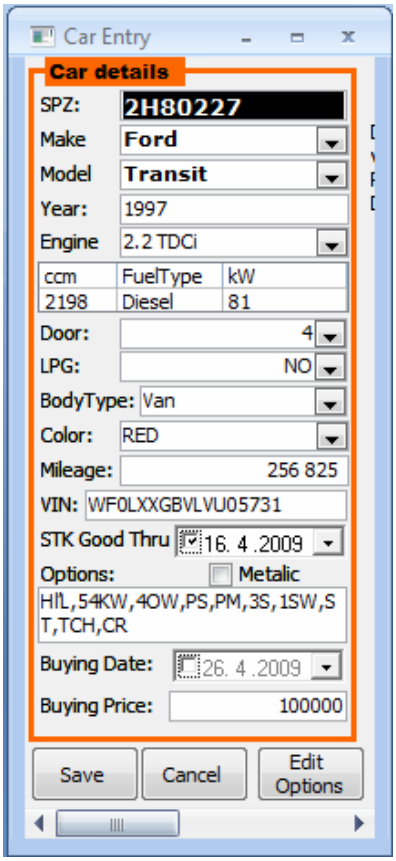
Základem uživatelského rozhraní je navržen formulář sloužící k vyhledávání vozů a spouštění sestav pro reportování (viz. Obrázek č. 4). Tento formulář je nastaven pro automatické zobrazení po spuštění systému a je tedy výchozím komunikačním prostředkem mezi uživatelem a databází. Umožňuje filtrovat jednotlivé vozy nebo skupiny vozů dle kritérií zadaných uživatelem, lze tedy vyhledat konkrétní vůz dle jeho registrační značky nebo filtrovat skupiny vozů vyhovující zadaným kritériím jako jsou například výrobce, model, rok výroby, ale také dle statusů, data pořízení vozidel atd. Vyhledávací formulář též umožňuje po zadání jména zaměstnance vyhledat právě přidělené vozidlo nebo také historický přehled vozidel, které zaměstnanec používal.



Obrázek č. 4 Hlavní vyhledávací formulář

Výstupem vyhledávání je formulář obsahující filtrovaný výpis jednotlivých vozidel splňující kritéria zadaná uživatelem (viz. Příloha č. 2). Tento výpis lze vytisknout pomocí předem definované sestavy nebo je možné jej exportovat do

aplikace MS Excel pro další zpracování. Pro rychlejší vyhledávání obsahuje tento formulář i pole pro filtrování dle registrační značky nebo jména zaměstnance. Poklepáním na jakýkoliv údaj v řádku konkrétního vozidla otevřeme kartu vozidla (viz. Příloha č. 3). Karta vozidla slouží k vyplňování všech údajů, které evidujeme v souvislosti s jeho provozem. Informace popisující základní fakta o vozidle jsou na této kartě nepřístupná pro editaci. Tyto údaje lze editovat na zvláštním formuláři primárně určenému k zadávání nových služebních vozů do databáze (viz. Obrázek č. 5). Dalšími formuláři přístupnými z karty vozu jsou formuláře pro zadávání historie uživatelů, statusů, stavu tachometru, škodných událostí, registračních značek vozu a kontaktních údajů finanční, pojišťovací a dodavatelské společnosti.



Obrázek č. 5 – Formulář vkládání a editace vozů

6.1.2.2. Sestavy

Sestavy slouží především k tisku reportů z aplikace. Pro snadný přehled o vozovém parku je účelné vytvořit přehledové reporty seznamu vozů rozdělených do skupin dle stavů, ve kterých se právě nacházejí. Stejnou, předem definovanou sestavu lze použít pro různé seznamy generované na základě nastavení filtru ve vyhledávacím formuláři. Snadno tak získáme reporty vozů čekajících na přidělení uživateli, aktivních vozů v užívání nebo seznam vyřazených vozů. Dále můžeme reporty použít pro souhrnné informace o výši celkových měsíčních splátek či nákladech na provoz skupiny vozů. Velkou výhodou sestav je též možnost připravení předem daných formulářů, sloužících například k účelu procesu inventarizace, kde se do jednotné matrice doplňují informace o voze a jeho uživateli (viz. Příloha č. 4)

7. Závěr

Závěrem lze konstatovat, že byla prokázána možnost vytvoření relačně databázové aplikace s praktickou využitelností za pomocí jak ekonomicky, tak i s ohledem na obtížnost tvorby, nenáročného nástroje Microsoft ACCESS. Program MS ACCESS je součástí standardní instalace kancelářského balíku od společnosti Microsoft ve verzi Professional, je tedy snadno dostupný a cenově značně výhodnější ve srovnání s ostatními programy pro tvorbu databázových aplikací.

