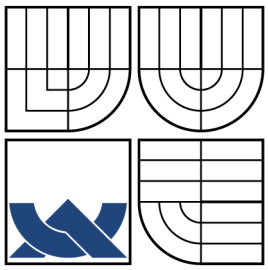
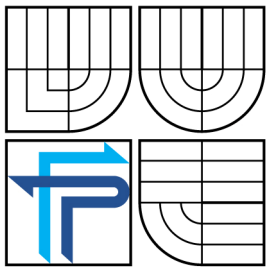


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
DEPARTMENT OF INFORMATICS

# NÁVRH DATABÁZE SVAZKŮ STROJE

DESIGN OF ELECTRIC MACHINE YOKE DATABASE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL SUCHOMEL
AUTHOR

VEDOUCÍ PRÁCE — ING. PETR DYDOWICZ, PH.D.
SUPERVISOR

BRNO 2009

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2008/2009
Ústav informatiky

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

**Suchomel Michal**

Manažerská informatika (6209R021)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává bakalářskou práci s názvem:

**Návrh databáze svazků stroje**

v anglickém jazyce:

**Design of Electric Machine Yoke Database**

Pokyny pro vypracování:

Úvod
Vymezení problému a cíle práce
Teoretická východiska práce
Analýza problému a současné situace
Vlastní návrhy řešení
Ekonomické zhodnocení, přínos návrhů řešení
Závěr
Seznam použité literatury
Přílohy

Seznam odborné literatury:

BASL, J. Podnikové informační systémy. 2002. 144s. ISBN 80-247-0214-s.
KOCH, M. DOVRTĚL, J.: Management informačních systémů. VUT FP Brno, 2006, 174 stran. ISBN 80-214-3262-4.
MOLNÁR, Z. Efektivnost informačních systémů. 2000. ISBN 80-7169-410-x.
ŘEPA, V. Analýza a návrh informačních systémů. 1999. 403s. ISBN 80-86119-13-0.
VLASÁK, R., BULÍČKOVÁ, S. Základy projektování informačních systémů. 2003. ISBN 80-246-0727-1.
VRANA, J. RICHTA, K. Zásady a postupy zavádění podnikových informačních systémů. 2005. 188 s. ISBN 80-247-1103-6.

Vedoucí bakalářské práce: Ing. Petr Dydowicz, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2008/2009.

L.S.

| | |
|---|---|
| Ing. Jiří Kříž, Ph.D.<br>Ředitel ústavu | doc. RNDr. Anna Putnová, Ph.D., MBA<br>Děkan fakulty |

V Brně, dne 27.05.2009

## ANOTACE

Bakalářská práce pojednává o návrhu na vytvoření databáze svazků stroje s textovým výstupem do modelovacího programu pro firmu Siemens Electric Machines s.r.o. Databáze pracuje na základě konkrétních požadavků dané firmy. Práce obsahuje vlastní návrh databáze i její vytvoření v programu Microsoft Access.

**Klíčová slova:**
Databáze, návrh databáze, Microsoft Access, Visual Basic for Applications, VBA

## ANNOTATION

This bachelor´s thesis deals with the project for building electric machine yoke database with text output to the modeling program for Siemens Electric Machines s.r.o. Database work based of concrete needs in the copany. Includes respective structure database and their creation in Microsoft Access.

**Keywords:**
Database, database design, Microsoft Access, Visual Basic for Applications, VBA

## BIBLIOGRAFICKÁ CITACE

SUCHOMEL, M. *Návrh databáze svazků stroje* . Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2009. 64 s. Vedoucí bakalářské práce Ing. Petr Dydowicz, Ph.D.

## PROHLÁŠENÍ

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná a neporušuje autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne ..... .....

podpis autora

## PODĚKOVÁNÍ

Děkuji tímto vedoucímu bakalářské práce panu Ing. Petru Dydowiczovi, Ph.D. za cenné připomínky a rady při vypracování bakalářské práce.

Dále bych chtěl poděkovat panu Ing. Janu Němcovi a panu Ing. Janu Šťastnému za pomoc při návrhu a vytváření databáze.

# Obsah

# Úvod

Ve své práci budu zkoumat, jak nejlépe navrhnout a vytvořit databázi svazků stroje pro konstruktéry společnosti Siemens Electric Machines s.r.o. Tato má sloužit nejen k přehlednému zobrazení již vytvořených svazků, ale také jako generátor textového výstupu parametrů pro modelovací program používaný ve firmě. Je žádoucí, aby byla databáze co nejkvalitnější a zároveň co nejjednodušší pro obsluhu zaměstnanci, proto bude databáze ovládána přes vstupní formulář, jehož funkce budou naprogramovány v jazyku Visual Basic for Applications (VBA). Uživatel nebude muset ručně zadávat data do tabulek, nebo psát složité dotazy pro vyhledávání, protože na všechny takové funkce budou naprogramovány tlačítka.

Nejprve se zaměřím na popis a analýzu současného stavu společnosti. V teoretické části vysvětlím základní pojmy související s databázemi a popíši program Microsoft Access 2003. V praktické části se pokusím navrhnout nejvhodnější databázi vzhledem k požadavkům společnosti, a podrobně ji popíšu a vysvětlím. V závěru své práce vše zhodnotím a odhadnu přínos mé práce pro firmu.

# 1 Vymezení problému a cíle práce

Ve výrobní společnosti Siemens Electric Machines s.r.o. mě požádali, abych analyzoval stav konstrukce svazků stroje. V současné době se celá konstrukce svazku musí provádět ručně s použitím obecných skic nebo modelů, což je neefektivní a složité. Popřípadě si konstruktér otevře starý svazek a upravuje ručně všechny hodnoty. Chybí tedy přehled vytvořených svazků stroje a možnost jejich efektivní úpravy. Proto jsem se rozhodl navrhnout a vytvořit databázi svazků stroje.

Cílem mé práce je vytvořit přehlednou a jednoduchou databázi svazků stroje s naprogramovanými funkcemi, která bude spolupracovat s dalšími aplikacemi. Při vytváření nového svazku načte databáze potřebná data z výpočtového listu v Excelu. Pokud bude konstruktér upravovat již vytvořený svazek, bude si moci načíst jeho data do formuláře. Databáze bude také generovat textový soubor, který bude specifikován pro načtení 3D grafickým modelovacím programem. Po načtení tohoto souboru bude mít konstruktér k dispozici hotový model svazků stroje. S modelovacím programem bude databáze komunikovat nejen jako výstupní, ale i jako vstupní. Namodelované svazky v grafickém programu bude možné pomocí textových souborů generovaných tímto programem nahrát do databáze.

Důležitou podmínkou firmy bylo, aby mohli databázi používat konstruktéři s běžnou znalostí PC. Proto jsem se rozhodl databázi vytvořit v programu Microsoft Access 2003, který je součástí balíčku Microsoft Office 2003. Tento balíček konstruktéři mají k dispozici na svých počítačích a běžně ho využívají, a proto většina pracovníků zná prostředí MS Access a umí se v něm orientovat.

Předpokládám, že vytvoření databáze svazů stroje povede ke zkvalitnění a zrychlení procesu konstrukce nejen svazku, ale i celého stroje. Konstrukce pak bude jednodušší, přehlednější a efektivnější.

# 2 Analýza problému a současné situace

## *2.1 Představení společnosti*

Nejprve chci představit společnost Siemens Electric Machines, ve které jsem svoji práci zpracovával.

### 2.1.1 Profil společnosti


*Obr. 1 : Logo Siemens*

Siemens Electric Machines s.r.o. je součástí celosvětového koncernu SIEMENS, který patří mezi největší globální elektrotechnické a elektronické koncerny. Koncern zaměstnává téměř 400 000 odborníků, kteří vyvíjejí a vyrábějí produkty, navrhují a instalují komplexní řešení na míru dle požadavků zákazníků a nabízejí širokou paletu služeb dle jejich individuálních potřeb. SIEMENS nabízí svým zákazníkům ve 190 zemích inovativní technologie a komplexní know-how. Tento koncern byl založen před 160 lety a působí v oblastech informace a komunikace, automatizace a pohony, energetiky, dopravy, zdravotnictví a osvětlení. V současné době patří SIEMENS s více než 18 500 zaměstnanci mezi největší zaměstnavatele v ČR.

Siemens Electric Machines s.r.o. (dále jen Siemens) sídlí v obci Drásov, asi 20 km od Brna a je společností s ručením omezeným. Vystupují za ni jednatelé firmy. Společnost je řízena mateřskou společností (řídící osobou se 100% vlastnictvím) Siemens Aktiengesellschaft (SIEMENS AG), se sídlem v Berlíně a Mnichově v Německu. Podnik má v současné době asi 600 zaměstnanců.

Společnost Siemens v Drásově je výrobní firmou. Hlavním programem společnosti je výroba a prodej speciálních vysokonapěťových asynchronních motorů, vysokonapěťových a nízkonapěťových generátorů a jejich komponent pro lodní, těžební, energetické a drážní aplikace. Společnost tedy vyrábí velké elektromotory a generátory, například pro lodě, lokomotivy, tramvaje, vodní a větrné elektrárny… Vše vyrábí v různých provedeních a upravuje podle přání zákazníků. Specializací Siemensu v Drásově je to, že vyrábí na zakázku, nemá tedy sériovou výrobu jako takovou.

### 2.1.2 Historie společnosti

Historie podniku začíná v roce 1912, kdy byl v Drásově založen závod na výrobu zemědělských strojů. Od roku 1918 se zde začaly opravovat elektromotory a počínaje rokem 1923 se začaly motory i vyrábět. Výroba motorů rostla a závod postupně přecházel pod BBC. Roku 1947 byl podnik zestátněn a v roce 1950 byl začleněn do sdružení MEZ jako jeho pobočka. V tomto sdružení podnik setrvával a dařilo se mu. V roce 1990 se závod osamostatnil v rámci koncernu ZSE Praha.

V roce 1994 byl podnik koupen firmou SIEMENS a začleněn do obchodní oblasti Large Drives (A&D LD) a postupně se zde rozvíjela výroba nízkonapěťových motorů a generátorů, později i vysokonapěťových generátorů. V roce 2001 byla založena společnost SEM Drásov Siemens Electric Machines s.r.o., a ta se roku 2006 přejmenovala na dnešní název Siemens Electric Machines s.r.o. Od roku 2007 zde probíhá i výroba asynchronních a synchronních motorů.[1]

### 2.1.3 Předmět podnikání

Předmětem podnikání společnosti je:

- koupě zboží za účelem jeho dalšího prodeje a prodej
- zprostředkovatelská činnost
- pořádání odborných kurzů, školení a jiných vzdělávacích akcí včetně lektorské činnosti
- zámečnictví
- kovoobráběčství
- nástrojařství
- výroba, instalace a opravy elektrických strojů a přístrojů
- projektování elektrických zařízení
- povrchové úpravy a svařování kovů
- činnost podnikatelských, finančních, organizačních a ekonomických poradců
- realitní činnost
- výzkum a vývoj v oblasti přírodních a technických věd nebo společenských věd

[1] (18)

## 2.2 Informační systém společnosti

Společnost Siemens Electric Machines s.r.o. používá moderní špičkové informační technologie. V konstrukci a vývoji se pracuje s nejnovějšími kreslícími a modelovacími programy, z toho plyne, že počítače musí splňovat jejich požadavky.

V podniku je velká část procesů řízena pomocí počítačů. Počítače se zde využívají v personalistice, zakázkách, konstrukci, plánování práce, skladování ...

### 2.2.1 Hardware a Software

Protože společnost využívá náročné operační systémy a programy, musí její počítače být kvalitní. Výpočetní technika společnosti se dá rozdělit do více úrovní kvality. Nejkvalitnější vybavení je třeba samozřejmě pro servery a pracovníky IT oddělení. Výkonné počítače jsou také v konstrukci a vývoji. Pro tato oddělení je třeba vysoký výkon výpočetní techniky kvůli náročným aplikacím. Většina jich je už vybavena dvoujádrovými procesory a kvalitní grafickou kartou s výstupem pro dva monitory. Právě zobrazení aplikací na dvou monitorech je velmi efektivní. Pokud se totiž pracovníkovi například načítá model stroje, může mezitím pracovat na něčem jiném. Dva monitory ocení také pracovníci při potřebě zobrazit více dokumentů najednou.

Na druhé straně v oblasti personalistiky, nákupu a řízení výroby již není kladen takový nárok na výkon, a pak plně dostačují průměrné počítače.

Software společnosti se musí udržovat, jak jsem již zmínil, aktuální. Neustále se aktualizuje a kupují se nové verze. V naprosté většině počítačů je operační systém Microsoft Windows XP a kancelářský balík Microsoft Office 2003. V konstrukci a vývoji se používají grafické a modelovací programy a speciální systémy pro rozdělování práce a týmovou spolupráci. Po celé společnost je zaveden automatizovaný informační systém, kde se evidují zásoby a všechny ostatní informace potřebné pro výrobu. Díky němu jsou navzájem propojena všechna oddělení – konstruktéři mohou například vyhledat ve skladu vhodný materiál pro stroj.

Do specifických programů se musí uživatelé přihlašovat. Tyto programy mají totiž většinou omezenou licenci na počet počítačů a je také nežádoucí, aby do nich měl přístup každý.

### 2.2.2 Počítačová síť

Počítačová síť je rozšířena do každého počítače společnosti. Tyto jsou pak pomocí ní mezi sebou vzájemně propojeny a připojeny na intranet a internet pomocí serverů. Všechna důležitá a společná data jsou uložena na serverech v serverovně. Každému uživateli jsou přidělena práva k přístupu na určité disky a složky na serverech. Administrátoři mají pod kontrolou celou síť a pomocí speciálního softwaru se mohou přepnout na jednotlivé počítače a administrovat je na dálku. Síť je vedena prozatím strukturovanou kabeláží, bezdrátové připojení WiFi je zavedeno jen v jednacích místnostech. Síť je zabezpečována servery z jedné centrální serverovny.

Díky speciálnímu globálnímu softwaru má podnik přístup i k některým datům ostatních podniků v rámci koncernu. Nejvíce spolupracuje s podnikem v Berlíně, pro který vyrábí. Tohoto globálního přístupu se využívá nejvíce při sdílení dokumentace, inovací a technických a technologických pokroků. Sdílené jsou převážně výkresy a modely částí strojů. Součástí systému je i globální seznam e-mailových adres pro celý koncern. Tato výhoda však má i svoji zápornou stránku, a to složitost některých operací a automaticky generované e-maily. Tyto e-maily se generují automaticky systémem při každé změně či úpravě v dokumentaci, a navíc jsou v anglické nebo německé jazyce.

### 2.2.3 Zpracování a zálohování dat

Většina dat společnosti se zpracovává na serverech, kam mají uživatelé společný přístup. S těmito daty může pracovat několik uživatelů najednou, což ovšem zpomaluje výkon. Jedná se především o data automatizovaného informačního systému firmy a společné modely strojů a výkresy. Proto si uživatelé některá data nahrávají na lokální disky a pracují s nimi tam,což jim umožňuje vyšší výkon i stabilitu systému.

Zálohování se týká veškerých serverů společnosti a částí lokálních stanic, obsah ostatních lokálních disků počítače si zálohují uživatelé sami. Firemní e-mailové schránky jsou zálohovány na server.

### 2.2.4 Informační systém firmy

Informační systém společnosti je aplikace typu klient/server sloužící ke komplexnímu řízení podniku, tzv. Enterprise resources planning (ERP). Jedná se o informační systém, který integruje a automatizuje velké množství procesů souvisejících s produkčními činnostmi podniku. Jedná se především o integraci výroby, materiálů, distribuce a uskladnění, logistiky a jiných částí podniku.

Tok informací uvnitř společnosti Siemens Electric Machines s.r.o. je řešen klasickým způsobem. Všechny důležité normy a pokyny mohou pracovníci nalézt na intranetu, stejně jako kontaktní informace, vzorové formuláře, atd. Sdílená a důležitá data se nacházejí na serverech, odkud k nim mají přístup uživatelé a systémy. Tyto servery mají také globální význam, jak jsem již zmínil, společnost Siemens v Drásově pracuje například s daty společnosti Siemens v Berlíně a naopak.

## *2.3 SWOT analýza*

V následující kapitole chci popsat podnik z ekonomického hlediska. Pomocí nástrojů interní a externí analýzy se pokusím odhalit silné a slabé stránky společnosti a její příležitosti a hrozby.

### 2.3.1 Interní analýza – 7S

Interní analýzu jsem vypracoval na základě analýzy 7S.