Mezi další výhody aplikací vytvořených v ACCESSu patří i jejich, do určité míry, snadná distribuce mezi ostatní uživatele. Pro uživatelské stanice není nutná instalace programu MS ACCESS, ale postačí jednoduchá utilita s názvem MS ACCESS Runtime, který lze bezplatně zkopírovat z internetových stránek firmy Microsoft. Společnosti se tak nabízí možnost ušetřit značnou finanční částku zakoupením kancelářského balíku MS Office pouze ve verzi Standard, která je ve většině ostatních kancelářských aplikací plně dostačující.

Částečným omezení práce s databází pomocí programu Runtime je nemožnost úpravy struktury databáze, uživatelských formulářů ani sestav. Práce s uloženými daty jako je zadávání, editace, filtrování a tisk sestav je však možná pomocí předem definovaných formulářů sloužících k ovládání aplikace, navíc nejeden administrátor databázového systému uvítá znemožnění neoprávněných zásahů uživatelů do celého systému, bez nutnosti konkrétního ošetření. Navíc, nová verze systému Runtime 2007 byla vylepšena o snazší filtrování a řazení dat ve formulářích, které není nutno předem programem definovat, tak jako tomu bylo u starších verzí.

Databázové aplikace vytvořené pomocí programu MS ACCESS mají samozřejmě i své nevýhody, které jsou dané především určením tohoto systému pro kancelářské počítače. Ve většině případů je databáze tvořena za účelem sdílení dat mezi více uživateli. Databázovou aplikaci lze rozdělit na část datovou, kde se nacházejí tabulky, a aplikační, která obsahuje především formuláře, sestavy, dotazy a moduly. Toto řešení je však optimální pouze do

velikosti ca. 200 MB a pouze pokud není k databázi připojeno více než 20 uživatelů. Pokud není databáze rozdělena a je právě sdílena více uživateli, není umožněno upravovat její strukturu ani formuláře či sestavy. Tento fakt značně komplikuje správu a provádění změn v systému, kdy je nutné zajistit, aby žádný uživatel s databází v daném okamžiku nepracoval. Aplikace vytvořené v MS ACCESS též sklízejí kritiku za enormní zatěžování firemní sítě a to i v případě, kdy je databáze rozdělena na aplikační a datovou část, která je umístěna na společném serveru. Oproti jiným databázovým systémům, které odesílají pouze samotný SQL požadavek řešený serverem, se odehrává veškerá činnost, včetně řešení složitých dotazů, v klientských počítačích, čímž dochází k odesílání mnoha nízkoúrovňových povelů a tím značnému snížení výkonu celého sytému.

Odstraňování a přidávání objektů databáze dochází k fragmentaci databázového souboru, jehož velikost neustále narůstá a opět dochází ke značnému snižování výkonu, proto je nutné pravidelně databázi komprimovat. Komprimace je možná pouze v případě, kdy databáze není uživateli využívána, což opět způsobuje komplikace, které lze řešit například pravidelným automatickým spouštěním procedury na serveru v průběhu noci, kdy se nepředpokládá práce s databází.

Značně problematické, především u starších verzí MS ACCESS, je také použití některých ovládacích prvků ve formulářích. Jedná se hlavně o nestandardní prvky typu ActiveX jako je například hojně využívaný nástroj pro zadávání data pomocí kalendáře Date and Time Picker. Tento ovládací prvek například nelze umístit na formulář společně s ovládacím prvkem Karta, aniž by se při běhu programu nevyskytovali chyby. Tato neošetřená kombinace se může jevit jako nepodstatná, ale v případě tvorby komplikovanějších formulářů s velkým množstvím zadávaných údajů významně komplikuje tvorbu formulářů a aplikace se stává pro uživatele značně nepřehledná. Při distribuci databázových aplikací s použitím prvků ActiveX na formulářích dochází též k problémům s chybějícími nebo nekompatibilními verzemi knihoven operačního systému MS Windows na klientských počítačích. Tyto knihovny se

musejí složitě vyhledat a nahradit správnými verzemi, což značně komplikuje distribuci aplikace.