**Strategie**

Společnost si zakládá na tom, že vyrábí produkty na zakázku a že její hlavní zákazník je koncern. Právě od těchto dvou bodů se vyvíjí strategie společnosti. Výrobní a obchodní strategie je koordinována v rámci koncernu Siemens AG– divize A&D LD se sídlem v Norimberku.

**Organizační struktura**

Společnost Siemens je obrovský světový koncern, působí v 190 zemích celého světa. Mateřská společnost Siemens AG (akciová společnost) vlastí většinu firem koncernu a řídí je. Siemens AG je rozdělen do 3 divizí podle oboru : Automation and Drives (automatizace a pohony), Power (zdroje) a Medical (zdravotnictví).

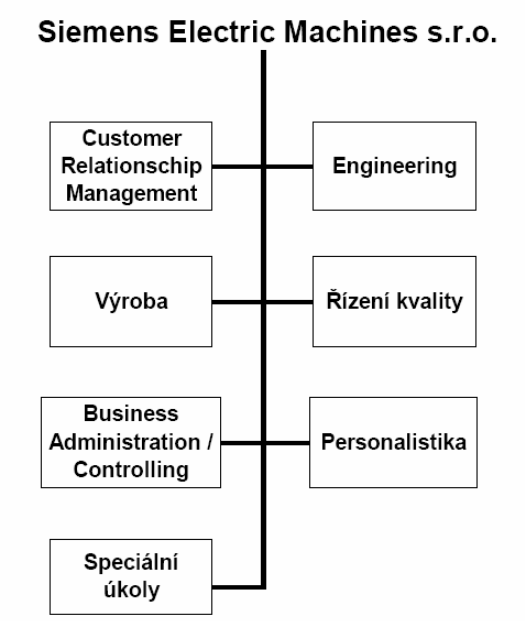
Společnost Siemens Electric Machines s.r.o. patří do následující struktury společnosti SIEMENS:

Siemens Electric Machines s.r.o. (I DT LD SEM)

- Industry Sector (I)
- Automation and Drives (A&D)
- Division: Drive Technologies (DT)
- Bussines Unit: Large Drives (LD)

Společnost provozuje jeden výrobní závod v Drásově. Závod zabezpečuje samostatné výrobní, obchodní a správní funkce.

Organizační schéma společnosti je znázorněno na obrázku 2.




*Obr. 2 : Organizační schéma*

**Informační systémy**

Informační systém jsem již popsal v kapitole 2.2

**Styl řízení**

O stylu řízení jsem se již krátce zmínil v předchozím textu. Společnost je zčásti řízena samostatně vedoucími pracovníky, ale nejvyšší řízení a volby strategií upravuje koncern Siemens, převážně mateřská společnost Siemens AG a potom divize A&D LD.

**Sdílené hodnoty**

Společnost aktivně a neustále přispívá k trvale udržitelnému rozvoji využívání technologií šetrných k životnímu prostředí. Společnost zavedla systém environmentálního managementu podle ČSN EN ISO 14001 a v prosinci 2006 získala certifikát.

Protiprávní a neetické jednání společnost Siemens netoleruje. Vytvořila závazné firemní směrnice, v jejichž rámci musí všichni vedoucí pracovníci a zaměstnanci na celém světě jednat eticky a v souladu se zákonem. Tyto směrnice tvoří základ práce společnosti a vzájemných vztahů nejen mezi zaměstnanci, ale i směrem k zákazníkům a partnerům.

**Spolupracovníci**

Mezi spolupracovníky společnosti panují převážně dobré vztahy. Přispívají k tomu zábavné a sportovní aktivity a rekreační středisko, které je pracovníkům k dispozici. Právě na různých sportovních soutěžích společnosti se dosáhne kolektivity pracovníků. Na pracovišti panuje většinou dobrá nálada.

**Schopnosti**

Schopnosti společnosti jsou dobré, neboť je již dobře zaběhnutá ve výrobě a díky moderním strojům a školenému personálu je schopna dodávat produkty na zakázku. Tímto se odlišuje od konkurence.

### 2.3.2 Externí analýza - Porterův model 5-ti konkurenčních sil

**Riziko vstupu potencionálních konkurentů**

Riziko vstupu potencionálních konkurentů není veliké. Hlavně proto, že společnost je již dobře zaběhnutá a že její odběratel je z 80% koncern. V tomto má společnost velikou výhodu, protože bude vždy mít odběratele pro své výrobky. A zbylých 20% odbytiště není významnou částí.

Pokud by nějaký konkurent chtěl vstoupit na trh, potřeboval by velké finanční prostředky na vybudování budov a strojů nutných pro výrobu.

**Rivalita mezi stávajícími konkurenty**

Rivalita mezi stávajícími konkurenty není nikterak vysoká. Společnost své produkty především vyváží, a tak konkurence v ČR pro ni nemá zásadní význam. A s konkurencí koncernu se vypořádává mateřská společnost.

**Smluvní síla odběratelů**

Společnost má sice více odběratelů, ale z 80% je to koncern Siemens. Ovšem tento koncern na společnost nevyvíjí žádný nátlak, protože je zároveň jejím 100% vlastníkem.

**Smluvní síla dodavatelů**

Podnik má více dodavatelů, takže na něho nemohou klást nátlak. Hlavním dodavatelem polotovarů je opět koncern Siemens, ale materiál a například dopravní služby zajišťuje pro společnost více různých dodavatelů.

**Hrozba substitutů**

Hrozba substitutů pro společnost existuje, ale jen do jisté míry. Konkurence sice může vyrábět podobné produkty jako společnost, ale většinou není schopna vyrábět tyto produkty přesně na zakázku, dle přání zákazníků. V tom má společnost výhodu, protože v této oblasti průmyslu je třeba vyráběné produkty upravovat podle přání zákazníků.

### 2.3.3 Výsledky SWOT analýzy

#### 2.3.3.1 Silné stránky

Asi nejsilnější stránkou firmy je to, že je součástí obrovského světového koncernu a že mateřská firma koncernu sídlí v sousedním státě, protože převážnou část odbytu společnosti tvoří výrobky a komponenty pro mateřskou firmu Siemens AG. S mateřskou společností pak společnost úzce spolupracuje a má také přístup k některým jejím materiálům. Toho se využívá převážně při používání výkresů, modelů a postupů výroby mateřské firmy. Společnost v Drásově také spolupracuje se společnostmi Siemens v Norimberku, Berlíně, Mnichově a ostatními společnostmi v Evropě i mimo ni.

Další silnou stránkou je také vytváření produktů výhradně na zakázku. Tím vzniká ve společnosti vysoký podíl know-how. Protože se stroje zpravidla neopakují,

musí pracovníci vyvíjet nové modely a nové postupy výroby, a tím se neustále zdokonalovat. Výroba na zakázku je obrovskou výhodou oproti pásovým výrobám, kde může klesnout zájem o produkt a společnost pak má problémy s odbytem.

Obrovskou výhodou je pro společnost možnost půjčení kapitálu od Siemens financial services. Siemens financial services je instituce, sídlící v Německu a poskytující pro celý koncern Siemens potřebný kapitál. Podnik si může od této instituce půjčit prakticky tolik, kolik bude potřebovat, a nemusí platit žádné úroky ani penále.

#### 2.3.3.2 Slabé stránky

Slabou stránkou společnosti je v současné době velký podíl nekvalifikovaných pracovníků. S prudkým růstem potřebovala společnost velký počet manuálních pracovníků a nestihla je zaškolovat do problematiky. Zaškolování během výroby tak způsobí problémy ve výrobních procesech.

Další slabou stránkou je také minimální opakovatelnost výroby, jež může přinášet určité výrobní problémy, neznalost výroby, náklady na školení pracovníků, potenciální problémy s materiálem atd.

#### 2.3.3.3 Příležitosti

Hlavní příležitostí společnosti je co nejvíce spolupracovat s koncernem Siemens, a to převážně s kolegy v Německu, kde mají vyspělejší programové vybavení, jsou dále ve výzkumu a vývoji a používají dokonalejší systémy komunikace a výroby. Pro oddělení konstrukce a vývoje je největší příležitostí používat tyto kvalitnější a efektivnější informační systémy a aplikace.

Další velkou příležitostí je v poslední době lepší opakovatelnost produktu, kdy ačkoliv je výrobek dělaný na zakázku, je shodný nebo velmi podobný těm, které společnost v minulosti již vyrobila.

#### 2.3.3.4 Hrozby

V současné době je pro společnost největší hrozbou nedostatek zakázek, který je způsobený probíhající celosvětovou ekonomickou krizí. Společnost musí neustále hledat nové zakázky, protože vyrábí produkty podle přání zákazníka. Tato hrozba souvisí s celým koncernem Siemens, protože právě on je největším odběratelem.

Další hrozbou je možnost vzniku nedodržování termínů dodání a s tím spojené náklady. Kvůli malé opakovatelnosti produktu by mohly vznikat konstrukční a výrobní problémy, díky tomu se výroba produktu může protahovat a tím budou vznikat nepředvídané náklady a snižovat se zisk z produktu.

## 2.4 Současný stav konstrukce svazků stroje

Poslední kapitolou analýzy současného stavu je aktuální způsob konstrukce svazků stroje. Chtěl bych popsat, jak se svazek konstruuje a jak by bylo možné tento proces zkvalitnit a urychlit.

Jak jsem již uvedl v předchozích kapitolách, podnik vyrábí produkty převážně na zakázku, a tudíž je malá opakovatelnost produktu. Ovšem pokud se zaměříme jen na části stroje, opakovatelnost je větší. Například svazek stroje je možno zkonstruovat pomocí obecného modelu.

Pokud chce konstruktér vytvořit nový svazek, nejprve se pokusí najít svazek podobný a předělat jej. Ovšem toto předělání znamená ruční přepisování hodnot ve starém modelu, což je značně nepraktické. Pokud se jedná o zcela nový svazek, který se nepodobá žádnému předchozímu, otevře si konstruktér obecnou skicu (model) svazku, a doplňuje do ní rozměry částí a model upravuje. Je vidět, že konstrukce svazku probíhá neautomatizovaně a že je závislá na znalostech konstruktéra o předchozích svazcích.

Žádný ucelený přehled vytvořených svazků stroje v elektronické podobě v podniku neexistuje. Konstruktér má k dispozici pouze výkresy starých svazků.

## 2.5 Shrnutí současného stavu

Jak je vidět z předchozích kapitol, společnost Siemens Electric Machines s.r.o. je dobře zaběhnutou společností s plně funkčním informačním systémem. Společnost používá moderní informační techniku a software a je propojena lokální a globální sítí.

V kapitole 2.4. jsem se zaměřil na analýzu současného stavu konstrukce svazků stroje a odhalil jsem nedostatky. Pro urychlení a zkvalitnění procesu konstrukce by bylo vhodné vytvořit databázi svazků stroje, která by nesloužila pouze k prohlížení a vyhledávání, ale která by umožnila automatizovanou konstrukci nového svazku. A tím se budu také ve své práci zabývat.

# 3 Teoretická východiska práce

## *3.1 Databáze*

Pojem databáze se dnes používá v mnoha slovních spojeních a v mnoha vědních oborech. Existuje mnoho definicí databáze. Andrew Oppel definuje databázi takto: „Databáze je kolekce vzájemně souvisejících dat, s nimiž pracujeme jako s ucelenou jednotkou.“[2] Obecně se dá říci, že databáze je propracovaný a ucelený systém pro ukládání dat a jejich zpravování a používání. V širším pojetí spadají do pojmu databáze i nástroje, které s daty pracují (ukládají je, mění je, mažou je).

Michael J. Hernandez upřesňuje: „Vůbec nezáleží na tom, jestli ke shromažďování a ukládání dat používáte papír, nebo počítačový program. Dokud shromažďujete data nějakým organizovaným způsobem, máte databázi.“[3] Z toho plyne, že databází se rozumí jak papírové kartotéky a archívy, tak elektronické databáze.

Pro pochopení pojmu databáze uvedu následující příklad. „Databázi si můžete nejsnáze představit jako papírovou kartotéku, známou z lékařských ordinací. Databáze obsahuje data (informace o pacientech) uložená na paměťovém médiu (papíry). Tato data mezi sebou mají určité vztahy (např. u jednoho pacienta chronologicky navazují) a jsou určitým způsobem členěna (desky jednotlivých pacientů, šuplíky podle příjmení) ... nástrojem pracujícím s daty je v našem případě sestřička.“[4]

### 3.1.1 Databázový systém

Databází myslíme pouze určitá „skladiště“ dat, ve kterých jsou data uložena a zpracovávána nezávisle na aplikačních programech. Databáze je tedy v technickém slova smyslu báze dat.

Pro samotný přístup k datům uloženým v databázi se používá speciální software. Nazývá se Database Management Systems (DBMS), česky se překládá jako Systém řízení báze dat (SŘBD). SŘBD zajišťuje práci s databází, neboť tvoří rozhraní mezi aplikačními programy a uloženými daty. Fyzická struktura uložených údajů potom nemusí být aplikačnímu programu, a tím pádem ani uživateli, vůbec známa.

---

[2] (8, s. 17)
[3] (3, s. 42)
[4] (15)

Databázový systém zajišťuje všechny základní služby, které jsou nezbytné pro organizaci databáze a její funkční chod, jako například:

- Přesouvání dat do fyzických datových souborů a naopak.
- Správa současného přístupu více uživatelů k datům.
- Podpora dotazovacího jazyka, který je tvořen množinou příkazů pro načítání dat z databáze.
- Mechanismy pro zálohování databáze a pro její zotavení po haváriích.
- Bezpečnostní mechanismy, které zabraňují v neoprávněném přístupu a v neoprávněných modifikacích.[5]

Příkladem databázových systémů jsou například softwarové produkty: Microsoft Access, Microsoft SQL Server, Oracle, Sybase, MySQL a další.

### 3.1.2 Databázový model

Databázový model reprezentuje strukturu a funkcionalitu databáze. Umožňuje definovat schéma databáze, určující organizaci dat, způsoby jejich ochrany a zajištění správnosti (integritní omezení) a přípustné operace s daty. „Databázový model je v podstatě architektura, podle které databázový systém ukládá objekty do databáze a podle které je vzájemně provazuje.“[6]

Podle typu modelovaných vztahů mezi záznamy v databázi se rozlišuje:

- lineární datový model
- hierarchický datový model
- síťový datový model
- relační datový model
- objektový datový model
- objektově-relační (také postrelační) datový model

[5] (8 ,s. 18)
[6] (8, s. 22)

Ve své práci budu popisovat pouze lineární a relační datový model, kterých se moje práce bude týkat.

**Lineární datový model**

Lineární datový model je prakticky souborová databáze, kde jsou data uložena jako samostatné soubory. Mezi jednotlivými datovými tabulkami (soubory) není žádná vazba. „Příkladem lineárního modelu je například kartotéka pacientů, kde jednotlivé karty s údaji o pacientech jsou seřazeny v krabici. Každá karta představuje jednu větu databázového souboru. Mezi kartami (větami) není žádný vzájemný vztah s výjimkou vztahu předchůdce a následovníka.“[7]

**Relační datový model**

Relační datový model je v současné době nejpoužívanější a nejrozšířenější pro svoji jednoduchost, snadnou pochopitelnost a univerzálnost. Na rozdíl od hierarchického a síťového modelu jsou relační databáze nezávislé na aplikaci a jsou založeny na relačním modelu dat a relační algebře.

„Vzniká z několika lineárních modelů, spojených dohromady pomocí položky (položek), kterým říkáme relační klíč. Toto spojení není trvalé, jako u předchozích modelů, ale vzniká v okamžiku, když potřebujeme mít společně k dispozici data ze všech spojených tabulek, a zaniká, když práci s modelem ukončíme. Jednotlivé lineární modely lze pochopitelně využívat i samostatně.“[8] „V relačním modelu máme proto možnost svázat záznamy jen podle potřeby, nikoli podle vazeb definovaných předem při prvotním ukládání záznamů do databáze.“[9]

Struktura se skládá z relací (dvourozměrných tabulek) a vztahů vytvořených nad nimi. Data jsou uložena v řádcích a sloupcích tabulek. Řádky se v terminologii z pohledu teorie relací nazývají n-tice relace a sloupce atributy relace. Atributy mají určen svůj konkrétní datový typ. Jednotlivé n-tice můžeme jednoznačně identifikovat pomocí primární klíče. Nad těmito tabulkami (relacemi) je pak možné definovat přípustné operace, tedy složité vztahy, pomocí cizích klíčů. Dotazovacím jazykem relačního modelu je jazyk SQL.

[7] (4, s. 20-21)
[8] (4, s. 22-23)
[9] (8, s. 28)

## *3.2 Návrh relační databáze*

Relační modely jsou vhodné pro řízení velkého množství dat a vyhledávání dat. Tyto modely jsou založeny na dvourozměrných tabulkách a vztahy mezi daty jsou vyjadřovány porovnáváním hodnot. „Relační datové modely nám umožňují zachytit v modelu nejenom data o zkoumaných objektech, ale také vzájemné vztahy těchto objektů, což nám umožňuje přiblížit se více reálnému světu.“[10] (4, s. 24)

Otec relačního modelu, Dr. E. F. Cood založil relační datový model na dvou matematických disciplinách – teorii množin a predikátové logice prvního řádu. Součástí teorie množin je pojem „relace“, ze kterého se odvodil název modelu. Díky tomu má relační model silný teoretický základ ve dvou matematických disciplinách, a je tedy jasná definice tohoto modelu.[11]

### 3.2.1 Pravidla pro tabulkovou prezentaci relace

Aby se relace mohla prezentovat pomocí dvourozměrných tabulek, musí být dodržena následující pravidla :

- každý řádek odpovídá jedné n-tici relace
- tabulka nesmí obsahovat duplicitní řádky
- pořadí řádků i sloupců je nevýznamné
- význam každého sloupce určuje jméno atributu
- žádné dva názvy sloupců (atributy) nejsou stejné
- hodnoty ve sloupcích jsou atomické[12]

### 3.2.2 Integrita dat

Integrita modelu obecně je takový stav, při kterém uložená data odpovídají vlastnostem objektů reálného světa. Tohoto stavu dosáhneme pomocí integritních omezení, která můžeme rozdělit na integritní omezení pro entity (relace) a integritní omezení pro vztahy entit (relační vazby).[13]

[10] (4, s. 24)
[11] (8, s. 46 - 51)
[12] (4, s. 27)
[13] (4, s. 28)

#### 3.2.2.1 Integritní omezení pro entity

**a.) Doménová integrita (integrita hodnot)**

Doména je pojmenovaná množina hodnot stejného typu, například doména křestních jmen. Doménová integrita znamená, že každá hodnota každého atributu relace musí být z množiny hodnot (domény) pro daný atribut přípustných. Z toho plyne, že doména musí být definovaná jako množina hodnot. Přípustné hodnoty pro daný atribut lze specifikovat pomocí datového typu pole, rozsahu hodnot, masky pro vkládání, neprázdné hodnoty a podobně.

**b.) Entitní integrita**

Entitní integrita stanoví, že každá relace musí mít určený primární klíč, který se skládá z jednoho nebo více atributů a jednoznačně identifikuje každý řádek relace. Primární klíč je jednoznačný a minimální, to znamená, že v relaci nesmí existovat dvě n-tice se stejným primárním klíčem a že žádný atribut primárního klíče nelze vypustit. Každý atribut primárního klíče musí být vyplněn a každá n-tice relace musí být vždy identifikovatelná pomocí hodnoty primárního klíče.

**c.) Referenční integrita**

Referenční integrita popisuje cizí klíč a propojení relací. Pro cizí klíč platí, že každá hodnota musí být plně zadaná nebo plně nezadaná, a že každá hodnota cizího klíče se shoduje s hodnotou primárního klíče některé n-tice jiné relace. Dále platí, že cizí klíč a odpovídající primární klíč musí být definovány na stejné doméně a že v databázi nesmí být žádná nesouhlasná hodnota cizího klíče.[14]

Primární klíč a cizí klíč tedy umožňují vytvářet spojení mezi relacemi, což je hlavním účelem relačního datového modelu.

#### 3.2.2.2 Integritní omezení pro vztahy entit

„Integritní omezení pro vztahy omezuje kardinalitu vztahu na poměry 1 : 1, 1 : N, N : 1, N : M. Tento poměr uvádí, kolik n-tic relací sobě navzájem odpovídá.“[15]

[14] (4, s. 28 -31)
[15] (4, s. 31)

**a.) vztah 1 : 1**

Jedna n-tice první relace (tabulky) odpovídá jedné nebo žádné n-tici druhé relace, např. člověk může vlastnit jen jeden nebo žádný řidičský průkaz.

**b.) vztah 1 : N**

Jedna n-tice první relace odpovídá jedné nebo více n-ticím z druhé relace, např. student může skládat více zkoušek, ale jedna konkrétní zkouška přísluší jednomu studentovi.

**c.) vztah N : 1**

Jedná se o stejný vztah jako 1 : N, pouze ho bereme z opačné strany.

**c.) vztah N : M**

Tento vztah vyjadřuje, že jedna nebo více n-tic první relace odpovídá jedné nebo více n-ticím druhé relace. Vztah N : M musíme řešit dekompozicí relace, tj. přidáním další tabulky, kde bude složený primární klíč. Příkladem může být společník, který vlastní více společností a společnost může vlastnit více společníků.

### 3.2.3 Normalizace dat

Normalizace je proces, ve kterém organizujeme databázi do struktury, která zachovává integritu uložených dat a minimalizuje nadbytečná data. Normalizace spočívá v odstranění redundantních dat, omezení složitosti (rozložení složité relace na dvojrozměrné tabulky) a zabránění aktualizačním anomáliím (např. abychom smazáním všech knih autora nepřišli o data o autorovi), což by mělo vést k databázi přehlednější, rozšiřitelnější a výkonnější. Normalizace by měla vést k vzniku tabulek, které lze snadno udržovat a efektivně se na ně dotazovat. Normalizované schéma musí zachovat všechny závislosti původního schémat a relace musí zachovat původní data, což znamená, že se musíme pomocí přirozeného spojení dostat k původním datům.[16]

Aby byla databáze normalizovaná, musí splňovat alespoň tyto tři normální formy:

[16] (4, s. 55-56)

**1. normální forma – multizávislost**

1. normální forma říká, že všechny atributy entity musí být jednoduché. Atributy nesmí být složené ani vícehodnotové.

**2. normální forma – funkční závislost**

Aby byla relace v druhé normální formě, musí být i v první normální formě, a navíc všechny její atributy jsou závislé na celém primárním klíči.[17]

**3. normální forma – tranzitivní závislost**

„Relace je ve třetí normální formě, pokud je v druhé normální formě a navíc všechny její neklíčové atributy jsou vzájemně nezávislé.“[18]

### 3.2.4 Postup návrhu modelu

Návrh relačního datového modulu se dá shrnout do 6-ti základních kroků :

1.) definice jednotlivých datových objektů , entit

2.) zobrazení objektů do diagramu a stanovení vztahů mezi entitami (relační poměr)

3.) dekompozice vazeb N:M na vazby 1:N nebo N:1

4.) stanovení primárních klíčů pro jednotlivé entity

5.) určení, jaké klíče budou použity pro jakou vazbu. Pro takové určení platí pravidlo: „Pro vazbu 1:N musí být jako klíč použit vždy primární klíč ze strany 1 jako cizí klič na straně N. Primární klíč ze strany N nemůže být použit jako cizí klíč na straně 1.“[19]

6.) stanovení sledovaných atributů entity.

Pomocí těchto 6-ti kroků jsme schopni navrhnout relační model.

[17] (4, s. 56-58)
[18] (4, s. 60)
[19] (4, s. 54)

## *3.3 Microsoft Office Access*

Microsoft Office Access (dále jen Access) je software speciálně navržený pro tvorbu databází a práci s nimi. Access je součástí balíčku Microsoft Office, takže je dostupný pro širokou veřejnost. Access poskytuje uživateli všechny funkce potřebné k navržení relační databáze a práci v takové databázi. Databáze programu Access mají příponu .mdb a jedná se o samostatné databázové soubory. Data v těchto souborech jsou obsažena v datových tabulkách. Tyto soubory mohou být používány jako samostatná databáze na jednom počítači, nebo je lze nahrát na server a nastavit multiuživatelský přístup.

V dalším popisu programu se budu zabývat verzí Microsoft Office Access 2003.

### 3.3.1 Databázové objekty Accessu

V následující kapitole se zaměřím na základní databázové objekty používané v programu Access 2003.

#### 3.3.1.1 Tabulky

Tabulky jsou základním stavebním prvkem Accessu a tvoří soubor dat databáze. Tabulka představuje soubor polí (sloupců) a záznamů (řádků). Pole obsahují informace jednoho typu (např. příjmení zákazníka) a do záznamů se ukládají všechny informace o konkrétním subjektu.[20]

V každé tabulce máme pro každé pole k dispozici název pole, datový typ pole a popis pole. Název pole může obsahovat mezery i speciální znaky, ale je lepší tyto nepoužívat, pokud chceme s polem pracovat například v programovém kódu. Popis pole je podrobnějším vysvětlením pole pro uživatele, které se zobrazí ve stavovém řádku při prohlížení databáze. Datovému typu se budu věnovat později.

Access nám samozřejmě umožňuje nadefinovat i primární klíče tabulek, a ty mohou být jednoduché i složené. U primárního klíče si můžeme zvolit, že nechceme povolit duplicitu, což je jedna z podmínek pro primární klíč.

[20] (8, s. 12)

#### 3.3.1.2 Formuláře

Formuláře tvoří příjemné rozhraní mezi uživatelem a daty v tabulce. Formuláře jsou tvořeny pro usnadnění zadávání a prohlížení dat v databázi. „Místo často komplikovaného zadávání dat přímo do tabulek lze využít přehledný formulář funkčně i graficky přizpůsobený vašim potřebám…“[21]

Access nabízí i řadu předdefinovaných formulářů a to z hlediska funkční a hlavně vizuální stránky. Při vytváření takového formuláře pomocí průvodce pak stačí vybrat jen potřebné ovládací prvky a vzhled a formulář dotvoří sám program.

Do formuláře lze přidávat i vlastní ovládací prvky, jako například tlačítka, přepínače, rozvírací seznamy, obrázky, popisky a jiné. Na jednotlivé ovládací prvky formuláře potom můžeme vytvořit procedury v jazyce Visual Basic for Application. O této možnosti se zmíním později.

#### 3.3.1.3 Dotazy

Dotazy jsou nezbytnou součástí všech databázových programů. Dotazy slouží k vyhledávání v datových tabulkách podle zadaných kritérií a k upravování a filtrování výsledků těchto vyhledávání. Dá se říct, že v dotazech je největší výhodnost databázových programů.

„Pomocí dotazů lze např. shromáždit data z několika tabulek a seřadit je podle zadaného klíče, provádět výpočty ve skupinách záznamů, aktualizovat data, nebo zadávat nové databázové objekty atd.“[22]

Access má k dispozici několik typů dotazů: jsou to výběrový, aktualizační, přidávací, odstraňovací, vytvářecí a křížový. U každého typu pak lze vybrat, jaká pole a z jakých tabulek chceme zobrazit, podle čeho chceme výsledek třídit a jaká kritéria chceme do dotazu zahrnout.

#### 3.3.1.4 Sestavy

Výše zmíněné formuláře a dotazy slouží spíše k zobrazení dat pro uživatele, ale příliš se nehodí k uložení do souboru nebo tisku. Proto je v Accessu dostupný ještě jeden prvek určený k prohlížení dat, a to sestava. Sestava sumarizuje a prezentuje data

[21] (8, s. 13)
[22] (8, s. 12)

v tištěné formě. K dispozici je několik předdefinovaných typů, které si může uživatel libovolně upravovat. Do sestavy lze přidat i graf, obrázek, hlavičku a podobně. Sestava tedy slouží jako fyzický výstup z databáze, který se uplatní při prezentaci dat, nebo při papírové archivaci.

#### 3.3.1.5 Makra

Makra se využívají pro automatizaci často se opakujících úkolů. Jedná se prakticky o záznam činnosti provedených uživatelem od spuštění zaznamenávaní makra po jeho ukončení. K vytváření maker stačí pouze pomocí myši spouštět vybrané funkce Accessu, takže uživatel nemusí mít žádné znalosti programování.

#### 3.3.1.6 Moduly

Moduly jsou oproti makrům výrazně složitější. Pro jejich vytváření je již potřeba znalost programování. Moduly jsou naprogramované příkazy a procedury v jazyku Visual Basic for Applications. Moduly jsou samostatné jednotky vytvořené v tomto jazyce a pro jejich spuštění je třeba do programového kódu, například tlačítka ve formuláři, doplnit odkaz na modul.

### 3.3.2 Datové typy Accessu

Datový typ pole tabulky by měl odpovídat druhu dat, která jsou v poli uložena. Pokud je v poli například uloženo příjmení, je vhodný datový typ text, pokud je v poli výsledek sčítání, použije se datový typ číslo a podobně. Datový typ také určuje, jaké hodnoty může uživatel do pole zadat. Pokud je například vybrán datový typ číslo, pak uživatel nemůže do pole zadávat písmena nebo znaky.

Access má na výběr celkem 9 datových typů. Podle zvoleného datového typu nám potom nabízí ještě upřesňující nastavení pro datový typ. Nyní jednotlivé datové typy stručně popíši.

**Text**

Datový typ Text je základním datovým typem. Slouží pro ukládání kombinace libovolných textových znaků (včetně číslic) do pole tabulky. Tento datový typ nabízí další upřesnění pole, nejdůležitější jsou pro nás Velikost pole a Titulek. Velikost pole je maximální počet znaků, který může pole obsahovat. Implicitně je tato velikost 50 znaků, maximální možná je 255 znaků. Do pole Titulek se zadávají upřesňující údaje

o poli, které se zobrazí uživateli na stavovém řádku. Zajímavou vlastností je možnost Zobrazit ovládací prvek na kartě Vyhledávání. Pokud zde vybereme Seznam nebo Pole se seznamem, pak se nám při vyplňování tabulky v poli zobrazí rozbalovací seznam s předdefinovanými hodnotami.

**Memo**

Datový typ Memo má stejné vlastnosti jako datový typ Text. Jediný rozdíl je v tom, že Memo má maximální velikost pole až 65 535 znaků a nelze indexovat.

**Číslo**

Datový typ Číslo se v praxi používá jen pro čísla, která se účastní matematických operací. Například pro rodné číslo, PSČ, číslo popisné aj. se používá datový typ Text. Pro datový typ Číslo máme k dispozici různé „druhy čísel“. Tyto druhy vybíráme v nabídce Velikost pole.

Druhy čísel (v závorce je upřesněn rozsah) :

„1. bajt (velikost 1 bajt, tj. 0 až 255),

2. celé číslo (2bajty, tj. -32 768 až 32 767),

3. dlouhé celé číslo (4 bajty, tj. – 2 147 483 648 až 2 147 483 647),

4. jednoduchá přesnost (reálné číslo na 7 desetinných míst),

5. dvojitá přesnost (reálné číslo na 15 desetinných míst),

6. replikační identifikátor (16bajtové pole použité pro vytvoření jednoznačného identifikátoru pro replikaci)“[23]

K dalším důležitým vlastnostem patří především Počet desetinných míst, kde můžeme nastavit pevný počet míst, a Formát. Vlastnost Formát nastavuje zobrazení pole. Uživatel si může vybrat, jestli chce pole zobrazit jako standardní číslo, číslo s pevným počtem desetinných míst, měnu, procenta nebo vědecký formát.

[23] (8, s. 45)

**Datum/čas**

Datový typ Datum/čas se používá pouze pro ukládání dat a časových hodnot. Access dává uživateli na výběr z předdefinovaných formátů, nebo možnost vytvořit si vlastní formát.

**Měna**

Datový typ Měna je obdobou číselného datového typu. Tento typ je vhodné použít, pokud se do pole budou ukládat peněžní částky. Pole tohoto datového typu mohou vstupovat do matematických operací. Přesnost typu Měna je 15 číslic nalevo a 4 čísla napravo od desetinné čárky.

**Automatické číslo**

Automatické číslo se používá výhradně jako jednoznačný identifikátor záznamu v tabulce. Při přidání nového záznamu do tabulky se toto pole vždy změní podle vlastnosti Nové hodnoty. Vlastnost Nové h odnoty lze nastavit jako: Náhodný (číslo se generuje náhodně) nebo Přírůstek (pokaždé se zvýší o 1).

**Ano/ne**

Tento datový typ ukládá do pole logickou hodnotu pravda nebo nepravda. V tabulce je tento prvek zobrazen jako zaškrtávací pole.

**Objekt OLE**

„OLE (Object Linking and Embedding) je technologie, která se používá ke sdílení souborů v různých aplikacích sady Office.“[24] Data pole typu Objekt OLE jsou například tabulka Excelu, dokument Wordu, obrázky, zvuky aj. Maximální velikost pole je 1GB.