Pro konečné zhodnocení databázového programu Microsoft ACCESS vyjmenujme jeho hlavní výhody:

+ Nízké pořizovací náklady
+ Menší nároky na odbornost vývojových pracovníků
+ Nativní prostředí MS Windows
+ Nenáročná distribuce pomocí MS ACCESS Runtime
+ Snadná tvorba komplikovaných dotazů v grafickém prostředí
+ Rychlost vytváření nových aplikací

a nevýhody:

- Primární určení – kancelářský nástroj
- Podmíněná správa a údržba
- Občasná nekompatibilita ovládacích prvků
- Výkon vysoce závislý na počtu uživatelů

I přes některé popsané nedostatky převažují z pohledu informačního zabezpečení malých a středních společností jeho výhody, a proto je možné MS ACCESS pro tvorbu databázových aplikací doporučit. Těžko lze na dnešním trhu najít byť jen srovnatelný nástroj, kterým lze v relativně krátké době a bez znalosti programování, vytvořit téměř libovolnou aplikaci pro práci se sdílenými daty. Společnosti tak mohou ušetřit nejen při investicích do vývojového softwaru, ale i na mzdových nákladech softwarových vývojářů, u kterých není požadována vysoká odbornost při tvorbě databázových aplikací. Příkladem nechť je právě systém evidence vozových parků společnosti, která provozuje vlastní flotilu vozů využívanou zaměstnanci. Menší a střední společnosti mají většinou na výběr pouze ze tří řešení, nakoupit standardizovaný software od specializované firmy, nechat si na zakázku vyvinout software dle vlastních požadavků nebo pověřit vlastního zaměstnance tvorbou evidence. Standardizovaný software obvykle neodpovídá konkrétním požadavkům na evidenci a jeho přizpůsobení může být finančně náročné. Zakázkový vývoj software je též velká zátěž pro rozpočet menší firmy a při

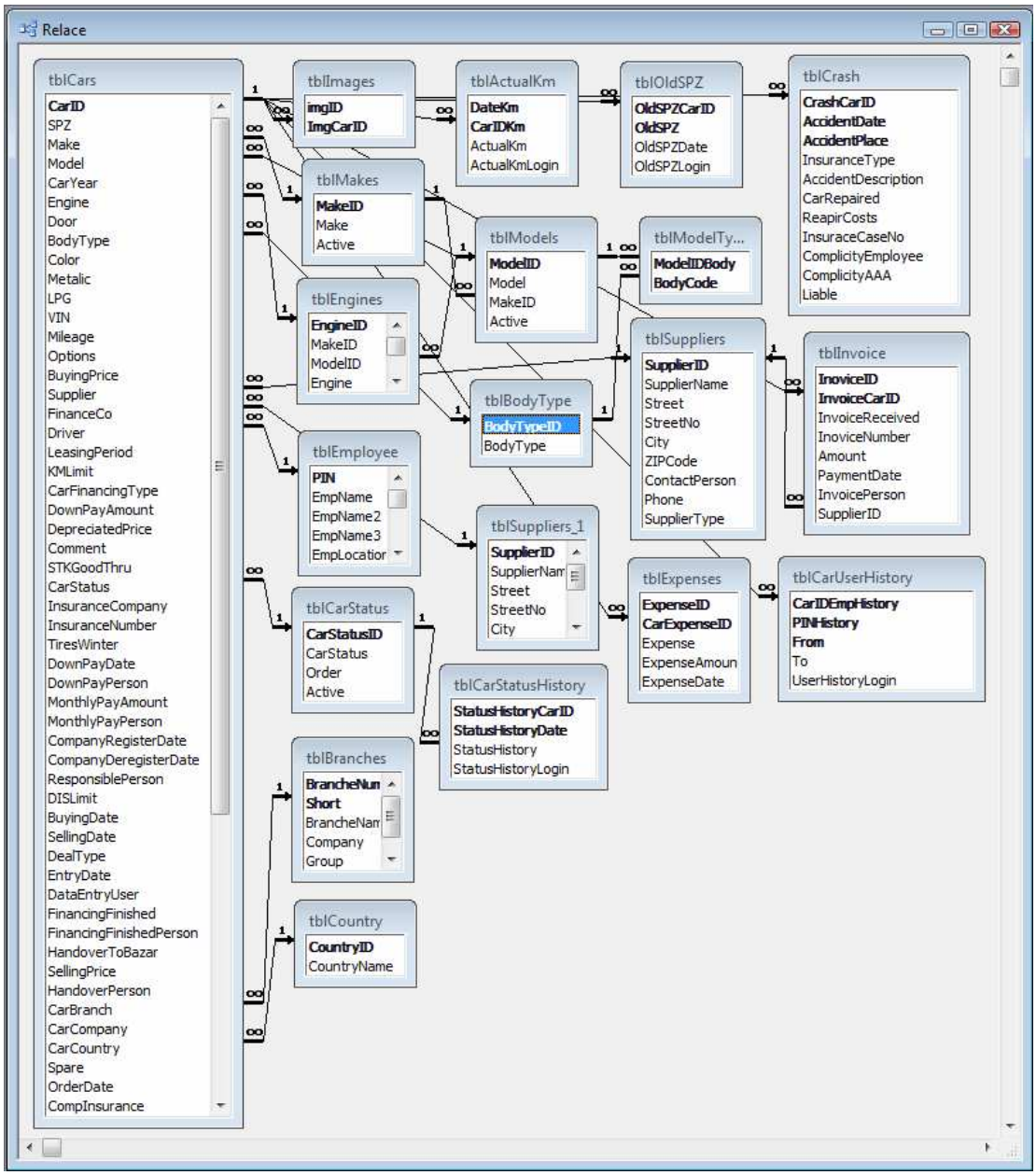
jakékoliv další úpravě je nutný kontakt vývojové společnosti spojený s další úhradou požadovaných změn. Jako nejvýhodnější se tedy jeví vývoj vlastního systému evidence služebních vozů nástrojem MS ACCESS.