**Hypertextový odkaz**

Datový typ hypertextový odkaz je uložený jako text a používán jako adresa odkazu.

[24] (21)

## 3.4 Jazyk Visual Basic for Applications (VBA)

Visual Basic for Applications (dále jen VBA) je plnohodnotný programovací jazyk společnosti Microsoft. VBA vychází z jazyka Visual Basic, ovšem jeho funkce jsou omezené pouze na práci s produkty Microsoft Office. Od klasického programovacího jazyka se liší v tom, že objekty a kolekce, které používá, jsou přizpůsobeny objektům řady Microsoft Office. VBA tedy slouží pro tvorbu převážně kancelářských aplikací na platformě Microsoft Office. V Accessu se dá VBA využít hlavně při programování ovládacích prvků na uživatelském formuláři.

Programování v tomto jazyce patří do objektově orientovaných a událostmi řízených technik. K dispozici je velké množství předdefinovaných objektů, které jsou souhrnně nazvány ovládací prvky (controls). Každý ovládací prvek má definovány své vlastnosti, metody a události.

Vlastnosti prvku (properties) udávají vzhled, umístění a chování prvku v aplikaci. Metody představují činnosti, které daný ovládací prvek může vykonávat, případně které mohou být vykonávány na něm. Příkladem metody je například Refresh (obnovení zobrazovaných dat) nebo SetFocus (zaměření – vybrání prvku). Každý prvek má seznam událostí, které mohou při běhu naprogramované aplikace vzniknout v přímém vztahu k tomuto prvku. Události slouží k programování procedur, které jsou vyvolány jako odezva na výskyt určité činnosti. Příkladem události je Click (klepnutí myší na ovládací prvek) nebo Change (při změně ovládacího prvku).

### 3.4.1 Základní ovládací prvky VBA

V této kapitole popíši jen základní ovládací prvky jazyku VBA. Budu uvádět ty prvky, které budou pravděpodobně použitelné v mé práci.

**Příkazové tlačítko – CommandButton**

Příkazové tlačítko patří nepochybně mezi nejčastěji používaný ovládací prvek. Na jeho událost Click (při kliknutí) se programují procedury, jejichž hlavním úkolem je vykonat soubor příkazů. Nejčastěji používaná tlačítka jsou typu načti, ok, uzavřít, uložit atd.

**Textové pole – TextBox**

Textové pole se ve formuláři vyskytuje nejčastěji. Do textového pole uživatel doplňuje textové nebo číselné hodnoty. Jeho nejvýznamnější vlastností je Value (hodnota), která představuje obsah pole.

**Pole se seznamem – ComboBox**

Pole se seznamem je ovládací prvek, ze kterého může uživatel vybrat jednu z předdefinovaných hodnot. Jakmile uživatel klikne myší na prvek, zobrazí se mu rozvírací seznam, ze kterého si vybere jednu hodnotu. Hodnoty mohou být předdefinované nebo přiřazené programovým kódem pomocí metody AddItem.

**Popisek – Label**

Popisek umožňuje uživateli vložit do formuláře popisky a komentáře k prvkům. Většinou se přidává před každé textové pole, aby bylo jasné, co má textové pole obsahovat.

**Obraz – Image**

Tento ovládací prvek se používá k zobrazení grafického obrázku. Může být pevně zadán uživatelem, nebo přiřazován programovým kódem pomocí vlastnosti Image.

**Přepínač – OptionButton**

Přepínač dává uživateli na výběr jednu z několika možností. Používá se ve skupinách minimálně po dvou, při čemž může být zvolen vždy jen jeden přepínač z dané skupiny.

**Zaškrtávací tlačítko – CheckBox**

Zaškrtávací tlačítko se používá, pokud je uživateli poskytnuta volba pouze Ano/Ne. Výsledkem tlačítka je pak logická hodnota pravda nebo nepravda.

### 3.4.2 Základní vlastnosti ovládacích prvků

Nyní bych chtěl popsat základní vlastnosti, které má většina z výše uvedených ovládacích prvků.

**Name**

Vlastnost Name určuje jméno ovládacího prvku. Tuto vlastnost musí mít každý prvek vyplněnou a není možné, aby na jednom formuláři nebo jiném objektu byly dva prvky se stejnou hodnotou vlastnosti Name.

**Value**

Value je hodnota ovládacího prvku. Tato vlastnost je dostupná jen u těch prvků, které mohou nějakou hodnotu získat (neplatí pro image).

**Left, Right, Top**

Vlastnosti prvků určující umístění ovládacího prvku ve formuláři nebo jiném objektu, Jsou vyjádřeny číselnou hodnotou, která může být i záporná.

**Width, Height**

Vlastnosti určující šířku a výšku ovládacího prvku. Nabývá pouze kladných číselných hodnot.

**Visible**

Tato vlastnost určuje, zda je ovládací prvek pro uživatele viditelný či nikoliv. Toho se využívá při skrytí některých nežádoucích prvků podle vybraných skutečností ve formuláři. Nabývá logických hodnot True/False (pravda/nepravda).

**Enable**

Podobná vlastnost jako Visible, která určuje, zda-li je prvek uživateli dostupný (jestli ho může měnit).

### 3.4.3 Proměnné a základní datové typy jazyka VBA

Proměnné slouží v programovém kódu k uchování hodnot a jejich následného použití. Je vhodné v deklarační části programu specifikovat datový typ proměnné, pokud jsme si jisti jejím obsahem. Pokud má proměnná specifikovaný datový typ, zabírá méně místa v paměti a lépe se s ní pracuje.

Proměnné se deklarují na začátku programového kódu v tomto formátu: Dim (název proměnné) As (datový typ). VBA nabízí velký výběr datových typů, já uvedu jen ty nejzákladnější, které pravděpodobně využiji ve své práci.

Základní celočíselný datový typ je Byte, který nabízí rozmezí od 0 do 255. Pro mé práci budou vhodnější Integer (-32 768 až 32 768) nebo Long (-2 147 483 648 až 2 147 483 647). Pro použití desetinných čísel je datový typ Single, Double nebo Decimal. [25]

Textový řetězec se ukládá do proměnných datového typu String, u kterého můžeme do závorky zadat maximální možnou délku řetězce. Datový typ Boolean slouží k ukládání logických hodnot pravda/nepravda. Pokud bude nějaká proměnná naplněná objektem (například sešitem programu Excel), použije se pro ni datový typ Object.

Pokud nevíme, jaké hodnoty bude proměnná nabývat, nebo pokud se její datové typy budou v průběhu programu měnit, zvolíme datový typ Variant. Tento datový typ totiž dokáže měnit svoje datové typy podle toho, jaký obsah má v aktuální době. VBA automaticky přiřadí každé proměnné s nespecifikovaným datovým typem datový typ Variant.

[25] (13)

# 4 Vlastní návrhy řešení

V kapitole 2.4 jsem uvedl hlavní nedostatky současného stavu konstrukce svazků stroje. V následující kapitole se pokusím tyto nedostatky vyřešit, navrhnout a vytvořit databázi svazků stroje.

## *4.1 Požadavky na databázi*

Jelikož jsem databázi svazků stroje zpracovával pro velký podnik, měl jsem od zaměstnanců (konstruktérů) předem určené požadavky na takovou databázi. Těch jsem se musel držet a měl jsem tedy stanovené hranice návrhu databáze.

Nejdůležitějším požadavkem konstruktérů bylo, aby databáze byla jednoduchá a aby se snadno obsluhovala. Právě na snadnou obsluhu jsem kladl největší důraz, protože většina konstruktérů má pouze uživatelské znalosti práce s počítačem. Specializují se především na technickou konstrukční práci a nechtějí se zdržovat vyhledáváním v databázi, kopírováním hodnot polí atd. Databáze má být jednoduchá proto, aby se do ní daly bez větších obtíží přidávat záznamy nebo tabulky, a také proto, aby bezproblémově a rychle běžela na současných počítačích v konstrukci.

Dalším požadavkem byla rychlost databáze. V podniku se používá řada programů, které zabezpečují týmovou práci na projektech a návaznost na předchozí produkty. Tyto programy jsou propojeny s databázemi nejen v podniku, ale i se servery v Německu. Ovšem takové programy jsou značně pomalé a také nespolehlivé – často nastane chyba a program se ukončí. Proto má být moje jednoduchá databáze svazků stroje co nejrychlejší, aby nezdržovala pracovníky od práce čekáním na naběhnutí programu.

Konstruktéři také chtěli, aby byla databáze nahraná na serveru, kde by k ní měli přístup pouze konstruktéři. Databáze má být ovšem konstruována tak, aby si mohlo v daném okamžiku prohlížet záznamy a pracovat s databází více uživatelů zároveň.

Společnost samozřejmě požadovala, aby vytvoření a používání databáze bylo co nejlevnější. Nechtěla investovat do žádného rozšiřujícího softwaru, ani školení zaměstnanců.

## 4.2 Rozhraní databáze

Po zvážení všech požadavků a možností společnosti a po konzultaci s konstruktéry připadaly v úvahu řešení v programu Microsoft Access nebo Microsoft Excel.

Vhodné by bylo i použití například Microsoft SQL serveru, ovšem toto by se týkalo spíše složitějších databází a jejich propojení. V mém případě je však třeba vytvořit jednoduchou databázi s pravděpodobně jedinou datovou tabulkou a tuto databázi vybavit formulářem s naprogramovanými funkcemi. Pro tento případ je tedy použití SQL Serveru zbytečné.

Konstruktéři společnosti Siemens většinou standardně vytvářejí datové tabulky v Excelu, které pak používají jako databáze. Tato možnost je ovšem zastaralá a zdaleka nesplňuje požadavky na databázi. Pokud bych chtěl databázi dělat v Excelu, musel bych hodně databázových funkcí programovat a program by pak byl zbytečně složitý a pomalý. Proto jsem tuto možnost již v počátku zavrhl a raději jsem se věnoval návrhu v Accessu, který má databázové funkce integrované.

Nakonec jsem tedy vyvodil jako nejlepší a prakticky jediné řešení zpracovávat databázi v rozhraní Microsoft Access 2003. Microsoft Access je totiž jednoduchý databázový program, který je součástí kancelářského balíčku MS Office, a proto ho většina běžných uživatelů zná. V podniku Siemens Electric Machines s.r.o. je tento balík nainstalován i s programem Microsoft Access. V Accessu je také možnost naprogramovat databázi funkce v jazyku Visual Basic for Applications. Právě programování funkcí nad databází bude hlavní náplní mé databáze.

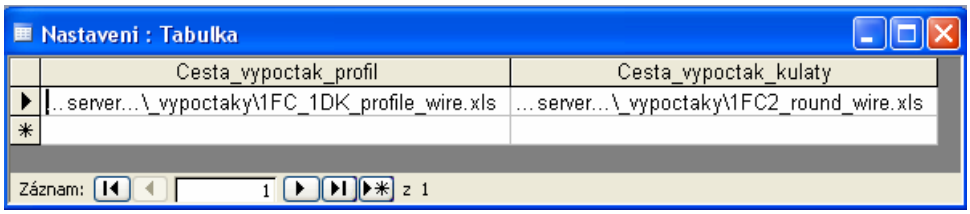
## 4.3 Datové tabulky

V mé databázi se budou vyskytovat jenom 2 tabulky. Jedna z nich bude datová tabulka s mnoha poli, která bude obsahovat data pro jednotlivé svazky stroje. Druhá tabulka bude sloužit jako tabulka s nastavením, kde budou uloženy například cesty k souborům.

### 4.3.1 Tabulka Nastavení

Tabulku Nastavení jsem vytvořil proto, aby měl program odkud brát informace o umístění excelových souborů s výpočtovými listy, ze kterých se budou do formuláře

načítat data. Zároveň však konstruktéři požadovali, aby se toto umístění dalo jimi změnit. Proto jsem cesty k souborům nezahrnul do programového kódu, kde by je konstruktéři těžce hledali. Tabulka Nastavení je vidět na obrázku 3, cesta k výpočtovému listu je nahrazena textem „…server…“.




*Obr. 3 : Tabulka Nastavení*

### 4.3.2 Tabulka Svazky

Tato datová tabulka obsahuje všechny informace o svazku stroje, které jsou nutné pro vymodelování svazku grafickým programem Unigraphics NX. Tabulka obsahuje mnoho polí, do kterých se ukládají přesné informace pro jednotlivé záznamy. Sloupce tabulky jsem pojmenovával podle názvů proměnných v Unigraphicsu, popřípadě podle vlastního uvážení a rad konstruktérů.

Každé pole tabulky má specifikovaný datový typ a další omezení pole. Používal jsem převážně datové typy Text a Jednoduchá přesnost. Číselné pole typu Jednoduchá přesnost jsem volil u těch polí, se kterými se dále matematicky pracovalo ve výpočtech. V ostatních případech jsem volil pole typu Text a různých délek vzhledem k charakteru pole a rad konstruktérů.

Tabulka svazky tvoří prakticky veškerou bázi dat potřebnou pro databázi svazků stroje. V jejích polích jsou obsažena veškerá hlavní konstrukční data, potřebná pro vymodelování svazků stroje. Tato data jsou do tabulky ukládána přes formulář Svazky. Data se mohou do tabulky přidávat samozřejmě i po zobrazení datového listu v Accessu, ale to je pro uživatele zbytečně složité. Na obrázku 4 je vidět náhled tabulky Svazky, celá tabulka je v příloze 1.

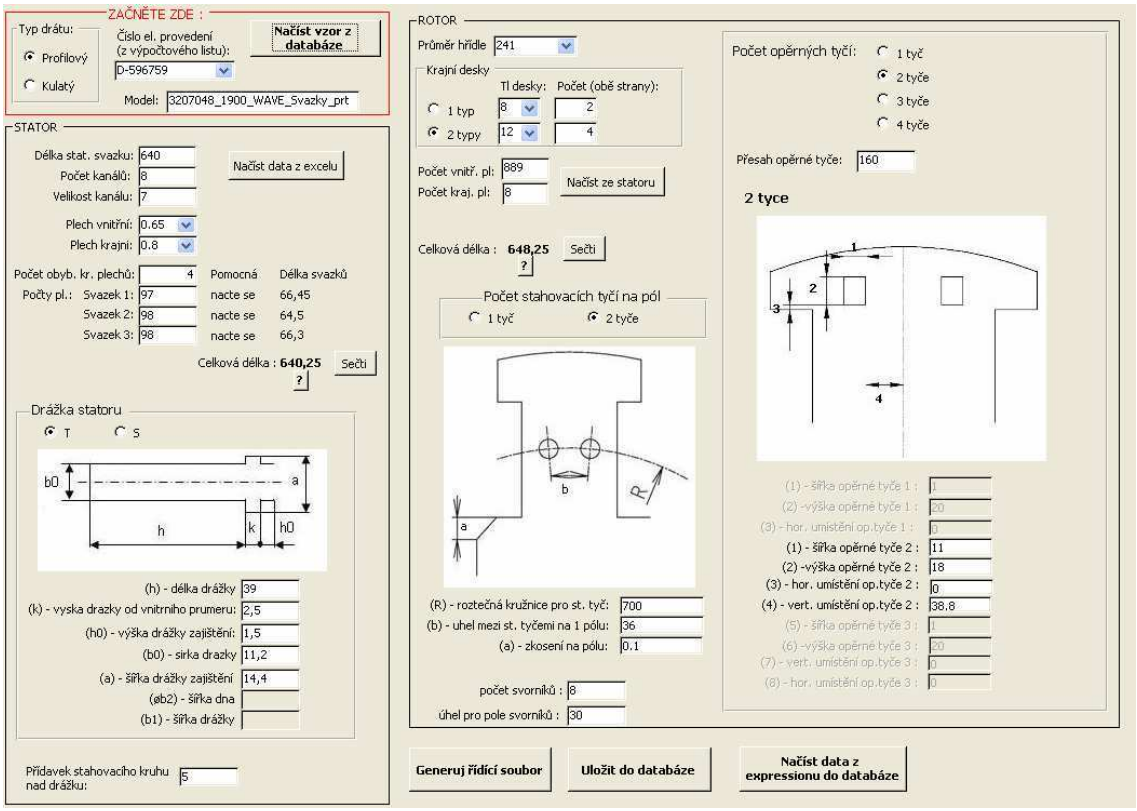
| Název pole | Datový typ | |
|---|---|---|
| ID | automatické čísl | |
| Ulozeni | text | jak bylo uloženo do databáze (načtením z *.exp, uložením přes formulář) |
| Model | text | číslo modelu |
| Cislo_provedeni | text | číslo elektrického provedení z výpočtového listu |
| B1400_N | text | počet rozpěrek -1 vrtaná |
| B1400_S00 | text | šířka prvního vinutí s přídavkem |
| B1400_S01 | text | šířka druhého vinutí s přídavkem 0,16 |
| B1400_V | text | výška vinutí s přídavkem 0,25 |
| B1400_X | číslo | poloha - rozteč rozpěrek |

*Obr. 4 : Náhled tabulky Svazky*

## 4.4 Formulář Svazky

Formulář Svazky je hlavním uživatelským ovládacím prvkem databáze svazků stroje. Přes tento formulář mají uživatelé přístup k veškerým hlavním funkcím databáze. Formulář bude vytvořen hlavně kvůli snadnému ovládání pro konstruktéry. Díky formuláři se nebudou konstruktéři muset učit ovládat Access a jeho prostředí, ale všechny funkce za ně bude dělat formulář s naprogramovanými tlačítky.

### 4.4.1 Vzhled formuláře




*Obr. 5 : Náhled vyplněného formuláře Svazky*

Na obrázku 5 je vidět náhled celého formuláře. Při zobrazení na monitoru s rozlišením 1280 * 1024, což je rozlišení 19“ displeje, zabírá tento formulář plochu bezmála celého monitoru. Proto je při otevření formuláře naprogramována jeho maximalizace na celou obrazovku.

Jak je patrné z obrázku, formulář je rozdělen na 3 orámované části. Každá z těchto částí ohraničuje větší celek logicky seskupených údajů. V první části orámované červenou čarou jsou údaje, které slouží k načtení vzorového svazku z databáze a identifikaci záznamu ve výpočtovém listu v Excelu. V druhé části se

nacházejí data nutná pro konstrukci statorového svazku stroje a ve třetí části jsou data pro konstrukci rotorového svazku stroje.

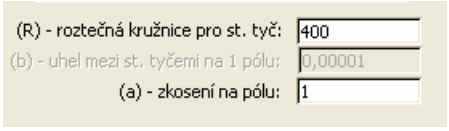
Na formuláři se nachází velké množství textových polí a specifických popisků pro každé textové pole. Popisky určují, co dané textové pole obsahuje. Pro každé pole se navíc zobrazí nápověda (dodatečný popis pro vyplnění pole) ve stavovém řádku po celou dobu, kdy se nachází kurzor v poli, nebo po najetí myší na textové pole a chvilkovém setrvání se zobrazí nápověda přímo vedle pole, jak je vidět na obrázku 6.


*Obr. 6 : Nápověda k textovému poli Svazek 1*

Formulář obsahuje tři obrázky, které se mění podle zatržených možností. Pro změnu obrázku nastavuji vlastnost Visible objektu Image na True nebo False. Ve spodní části formuláře se nacházejí tři tlačítka s hlavními funkcemi formuláře. Tato tlačítka jsou zobrazena na obrázku 7. Právě tato tlačítka slouží k ovládání databáze konstruktéry. Jejich funkce popíši postupně v následujících kapitolách.


*Obr. 7 : Tlačítka formuláře*

Některá pole formuláře a jejich popisky se podle jednotlivých voleb mění na zakázaná nebo povolená. Zakázaná pole ze zobrazují světle šedě a uživatel do nich nemůže napsat žádnou hodnotu, ani je jinak měnit. Zakázaná pole použiji ve formuláři z toho důvodu, že při některých volbách se takováto pole nepoužívají, nebo jsou pevně stanovena. Textová pole jsem měnil na zakázaná a naopak povolená pomocí vlastnosti Enabled a nastavování hodnot této vlastnosti na True nebo False. Příklad zakázaného pole je vidět na obrázku 8, zakázané je pole s popiskem (b).




*Obr. 8 : Zakázané pole*

## 4.5 Funkce formuláře Svazky

V následující kapitole popíši všechny hlavní funkce formuláře svazky. Jedná se především o načtení výpočtového listu, načtení vzoru z databáze do formuláře, generování řídícího souboru pro Unigraphics, uložení dat z formuláře do databáze a načtení dat z expressionu do databáze.

Kvůli zabránění zbytečné složitosti práce popisuji opravdu pouze hlavní funkce formuláře a do příloh umísťuji jen výtahy z programového kódu formuláře. Snažím se vystihnout důležité a nové funkce, naprogramované ve formuláři, a zanedbávám rutinní procedury. U některých textových polí jsou nastaveny procedury na událost AfterUpdate (po aktualizaci). Procedury, které tyto události vyvolávají, mají převážně charakter výpočtu technických proměnných, proto je v této práci neuvádím, abych nepoškodil know-how společnosti. Ze stejného důvodu neuvádím detailní výpočty všech proměnných.

### 4.5.1 Spuštění formuláře a otevření výpočtového listu

Při otevření a načtení formuláře se spustí programový kód, který má za úkol otevření výpočtového listu z Excelu. Pro konstrukci svazků stroje jsou k dispozici dva výpočtové listy, pro profilový drát a pro kulatý drát. Z těchto výpočtových listů je třeba načíst sloupec hodnot „Elektrické provedení“ do rozvíracího seznamu. Elektrické provedení se může vyskytovat v každém výpočtovém listu vícekrát, a načte se tedy do rozvíracího seznamu také vícekrát, ale je to nejlepší identifikátor daného svazku ve výpočtovém listu.

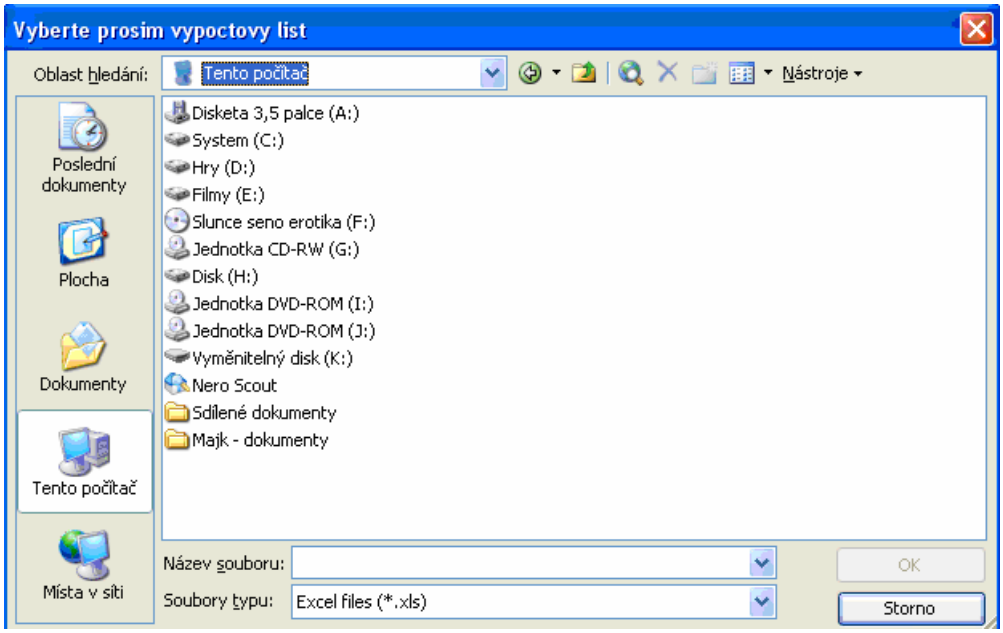
| Elektrické provedení | Typ | Jmenovitá data | | | | | |
|---|---|---|---|---|---|---|---|
| | | El.výkon | napětí | frekvence | účiník | počet pólů | proud statoru |
| | | SN | UN | fN | cosφ | 2p | I2N |
| | | kVA | V | Hz | | | A |
| 01 1111 1111 | 1FC2 561-4 | 1000 | 400 | 50 | 0,8 | 4 | 1443,0 |
| 01 2222 2222 | 1FC2 561-4 | 980 | 400 | 50 | 0,8 | 4 | 1414,1 |
| 01 3333 3333 | 1FC2 562-4 | 1000 | 400 | 50 | 0,8 | 4 | 1443,0 |
| 01 4444 4444 | 1FC2 562-4 | 962,5 | 400 | 50 | 0,8 | 4 | 1388,9 |
| 01 5555 5555 | 1FC2 562-4 | 1110 | 400 | 50 | 0,8 | 4 | 1601,7 |

*Obr. 9 : Náhled výpočtového listu*

Podle toho, který výpočtový list zvolíme, se načtou některá data do textových polí, respektive se zatrhnou určité volby. Také se načte sloupec s elektrickým provedením do rozvíracího seznamu.

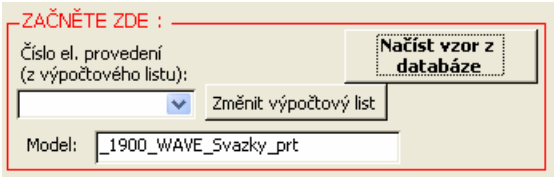
Otevření výpočtového listu v Excelu jsem řešil dvěma způsoby, při čemž v současné době konstruktéři používají druhý způsob.

První způsob používal k otevření funkci FileDialog. Při otevření formuláře se zobrazilo dialogové okno jako na obrázku 10. V tomto okně uživatel vybral výpočtový list a potvrdil.




*Obr. 10 : Dialogové okno pro otevření Excelu*

Jakmile uživatel vybral jeden z výpočtových listů, načetl se hodnoty elektrického provedení do rozvíracího seznamu a zatrhly se volby pro daný typ drátu. Pokud chtěl uživatel svou volbu změnit, musel kliknout na tlačítko „Změnit výpočtový list“, jak je vidět na obrázku 11. Potom se znovu objevilo dialogové okno pro dotaz na cestu k souboru.




*Obr. 11 : Tlačítko pro změnu výpočtového listu*

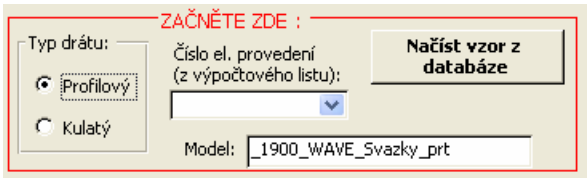
Část programového kódu s touto variantou je v příloze 2.

Výhodou této možnosti byla univerzálnost řešení a nezávislost na umístění výpočtového listu. Uživatel se mohl pomocí průzkumníka dostat kamkoli na disk a označit výpočtový list. Ovšem nevýhodou bylo, že cestu k výpočtovému listu musel

uživatel zadávat při každém otevření formuláře znovu. Toto konstruktéry zdržovalo a bylo to zbytečné, protože umístění výpočtových listů je na serveru stálé a nemění se.

Z těchto důvodů jsem se rozhodl zvolit jiné řešení, v němž bude cesta k výpočtovým listům zadána pevně. Nejprve jsem cestu zadal napevno do programového kódu. Ovšem v případě změny umístění výpočtových listů by bylo třeba zasáhnout do programového kódu, a to by mohlo být pro konstruktéry složité. Proto jsem vytvořil již zmiňovanou tabulku Nastavení. Tato tabulka obsahuje pouze dvě pole, ve kterých jsou uloženy cesty k výpočtovým listům pro profilový a pro kulatý drát.

Při spuštění formuláře se potom načtou do polí hodnoty elektrického provedení pro oba dva výpočtové listy. A podle toho, který typ drátu je zatržený, se načtou do rozvíracího seznamu čísla elektrického provedení z daného výpočtového listu (respektive pole). Jak zatrhnout typ drátu je vidět na obrázku 12.




*Obr. 12 : Zatržení typu drátu*

Programový kód pro otevření výpočtového listu pro profilový drát je obsažen v druhé části přílohy 2.

### 4.5.2 Načtení výpočtového listu do formuláře

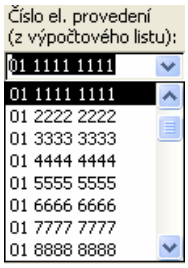
Nejprve jsem výpočtový list přiřadil proměnné MyXL pomocí příkazu GetObject. Z výpočtového listu se musí nejdříve načíst číslo elektrického provedení z Excelu do rozvíracího seznamu. Toto načítání musí být interaktivní, protože výpočtové listy se mohou měnit. Jelikož se ve výpočtovém listu vyskytují ve sloupcích mezery, musel jsem použít složitější cyklus. Zvolil jsem cyklus Do Until se třemi spojenými podmínkami logickou spojkou And. Cyklus se tedy provádí, dokud nejsou splněny všechny tři podmínky zároveň. Tyto tři podmínky určují, že hodnota buňky A a určitého řádku (x) musí být prázdná. Hodnoty řádku v jednotlivých podmínkách se liší o 5, aby se předešlo předčasnému ukončení načítání kvůli mezerám ve sloupci. Jakmile jsou řádky x, x+5 a x+10 prázdné, cyklus se ukončí.

Hodnota buňky ve sloupci A a daném řádku (např. A10) se potom načte do pole na určité místo podle hodnoty řádku. Poté se spustí procedura „Fce_Reload“, která nejprve vymaže rozvírací seznam pro číslo elektrického provedení a následně do něj pomocí cyklu nahraje všechny hodnoty elektrického provedení pro daný výpočtový list. Procedura „Fce_Reload“ dále vynuluje zatržené volby a textová pole a nastaví hodnotu předdefinovaných polí.

Tím se ukončilo přiřazení výpočtového listu formuláři. Výtah programového kódu této části je v příloze 3. V příloze 3 je uveden kód pro načtení výpočtového listu pro profilový drát, pro kulatý drát je postup naprosto stejný.

### 4.5.3 Načtení polí formuláře podle elektrického provedení

Jakmile konstruktér vybere z rozvíracího seznamu (Pole_list) hodnotu elektrického provedení, zkopírují se z daného výpočtového listu hodnoty některých polí do formuláře. Tato funkce je naprogramovaná na událost AfterUpdate pro rozvírací seznam a v případě potřeby (znovunačtení) na tlačítko „Načíst data z Excelu“.




*Obr. 13 : Rozvírací seznam pro elektrické provedení*

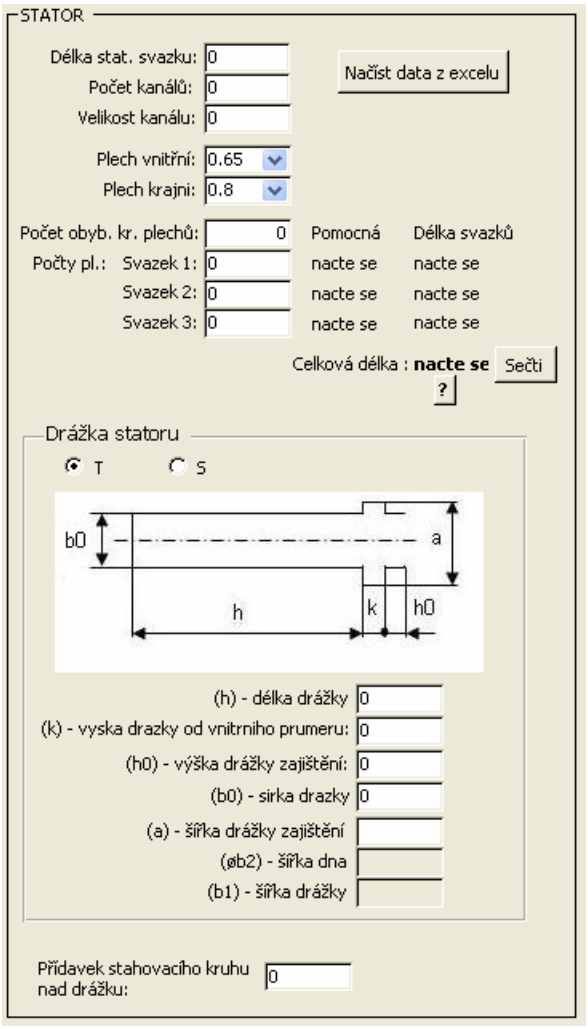
Podle vlastnosti ListIndex rozvíracího seznamu se určí řádek s hodnotami v Excelu a tyto hodnoty se načtou do polí formuláře. V příloze 4 je vidět ukázka takového načítání pro soubor polí týkajících se drážky statoru.

Na událost AfterUpdate, tedy po změně hodnoty, rozvíracího seznamu Pole_list (s elektrickým provedením) je naprogramována ještě jedna významná funkce. Po změně hodnoty rozvíracího seznamu prohledá program databázi a zjistí, jestli je svazek se stejným číslem elektrického provedení v databázi. Pokud ano, umožní tento svazek načíst do formuláře. Tím konstruktérovi značně usnadní vyplňování polí. Konstruktér si všechna pole a zatržené volby projde a upraví jen ty, které chce změnit. Tato funkce je naprogramována v příloze 5, při čemž volanou proceduru v modulu funkce „Modul_funkce.nacti_vzor(idd)“ vysvětlím později v kapitole 4.6.