Praktickým příkladem využití nástroje MS ACCESS je dříve popsaná databázová aplikace, jejímž hlavním tvůrcem je autor této práce, která v době vzniku tohoto textu sloužila k evidenci vozového parku nadnárodní společnosti. Společná databáze, do které přistupovali uživatelé v pěti různých evropských zemích, evidovala záznamy přibližně o osmi stech aktivních služebních vozech, které mohly být přiděleny kterémukoliv ze tří tisíc zaměstnanců, jejichž základní údaje byly součástí databáze. Uživatelské počítače připojené k síti LAN, kde se nacházel server s databázovým souborem, přistupovali k databázi pomocí aplikace spouštěné utilitou MS ACCESS Runtime. Uživatelé z ostatních zemí měli aplikaci k dispozici, z důvodu úspory objemu přenášených dat, pomocí vzdáleného klienta Citrix Metaframe. Nutno dodat, že veškerá data byla spravována přibližně dvaceti uživateli a průměrná perioda obměny jednotlivých vozů byla 6 měsíců.

8. Seznam literatury

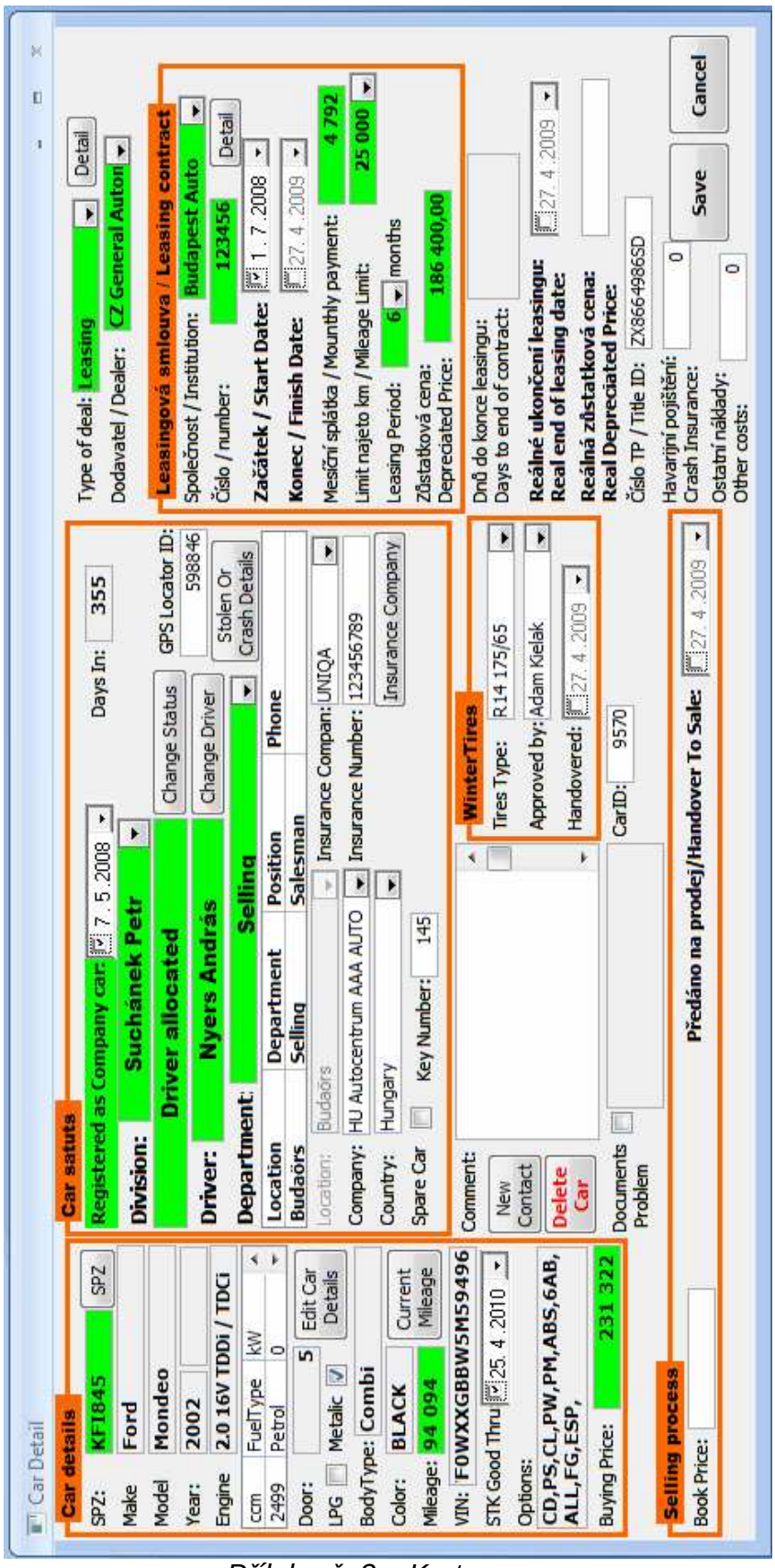
1. VIESCAS, J. L.: Mistrovství v Microsoft Office ACCESS 2003. Vydání první. Brno: CP Books, a.s. 2005. ISBN 80-251-0537-7
2. HELD, B.: ACCESS VBA: velká kniha řešení. Vydání první. Brno: Computer Press, 2006. ISBN 80-251-1112-1
3. GROFF, J. R., Weinberg, P. N.: SQL Kompletní průvodce. Brno: Computer Press, 2005. ISBN: 80-251-0369-2