### 4.5.4 Funkce pro načtení vzoru

Pro načtení vzorového svazku z databáze slouží tlačítko „Načíst vzor z databáze“. Při kliknutí na toto tlačítko se otevře formulář „Svazky_vzor“. Funkce tohoto formuláře popíši v kapitole 4.6.

### 4.5.5 Další funkce pro oblast Stator

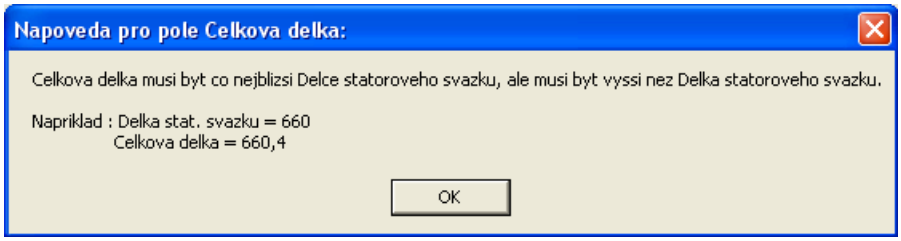



*Obr. 14 : Oblast Stator*

Jak je vidět na obrázku 14, v oblasti statoru se nachází tlačítko „Načíst data z Excelu“, jehož funkci jsem popsal v kapitole 4.5.3. Pro tloušťky vnitřních a krajních plechů jsou k dispozici rozvírací seznamy se dvěma předdefinovanými hodnotami 0,65 a 0,8. Počty plechů pro Svazek 1, 2 a 3 doplní konstruktér , při čemž mu k tomu slouží pomocné hodnoty napravo od pole. Na obrázku 14 je v nich napsáno „nacte se“ (načte se), stejně jako u ostatních polí s výpočty, protože nejsou vyplněny předchozí pole nutné pro výpočty těchto pomocných hodnot. Při vyplňování polí se postupně

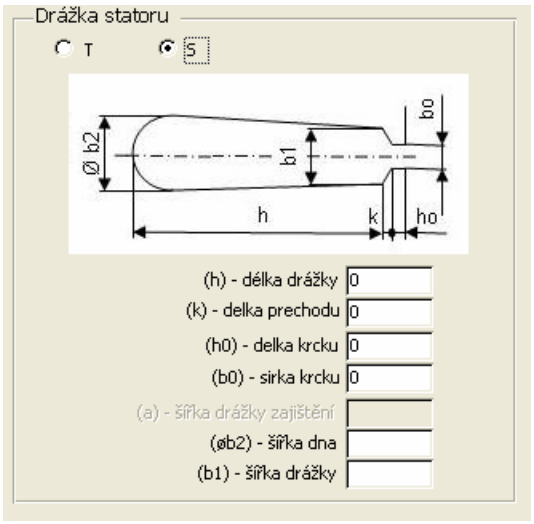
dopočítávají hodnoty, protože jsou u jednotlivých textových polí nastaveny události AfterUpdate (po aktualizaci).

Napravo od pomocných hodnot jsou délky jednotlivých svazků a pod nimi celková délka svazků. Pro výpočet celkové délky slouží tlačítko „Sečti“, které sečte délky jednotlivých svazků. Tlačítko se symbolem „?“ slouží jako nápověda pro správné pochopení celkové délky. Při kliknutí na toto tlačítko se zobrazí MsgBox, který je na obrázku 15.



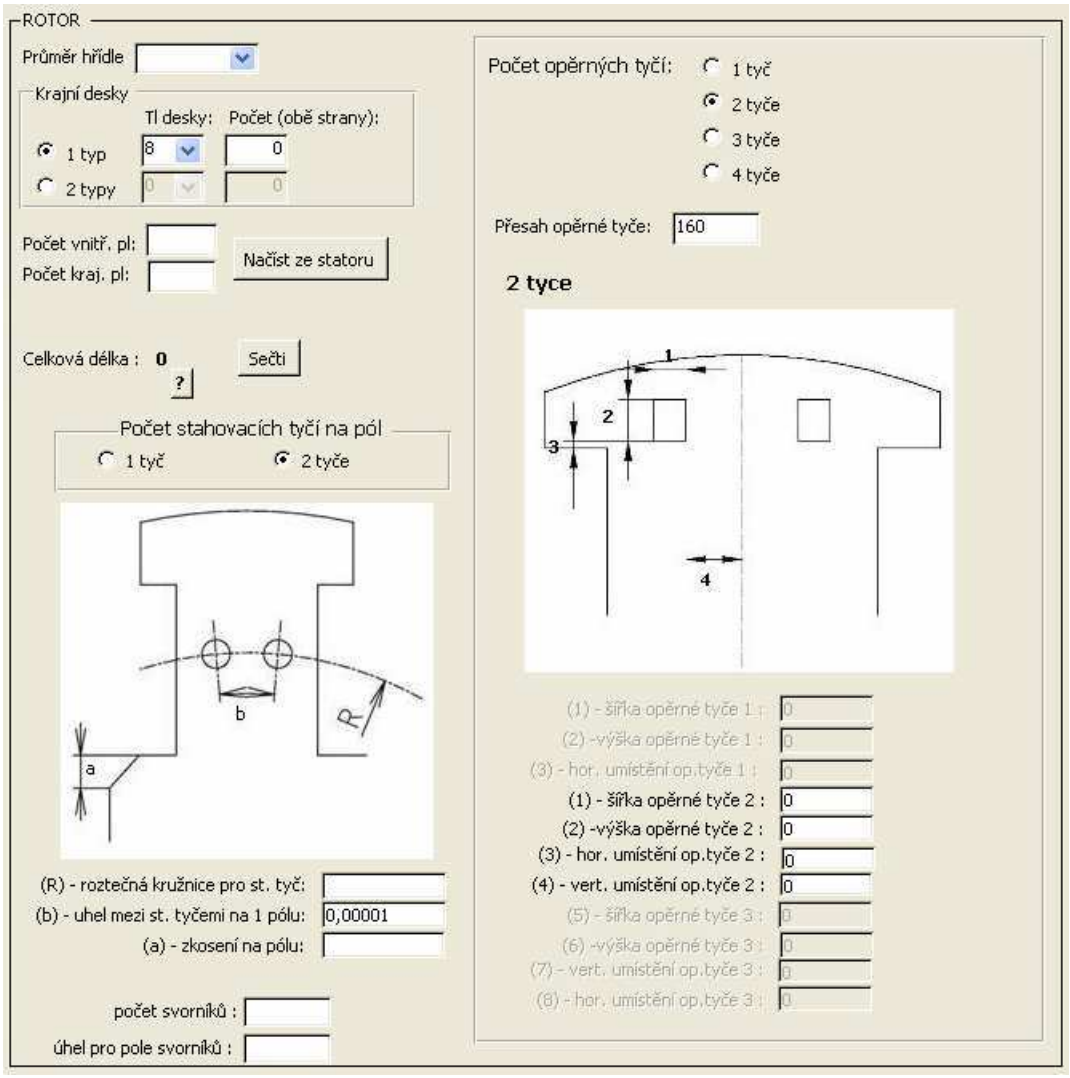

*Obr. 15 : MsgBox celková délka*

Podle volby drážky statoru „T“ nebo „S“ se upraví soubor polí pro drážku statoru. Pole pro drážku typu T jsou vidět na obrázku 14, pro drážku typu S na obrázku 16. Jak je vidět, upravují se i popisky k textovým polím a některá textová pole jsou zakázaná.




*Obr. 16: Drážka statoru typu S*

### 4.5.6 Další funkce pro oblast Rotor




*Obr. 17 : Oblast Rotor*

Jak lze vidět na obrázku 17, pro oblast rotoru je k dispozici opět několik pomocných funkcí a předdefinovaných rozvíracích seznamů. Seznamy jsou pro průměr hřídele a tloušťku krajních desek, protože v těchto případech se vybírá jedna z předdefinovaných hodnot. Pokud je zvolen jeden typ krajních desek, zakáží se pole pro druhý typ. Tlačítko „Načíst ze statoru“ slouží k načtení počtu vnitřních a krajních plechů z oblasti statoru. Obě tlačítka kolem pole s celkovou délkou mají stejný význam jako u statorové oblasti a jsou popsány v kapitole 4.5.5.

V oblasti rotoru jsou k dispozici ještě dvě skupiny voleb ohraničené leptavým rámem. První skupinu voleb představuje „Počet stahovacích tyčí na pól“. Podle toho, zda je zvolena jedna nebo dvě tyče, se mění obrázek a nastavuje se uzamčení pole pro

úhel mezi stahovacími tyčemi na jednom pólu. Pokud je totiž zvolena jedna tyč, je tento úhel konstantní a nelze jej měnit.

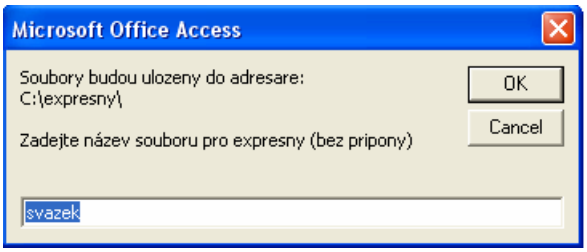
Další skupinou voleb je „Počet opěrných tyčí“, kde máme na výběr od jedné do čtyř tyčí. Podle toho, kterou volbu zatrhneme, se nám zpřístupní pouze určitá pole a ostatní zůstanou zamčená. Stejně jako u ostatních skupin voleb, i zde se mění obrázky podle zatrženého počtu opěrných tyčí. Tuto skupinu voleb jsem řešil pomocí příkazu Select Case, kdy podmínkou pro rozdělení byla hodnota zatržené volby (1 - 4).

### 4.5.7 Tlačítko Generuj řídící soubor

Funkcí tohoto tlačítka je vygenerovat textový soubor, který slouží jako vstupní soubor pro načtení hodnot expressionu grafickým programem Unigraphics. Jedná se o jednu z hlavních funkcí této databáze.

Na začátku funkce se zavolá procedura „nacti_excel“. Tato procedura slouží ke spočítání všech proměnných, které se budou do textového souboru generovat. V proceduře se využívají data z výpočtového listu v Excelu, která doplňují hodnoty textových polí a rozvíracích seznamů formuláře. Tato procedura je značně rozsáhlá a náročná svými výpočty, ukázky částí procedury jsou v příloze 6.

Po spočítání všech proměnných vyskočí dialogové okno typu InputBox. Toto okno oznámí uživateli, kam bude textový soubor s expressiony umístěn, a zeptá se ho na název souboru. Toto okno je vidět na obrázku 18.




*Obr. 18 : InputBox s dotazem na název souboru*

Program soubor automaticky umístí do adresáře, ve kterém se nachází soubor s databází, a v něm vytvoří složku „expresnyL, pokud již neexistuje. Program také zjistí, jestli neexistuje v adresáři soubor se stejným názvem. Pokud ano, upozorní nás dialogem a zeptá se, jestli chceme soubor přepsat, jak je vidět na obrázku 19. Pokud zvolíme „Ne“, objeví se nám znovu InputBox s dotazem na název souboru. Všechny tyto funkce jsou naprogramované v příloze 7.


*Obr. 19 : Upozornění na existenci souboru*

V další části program otevře textový soubor pro zápis pomocí funkce Open For Output a do tohoto textového souboru zapisuje postupně hodnoty jednotlivých proměnných pomocí funkce Print. Hodnoty proměnných musí být však nejprve převedeny na text pomocí funkce Str, aby s nimi funkce Print mohla bezchybně pracovat. Jakmile jsou zapsány všechny proměnné, textový soubor se zavře příkazem Close.

Po vygenerování se tento soubor otevře v poznámkovém bloku. K tomu jsem použil metodu Run objektu Wscript.Shell. Nakonec program zavolá proceduru pro ukládání do databáze.

Náhledy výše zmíněných funkcí jsou naprogramované v druhé části přílohy 7.

Vygenerovaný textový soubor s parametry si potom konstruktér načte do programu Unigraphics jako vstupní soubor s expressiony. Tyto soubory se vytvářejí s příponou *.exp, se kterou se dá pracovat jako s textovou příponou *.txt. Unigraphics pak z předdefinovaných skic a modelů a s vstupními expressiony vytvoří model svazků stroje. Náhled vygenerovaného souboru s expressiony je vidět na obrázku 20.

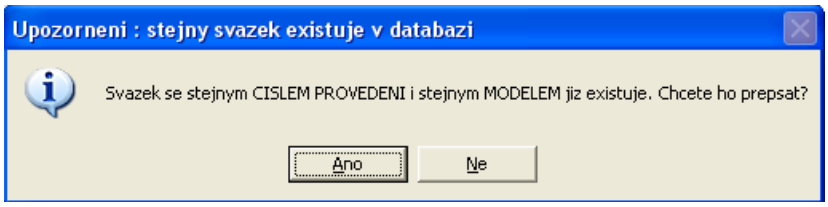
```
B1856_E00= 1 //pocet stahovacich tyci na pol 0-jedna 1-dve tyce
B1856_F= 30 // roztec mezi stahovacimi tycemi na jednom polu
B1856_L01= 860 //delka svorniku
B1856_N00= 4 //pocet stahovacich tyci
B1856_N01= 8 //pocet svorniku
B1856_U00= 90 //uhel pro stahovaci tyc
B1856_U01= 30 //uhel pro pole svorniku
B1867_L00= 642.2 //celkova delka plechu tl. 0.65
B1867_L01= 8 //celkova delka plechu tl. 0.80
B1867_N00= 988 //pocet plechu 0.65 na svazek
B1867_N01= 10 //pocet plechu 0.80 na svazek
B1870_C= 12.5 //zkoseni na polu
B1870_D00= 315//prumer opracovani celni desky pro pripojeni k hrideli
```

*Obr. 20 : Náhled souboru s expressiony*

### 4.5.8 Tlačítko Uložit do databáze

Tlačítko s popiskem „Uložit do databáze“ ukládá všechny vypočítané proměnné a hodnoty textových polí do datové tabulky Svazky.

Programový kód tohoto tlačítka nejprve zkontroluje, zda se v datové tabulce nenachází soubor se stejným číslem elektrického provedení. Pokud ne, vytvoří nový záznam a začne do něj zapisovat hodnoty. Když svazek se stejným číslem elektrického provedení existuje, zjišťuje program, jestli má i stejné číslo modelu. Pokud existuje svazek se stejným číslem elektrického provedení i modelu, program se zeptá, zda chceme svazek přepsat (viz obrázek 21), protože se prakticky musí jednat o tentýž svazek. Pokud uživatel potvrdí volbu „Ano“, svazek se přepíše, pokud uživatel vybere „Ne“, procedura se ukončí (nelogická volba).




*Obr. 21 : Existence stejného svazku*

Poté již program zapisuje do jednotlivých polí záznamu v tabulce hodnoty proměnných. Nakonec program hodnoty nového záznamu uloží a ukončí propojení s databází. Programový kód těchto funkcí je zobrazen v příloze 8, z ukládání do záznamu je v příloze pouze malá část.

### 4.5.9 Tlačítko Načíst data z expressionu do databáze

Funkce, která načítá data ze souborů expressionů do datové tabulky Svazky je asi nejsložitější funkcí celé databáze. Soubor s expressiony je veliký soubor, který obsahuje několik tisíc parametrů, a já z něho potřebuji vybrat jen takové, které se týkají svazků, a které chci ukládat do databáze. V následujícím textu popíši slovně funkci této procedury. Vybrané části kódu se nacházejí v příloze 9.

Nejprve si program vytvoří pomocný soubor „temp.txt“ na disku H (záložní disk uživatelů). Poté uživatel vybere soubor pomocí FileDialogu s nastavenými filtry textový soubor s příponou *.exp, který program otevře pro vstup (Open For Input), a začne načítat jeho řádky pomocí funkce Line Input do proměnné „strline“.

Jakmile narazí program na řádek s cestou souboru, odstraní z cesty všechny znaky před lomítky, čímž mu zůstane název souboru, který je shodný s názvem modelu. Název modelu potřebuje program znát, protože expressiony svazků stroje jsou uloženy v textu ve formátu „!*název_modelu* _1900_WAVE_Svazky_prt“. Tento text tedy bude

program vyhledávat. Funkcí Line Input se vždy načte jeden řádek a pomocí funkce Instr zjistí, jestli obsahuje daný řetězec. Pokud text nenajde v celém souboru, oznámí to uživateli, InputBoxem si vyžádá jiný název a hledá od začátku dokumentu.

Jakmile najde text, začne jej po řádcích kopírovat do pomocného souboru až do té doby, než najde další vykřičník, čímž začíná jiný soubor expressionů. Pak zavolá funkci „Zapis_db_sv“, která uloží pomocný soubor do databáze.

Funkce „Zapis_db_sv“ nejprve otevře záznam v tabulce Svazky a prohledá celou tabulku, zda-li se v tabulce nevyskytuje svazek se stejným modelem. Pokud ano, vždy porovnává hodnotu v databázi s hodnotou v textovém souboru. Když se hodnoty liší, upozorní uživatele a zeptá se, jestli chce uživatel hodnotu přepsat nebo zanechat. Toto porovnávání je v kódu rozděleno podle toho, zda jsou obě porovnávané proměnné datového typu číslo nebo text, popřípadě jejich kombinace (v příloze 9 ukázka pro kombinaci typů). Toto rozdělení je třeba kvůli oddělení mezer v hodnotách.

Pak pro oba případy nejprve v každém řádku vyhledá parametr, hodnotu parametru a komentář. Vyhledávání je založeno na funkci Instr a pevné struktuře souboru s expressiony (viz obrázek 20). V každém řádku je nalevo od pozice „=“ název parametru, napravo od „=“ se nachází hodnota parametru a za „//“ je komentář k parametru. Této struktury jsem využil v načítání jednotlivých položek řádku (viz příloha 9).

Nakonec zbývá otevřít záznam v datové tabulce Svazky, buď existující nebo nově vytvořený, a pomocí příkazu Select Case roztřídit data podle parametru expressionu do polí datové tabulky. Po roztřídění se záznam uloží a uzavře. Tento programový kód je v poslední části přílohy 9.

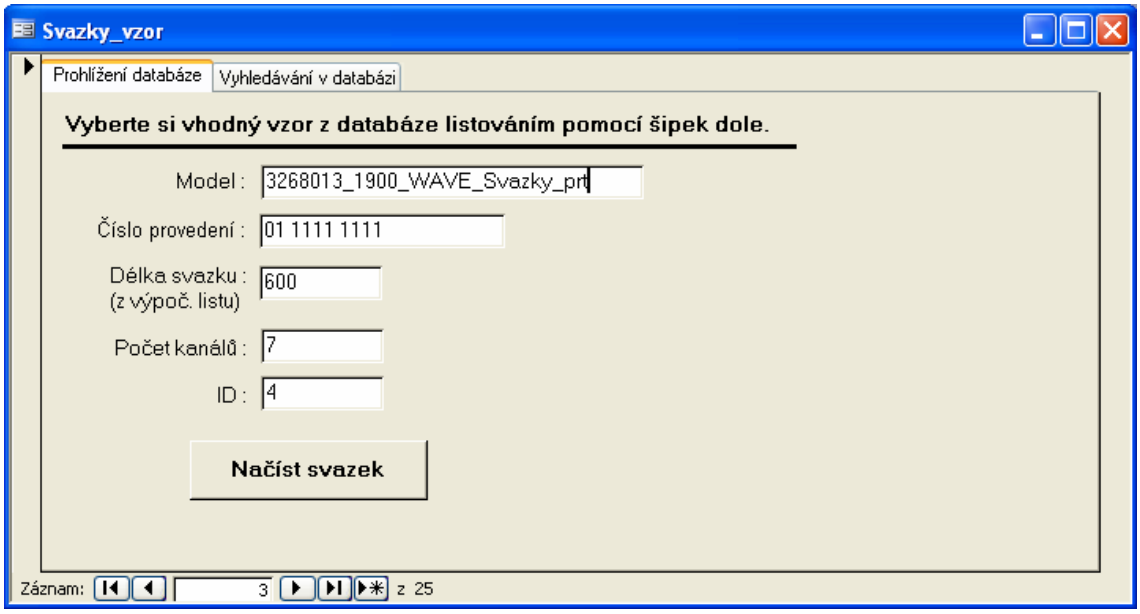
## *4.6 Formulář Svazky_vzor*

Formulář Svazky_vzor slouží k načtení svazku z databáze do formuláře Svazky. Toto je užitečné v případě, že chce konstruktér upravit stávající svazek, nebo chce vytvořit nový, který je podobný některému svazku z databáze. Na tomto formuláři se nacházejí dvě záložky, a to Prohlížení databáze a Vyhledávání v databázi. K vytvoření těchto záložek jsem použil ovládací prvek MultiPage.

### 4.6.1 Záložka Prohlížení databáze

V záložce Prohlížení databáze jsou pole se základními informacemi o svazku stroje. Pomocí nich může konstruktér vybrat vhodný svazek z databáze a načíst ho do formuláře Svazky. Databází se listuje prostřednictvím ovládacích prvků Accessu, umístěných na spodní liště formuláře.

Tato záložka obsahuje jedno příkazové tlačítko „Načíst svazek“, které načte aktuálně zobrazený svazek do formuláře Svazky. Záložka je zobrazena na obrázku 22.



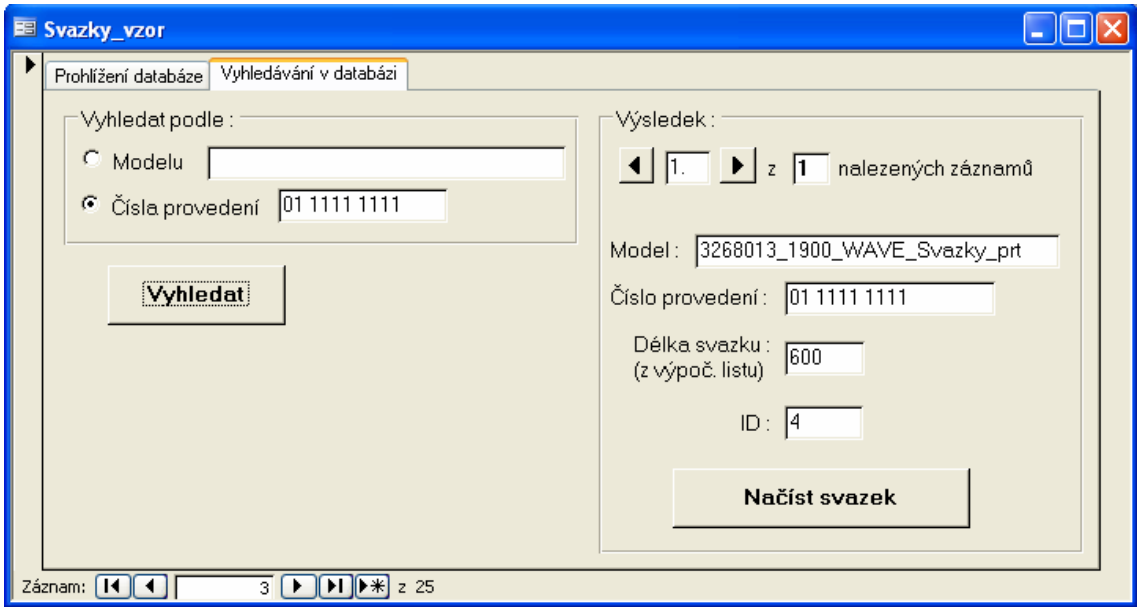

*Obr. 22 : Záložka Prohlížení databáze*

**Funkce tlačítka Načíst svazek**

Program nejprve uloží do proměnné idd aktuální ID zobrazeného svazku a spustí proceduru z modulu Modul_funkce „nacti_vzor(idd)“ s parametrem idd.

Tato procedura otevře požadovaný záznam podle parametru idd. Podle pole „Tvar_drazky“ zatrhne volbu ve formuláři Svazky a přiřadí správný výpočtový list pomocí funkce Fce_Reload (viz kapitola 4.5.2). Poté zavolá funkci Nacti_data (viz kapitola 4.5.7). Tyto dvě funkce se spouštějí proto, aby se správně načetly pomocné výpočty ve formuláři, které se do datové tabulky neukládají. Jakmile obě funkce proběhnou, začnou se do jednotlivých polí formuláře Svazky přiřazovat pole z datové tabulky. Nakonec se otevřený záznam uzavře a otevře se formulář Svazky s načtenými hodnotami. Úvodní část programového kódu se nachází v příloze 10.

### 4.6.2 Záložka Vyhledávání v databázi




*Obr. 23 : Záložka Vyhledávání v databázi*

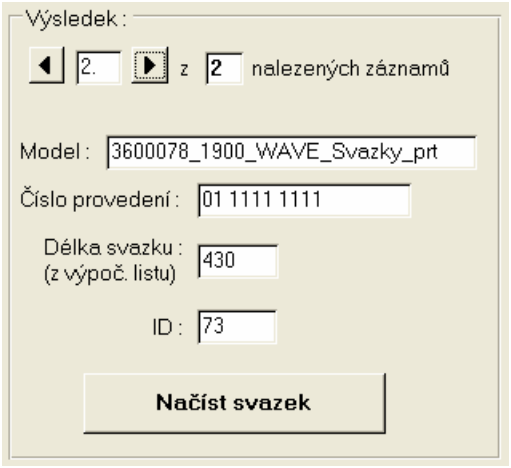
Tato záložka slouží k vyhledávání požadovaného záznamu v datové tabulce Svazky. Konstruktér nemusí listovat databází a hledat požadovaný svazek, stačí mu zadat kritérium vyhledávání a program vyhledá svazek za něj. Vyhledávat se dá podle čísla elektrického provedení nebo podle čísla modelu. Zatržena může být vždycky jen jedna volba.

Kód tlačítka Vyhledat nejprve vynuluje proměnné a upraví jejich syntaxi a otevře záznamy tabulky Svazky. Pomocí funkce Select Case se zjistí, podle kterého pole se má vyhledávat. Pro obě větve příkazu Select Case se potom postupuje stejným způsobem. Program prohledává datovou tabulku, a pokud nalezne záznam se shodnou hodnotou kritéria, uloží hodnotu ID tohoto záznamu do pole „vyhl_array“. Do tohoto pole se ukládají záznamy v pevném formátu „*ID_záznamu, ID_dalšího záznamu, … .* S tímto formátem pak pracují funkce tlačítek pro předcházející a následující seznam.

Poté program přiřadí nalezené záznamy do proměnné „rst2“. Nakonec zavolá proceduru při kliknutí na tlačítko Další, díky které se začnou zobrazovat záznamy od prvního nalezeného. Programový kód tohoto úseku je obsažen v příloze 11.

Jestliže není v databázi žádný záznam nalezen, program na tuto skutečnost upozorní a ukončí průběh kódu.

Výsledek vyhledávání se pak zobrazí v pravé části formuláře, která má podobnou strukturu jako záložka Prohlížení databáze (viz obrázek 24). V horní části se nachází informace o pořadí aktuálního záznamu v počtu nalezených výsledků. Pro pohyb mezi nalezenými záznamy slouží tlačítka s šipkou vlevo (předchozí záznam) a šipkou vpravo (následující záznam).




*Obr. 24 : Výsledek vyhledávání*

#### 4.6.2.1 Tlačítko Další – pro následující záznam

Procedura tlačítka nejprve navýší globální proměnnou „akt“ o jedničkou. Proměnná „akt“ značí pořadí aktuálního záznamu ve vyhledaných záznamech. Pokud je hodnota proměnné „akt“ větší než je počet nalezených záznamů, procedura se ukončí, protože již máme zobrazen poslední záznam.

V další části zjistí program ID záznamu, který se má zobrazit. Pokud chce uživatel zobrazit například druhý nalezený záznam, najde program v prvním prvku pole vyhl_array čárku, která značí konec prvního prvku pole, a vezme text mezi touto čárkou a následující čárkou. Tato část textu se převede na číslo, které je hodnotou ID požadovaného záznamu (proměnná „ID_nove“). Poté program vyhledá toto ID v datové tabulce Svazky a přiřadí hodnoty z tabulky do polí formuláře. Nakonec vypíše hodnotu pořadí aktuálního záznamu do formuláře a přesune se v datové tabulce na konec, aby při dalším vyhledávání začínal od prvního záznamu.

Programový kód, popisující výše popsané funkce, je zobrazen v druhé části přílohy 11.

#### 4.6.2.2 Tlačítko Předchozí

Funkce tlačítka pro předchozí záznam je principielně stejná jako u tlačítka pro následující záznam. Jediný rozdíl je v tom, že se na začátku procedury proměnná „akt“ o jedničku snižuje. A pokud je hodnota proměnné „akt“ menší než 1, nastaví se její hodnota na 1 a procedura se ukončí. V takové případě je totiž zobrazen první záznam. Kód této rozdílné části je v poslední části přílohy 11.

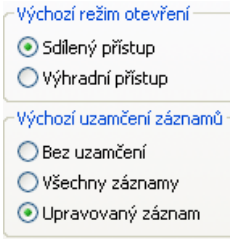
## *4.7 Umístění databáze a uživatelské rozhraní*

Databáze svazků stroje bude umístěna na síťovém serveru. Server je rozdělen na složky, při čemž k některým složkám mají přístup jen určití uživatelé. Konstruktéři mají pro tento účel vytvořeno více složek, do kterých mají přístup jen oni.

Databázi svazků stroje tedy umístím do jedné z takových složek, protože k ní mají mít přístup pouze konstruktéři. Databáze se bude celá spouštět ze serveru. Je nežádoucí, aby si konstruktéři kopírovali databázi na svoje lokální disky, protože by pak ztrácela aktuálnost. Proto je konstruktérům doporučeno, aby si kopírovali pouze zástupce na soubor s databází.

Je nutné, aby mohlo s databází pracovat více uživatelů najednou, proto je třeba tomuto databázi přizpůsobit. Access má k dispozici nástroje, pomocí kterých se dá toto upravit, najdeme je v nabídce Nástroje-Možnosti na kartě Upřesnit.

V tomto nastavení jsem zvolil výchozí režim otevření jako sdílený přístup. Díky tomu může otevřít databázi více uživatelů zároveň. Dále jsem nastavil uzamčení upravovaného záznamu. Pokud bude chtít uživatel upravovat záznam, který již upravuje někdo jiný, Access ho na tuto skutečnost upozorní a nedovolí mu tento záznam upravovat. Následující nastavení je zobrazeno na obrázku 25.




*Obr. 25 : Sdílení a uzamčení záznamů*

# 5 Ekonomické zhodnocení a přínos návrhu řešení

Přínos vytvoření databáze svazků stroje spočívá ve zjednodušení, zrychlení a zkvalitnění procesu konstrukce svazků stroje. Všechny tyto přínosy jsou realizovány díky uživatelskému formuláři.

Díky naprogramovaným funkcím formuláře a možnosti načtení vzorového svazku z databáze je konstrukce svazku jednodušší. Konstruktér už nemusí upravovat hodnoty parametrů ručně v grafickém programu Unigraphics, ale jen zadá několik doplňujících hodnot do polí formuláře a ten vygeneruje textový soubor se všemi parametry. Tím se proces nejen zrychlí, ale i zkvalitní, protože výpočty parametrů jsou prováděny automaticky, a je tedy menší pravděpodobnost, že dojde k chybě.

Konstrukce svazků stroje je díky této databázi ale především mnohem efektivnější. Konstruktér vyplní do formuláře pouze několik upřesňujících dat a další desítky parametrů spočítá program automaticky. Díky tomu databáze umožňuje větší efektivnost konstrukce a tím i výroby. Místo aby se konstruktér zdržoval počítáním parametrů a přepisováním hodnoty, může vykonávat další práci. Pomocí databáze a jejího naprogramovaného formuláře se čas potřebný ke konstrukci svazků stroje zkrátí průměrně o 1/3 doby.

Celkový svazek stroje se skládá z více svazků a pro každou takovou „část“ je k dispozici model. Dříve museli konstruktéři upravovat hodnoty v každé této části modelu. Ovšem formulář databáze svazků stroje generuje parametry pro všechny části dohromady do jednoho textového souboru. Do modelovací programu Unigraphics se nahraje tento jeden soubor pro jakoukoliv část a Unigraphics si z něj vybere jen ty parametry, které jsou v dané části obsaženy.

S touto databází také budou přehledně uspořádány všechny doposud vytvořené svazky stroje na jednom místě. Pomocí dalšího formuláře lze tuto databázi pohodlně prohlížet a také v ní vyhledávat a prohlížet výsledky vyhledávání. Tím se značně urychlí vyhledávání informací o svazcích stroje, protože v minulosti byly tyto informace převážně v papírově podobě.