4. KRUCZEK, A.: Microsoft Office ACCESS 2007, Brno: Computer Press, 2007. ISBN 978-80-251-1608-1
5. VOSTROVSKÝ, V.: Vytváření databází v ORACLE, 2004, Provozně ekonomická fakulta ČZU, ISBN 80-213-1191-6
6. FARANA, R.: Tvorba relačních databázových systémů, VŠB – Technická univerzita Ostrava, 1999, ISBN 80-7078-706-6
7. BERKA, P.: Dobývání znalostí z databází, Academica, nakladatelství Akademie věd ČR, 2003, ISBN 80-200-1062-9
8. POKORNÝ, J., HALAŠKA, I.: Databázové systémy, skripta FEL ČVUT, 2003
9. URL: <www.wikipedia.com>
10. HILLYER, M., 2005, URL: <www.mysql.com>
11. CHAPPLE, M., 2006, URL: <www.about.com>

9. Přílohy



Příloha č. 1 – Schematický diagram

Příloha č. 2 – Formulář přehledu vozů



Příloha č. 3 – Karta vozu

INVENTARIZAČNÍ PROTOKOL

Služební vozy k 31.12.2007

Datum a čas začátku a konce inventury:	31.12.2007
Misto provedeni inventury:	Brno
Inventarizační komise: Jméno pracovníka:	Jana Punčochářová

Status vozu:	Driver allocated
Odpovědná osoba:	Kejik Radovan
Pozice:	Sales Closing Manager
Odděleni:	Bazaar Management
Telefon:	91603

Značka automobilu	SPZ:
Ford Mondeo	**8A84559**

Já, níže podepsaný prohlašuji, že užívám tento služebni automobil, kterýmně byl přidělen, a kterýje k uvedenému datu provozu schopný.

....

podpis odpovědná osoba

Příloha č. 4 – Sestava inventarizační protokol