Shrnutí přínosů databáze svazků stroje společnosti:

- zrychlení konstrukce stroje
- zkvalitnění konstrukce svazků stroje
- zefektivnění konstrukce a práce
- zjednodušení konstrukce svazků stroje
- přehledné uspořádání, prohlížení a vyhledávání v databázi vytvořených svazků stroje
- úspora času a práce konstruktéra

Náklady na vytvoření databáze byly prakticky rovny výši mé hodinové sazby vynásobené dobou strávenou na vytvoření databáze. Software ani hardware společnosti nebylo třeba upravovat, což také bylo jedním z požadavků firmy. Vzhledem k době, kterou jsem nad databází a jejím odladění strávil, a výši mé hodinové mzdy, jsem náklady vykalkuloval na částku asi 10 tisíc korun.

Finanční přínosy databáze jsou velmi těžko peněžně vyčíslitelné. Zcela jistě databáze vedla ke zkvalitnění a hlavně zrychlení konstrukce svazků stroje. Z toho plyne, že se ušetřil pracovní čas konstruktéra, což vede nejen k peněžní úspoře, ale hlavně k větší efektivnosti konstrukce i výroby. Tento pracovní čas pak může konstruktér vynaložit na vytváření dalších hodnot pro společnost. Jsem si jist, že v průběhu času výnosy z databáze převýší náklady na její pořízení.

# Závěr

Tato bakalářská práce pojednává o návrhu databáze svazků stroje a jejím cílem bylo takovou databázi vytvořit. V mé práci se mi podařilo vytvořit takovou databázi svazků stroje, která generuje textový soubor s parametry pro grafický program, a která načítá data z tabulek v Excelu.

Jelikož jsem návrh této databáze konzultoval v průběhu vytváření s konstruktéry, je výsledná databáze odrazem jejich základních požadavků. V databázi jsem se snažil ošetřit všechny možné chyby, které nastaly a mohly by nastat. Je ovšem docela možné, že se nějaká menší chyba ještě objeví a bude nutné poupravit programový kód. Já jsem ve společnosti zaměstnán, takže v případě problémů je ihned upravím.

Pravdou je, že tato databáze je dělaná přesně pro společnost Siemens Electric Machines s.r.o., ovšem její použití by mohlo být univerzálnější. Hlavní síla databáze spočívá ve formuláři a jeho naprogramovaných funkcích. A právě použití těchto funkcí je možné v jakékoliv jiné databázi. Proto jsem se ve své práci nezaměřoval na přesné výpočty proměnných a ukázky programového kódu s podmínkami závisejícími na hodnotách proměnných. Šlo mi spíše o princip, jakým databáze funguje, a o zobecnění naprogramovaných funkcí formuláře.

Jak jsem uvedl ve SWOT analýze v kapitole 2.3.3.3, jedna z příležitostí společnosti je využívat kvalitnější a efektivnější informační technologie. Databáze svazků stroje právě tuto příležitost naplňuje a otvírá možnost pro budoucí vývoj. Ve společnosti se počítá s rozvojem této databáze o další komponenty stroje, jako například ventilátory nebo chladiče.

Díky databázi svazků stroje se proces konstrukce svazků stroje zkvalitnil, zrychlil a usnadnil. Databáze a funkce formuláře jsou dostatečně rychlé a nezatěžují nijak významně současné počítače. Protože je databáze vybavena uživatelským formulářem s naprogramovanými funkcemi, velmi lehce se obsluhuje a její ovládání je příjemné a snadné. To bylo mým hlavním cílem, vytvořit jednoduchou a uživatelsky příjemnou databázi. A dle mého názoru tato databáze takový cíl splňuje.

# Seznam literatury

## Knihy


1 HALVORSON, M. *Visual Basic 6.0 Professional Krok za krokem*. 2001. ISBN 8072264451

2 HELD,B. *Access VBA Velká kniha řešení*. 2006. ISBN 80-251-1112-1

3 HERNANDEZ, M. J. *Návrh databází*. 2006. ISBN 80-247-0900-7

4 KOCH, M. *Datové a funkční modelování*. 2006. ISBN 80-214-3252-7

5 *Mistrovství v Microsoft Visual Basicu 6.0 : Vyšší škola programovacích technik pro tvorbu firemních aplikací v prostředí klient-server*. 1999. Přel. J. Černý. ISBN 80-7226-205-X

6 MORKES, D. *Microsoft Office Access 2003*. 2004. ISBN 80-251-0179-7

7 MORKES, D. *Učebnice Visual Basicu 6.0*. 2000. ISBN 80-7226-312-9

8 OPPEL, A. *Databáze bez předchozích znalostí*. 2006. ISBN 80-251-1199-7

9 VIESCAS, J. *Mistrovství v Microsoft Access 2000 : Kompletní průvodce efektivního uživatele i tvůrce databází*. 2000. ISBN 80-7226-274-2

10 VOGLOVÁ, B. *Access v kanceláři - typické činnosti krok za krokem*. 2002. ISBN 80-247-0321-1

11 WALKENBACH, J. *Microsoft Excel 2000 a 2002 : programování ve VBA*. 2004. ISBN 80-7226-547-4


## Poznámky a konspekty z přednášek, konferencí, kurzů atd.


12 KOCH, M. Prezentace z předmětu Datové a funkční modelování. 2006

13 DYDOWICZ, P. Prezentace z předmětu Kancelářské aplikace. 2007


## Firemní materiály


14 *Výroční zpráva 2006-2007*. Siemens Electric Machines s.r.o. 2007. Výroční zpráva

## Internetové portály


15 *Databáze*.[online] [citováno 2009-02-20].

Dostupné z : http://www.adaptic.cz/znalosti/slovnicek/databaze.htm

16 *Návrh databáze*.[online] 2006. [cit 2009-02-25]

Dostupné z: http://www.abclinuxu.cz/blog/kacirstvi/2006/9/25/151490

17 *Navrhování databáze*. [online]

Dostupné z: http://office.microsoft.com/cs-cz/access/HP051891361029.aspx.

Poslední aktualizace 2006-11-20. [citováno 2009-02-27] Revize 13.0

18 *Siemens Electric Machines s.r.o.* [online] citováno 2009-01-20.

Dostupné z : http://www.semd.cz/main.php. Oficiální stránky společnosti

19 SKŘIVAN, J. *Databáze a jazyk SQL*.[online] 2000. [citováno 2009-03-05].

Dostupné z : http://interval.cz/clanky/databaze-a-jazyk-sql/

20 SLEZÁK,P. *VBA - jak začít*. [online]

Poslední aktualizace 2006-04-15. [citováno 2009-03-07]

Dostupné z: http://www.slezak-petr.cz/VBA/VBA_web.htm.

21 *Uložení obrázků v databázi*. [online] [citováno 2009-03-06].

Dostupné z: http://office.microsoft.com/cs-cz/access/HP052802251029.aspx

22 ZBÍRAL, D. *Visual Basic for Applications, VBScript*.[online] 2004.

[citováno 2008-03-09 ]. Dostupné z: http://www.david-zbiral.cz/vb.htm.

# Seznam obrázků

# Seznam použitých zkratek

MS – Microsoft

SQL - Structured Query Language

SŘBD - Systém řízení báze dat

VBA – Visual Basic for Applications

# Přílohy

**Seznam příloh:**

# Příloha 1 – Tabulka Svazky - struktura

| Název pole | Datový typ | |
|---|---|---|
| ID | automatické číslo | |
| Ulozeni | text | jak bylo uloženo do databáze (načtením z *.exp, uložením přes formulář) |
| Model | text | číslo modelu |
| Cislo_provedeni | text | číslo elektrického provedení z výpočtového listu |
| B1400_N | text | počet rozpěrek -1 vrtaná |
| B1400_S00 | text | šířka prvního vinutí s přídavkem |
| B1400_S01 | text | šířka druhého vinutí s přídavkem 0,16 |
| B1400_V | text | výška vinutí s přídavkem 0,25 |
| B1400_X | číslo | poloha - rozteč rozpěrek |
| B1400_Y | číslo | poloha - rozteč rozpěrek |
| B1400_Z | číslo | poloha - rozteč rozpěrek |
| B1478_S00 | text | šířka rozpěrky vinutí |
| B1478_V00 | text | výška rozpěrky vinutí |
| B1559_D | číslo | průměr tlumící tyče |
| B1559_D01 | číslo | poloha tlumících tyčí vůči středu |
| B1559_N | číslo | počet tlumících tyčí |
| B1559_U | číslo | úhel pro pole tlumících tyčí |
| B1559_Y | číslo | vzdálenost tlumících tyčí od průměru rotorového plechu |
| B1840_E00 | text | opěrná tyč 1 (1-ano 0-ne) |
| B1840_E01 | text | opěrná tyč 2 (1-ano 0-ne) |
| B1840_E02 | text | opěrná tyč 3 (1-ano 0-ne) |
| B1840_L | text | délka opěrné tyče |
| B1840_S00 | text | šířka opěrné tyče 1 |
| B1840_S01 | text | šířka opěrné tyče 2 |
| B1840_S02 | text | šířka opěrné tyče 3 |
| B1840_V00 | text | výška opěrné tyče 1 |
| B1840_V01 | text | výška opěrné tyče 2 |
| B1840_V02 | text | výška opěrné tyče 3 |
| B1840_X01 | text | vertikální umístění opěrné tyče 2 |
| B1840_X02 | text | vetikální umístění opěrné tyče 3 |
| B1840_XF01 | číslo | vertikální umístění opěrné tyče 2 (if) |
| B1840_XF02 | číslo | vertikální umístění opěrné tyče 3 (if) |
| B1840_Y00 | text | horizontální umístění opěrné tyče 1 |
| B1840_Y01 | text | horizontální umístění opěrné tyče 2 |
| B1840_Y02 | text | horizontální umístění opěrné tyče 3 |
| B1840_YF00 | číslo | horizontální umístění opěrné tyče 1 (if) |
| B1840_YF01 | číslo | horizontální umístění opěrné tyče 2 (if) |
| B1840_YF02 | číslo | horizontální umístění opěrné tyče 3 (if) |
| B1856_D01 | text | průměr svorníku |
| B1856_D02 | text | roztečná kružnice pro stahovací tyč |
| B1856_D03 | číslo | roztečná kružnice pro svorník |
| B1856_E00 | text | počet stahovacích tyčí na pól (0-jedna 1-dve) |
| B1856_F | číslo | rozteč mezi stahovacími tyčemi na 1 pólu (if) |
| B1856_L01 | text | délka svorníku |
| B1856_N00 | text | počet stahovacích tyčí |
| B1856_N01 | číslo | počet svorníků |
| B1856_U00 | číslo | úhel pro stahovací tyče |
| B1856_U01 | číslo | úhel pro pole svorníků |
| B1867_L00 | číslo | celková délka plechu tl. 0,65 |
| B1867_L01 | číslo | celková délka plechu tl. 0,80 |
| B1867_N00 | číslo | počet plechů 0,65 na svazek |
| B1867_N01 | číslo | počet plechů 0,80 na svazek |
| B1870_C | text | zkosení na pólu |
| B1870_D00 | text | průměr opracování čelní desky pro připojení k hřídeli |
| B1870_D01 | text | opracovaná středová díra |
| B1870_D02 | číslo | |
| B1870_D04 | číslo | čistý průměr rotorového plechu |
| B1870_D05 | číslo | čistý průměr statorového plechu |
| B1870_D60 | text | neopracovaná středová díra |
| B1870_E00 | text | zobrazení (1-ano 0-ne) |
| B1870_E01 | text | zobrazení (1-ano 0-ne) |
| B1870_F00 | text | vzorec pro pole 1 desek - počet (if) |
| B1870_F01 | text | vzorec pro pole 1 desek - počet (if) |
| B1870_M00 | číslo | vzduchová mezera |
| B1870_M01 | text | vzduchová mezera 2 |
| B1870_N00 | číslo | počet 1. krajnich desek na každé straně pólu |
| B1870_N01 | číslo | počet 2. krajních desek na jedne straně pólu |
| B1870_N02 | číslo | počet pólů |
| B1870_O | číslo | dělení tlumících tyčí |
| B1870_S00 | číslo | šířka nástavce |
| B1870_S01 | číslo | šířka pólu |
| B1870_T00 | číslo | tloušťka 1. krajní desky |

| | | | |
|---|---|---|---|
| | B1870_T01 | číslo | tloušťka 2.krajní desky |
| | B1870_U00 | číslo | |
| | B1870_V00 | text | výška nástavce |
| | B1870_V01 | text | výška pólu |
| | B1870_XF00 | text | tloušťka 1.krajní desky (if) |
| | B1870_XF01 | text | vzorec pro pole 1desek - distance (if) |
| | B1900_L | číslo | délka svazku pro celý stroj konstantní |
| | B2690_E | text | šířka ožehlené drážky |
| | B2690_F | text | výška ožehlené drážky |
| | B2690_H | text | čepy vlásenky na průměru |
| | B2690_J | text | délka ožehlení na dně drážky |
| | B2690_K | text | délka ožehlení u vzduchové mezery |
| | B2690_M | text | délka cívky izolované |
| | B2800_F | číslo | tangenciální posuv(celkový rozptyl), k ose je potřeba /2 |
| | B2800_FL | číslo | délka svazku |
| | B2800_FN0 | číslo | počet plechů na svazku 1 - orientační hodnota |
| | B2800_FN1 | číslo | počet plechů na svazku 2 - orientační hodnota |
| | B2800_FN2 | číslo | počet plechů na svazku 3 - orientační hodnota |
| | B2800_FN3 | číslo | orientační hodnota počtu plechů na 0,65 na paket |
| | B2800_L | číslo | hodnota délky svazku z výpočtového listu |
| | B2800_N00 | číslo | počet paketů |
| | B2800_N01 | číslo | počet drážek |
| | B2800_N02 | číslo | počet rybin |
| | B2800_N03 | číslo | počet kanálů - mezi pakety |
| | B2800_N04 | číslo | počet krajních plechů |
| | B2800_N05 | číslo | počet plechů na svazku 1 |
| | B2800_N06 | číslo | počet plechů na svazku 2 |
| | B2800_N07 | číslo | počet plechů na svazku 3 |
| | B2800_T00 | číslo | tloušťka svazku 1 |
| | B2800_T01 | číslo | tloušťka svazku 2 |
| | B2800_T02 | číslo | tloušťka svazku 3 |
| | B2800_X | číslo | vzdálenost mezi pakety |
| | B2896_D00 | číslo | průměr v drážkách |
| | B2896_D01 | číslo | malý průměr statorového plechu |
| | B2896_D02 | číslo | velký průměr statorového plechu |
| | B2896_H | text | hloubka rybinové drážky |
| | B2896_H01 | číslo | hloubka drážky |
| | B2896_S | číslo | šířka drážky u drážky T a šířka krčku u drážky S |
| | B2896_S00 | číslo | šířka drážky zajištění (pro drážku T) |
| | B2896_S01 | číslo | šířka rybinové drážky |
| | B2896_S02 | číslo | šířka drážky ( pro typ S) |
| | B2896_S03 | číslo | šířka dna drážky (pro typ S) |
| | B2896_T01 | číslo | síla statorového plechu |
| | B2896_T02 | číslo | síla krajního plechu |
| | B2896_U | číslo | úhel rybiny |
| | B2896_V | číslo | výška drážky od vnitřního průměru u drážky T a délka krčku u drážky S |
| | B2896_V00 | číslo | výška drážky zajištění u drážky T a délka přechodu u drážky S |
| | B2897_D00 | číslo | průměr statorového kruhu malý |
| | B2897_D01 | číslo | průměr statorového kruhu velký |
| | B2897_T | číslo | síla statorového kruhu |
| | Tvar_drazky | text | pomocná hodnota - tvar drážky (S nebo T) |
| | | | |

# Příloha 2 – Otevření výpočtového listu

*Otevření výpočtového listu pomocí File Dialogu (cesta k výpočtovému listu je nahrazena " …server...")*

```
 'zjisteni cesty pro vypoctak pomoci adresare
 With Application.FileDialog(msoFileDialogFilePicker)
  '.AllowMultiSelect = True
   .InitialFileName = "...server...\_vypoctaky\"  'vychozi slozka dialogu
   .Filters.Clear
   .Filters.Add "Excel files", "*.xls", 1
   .Filters.Add "All files", "*.*", 2
   .Title = "Vyberte prosim vypoctovy list"
   .Show

    Cesta_vypoctak = ""

    For Each vrtSelectedItem In .SelectedItems
      Cesta_vypoctak = vrtSelectedItem 'vypise cestu ke zvolenemu souboru
    Next vrtSelectedItem


 End With

    If Cesta_vypoctak = "" Then  'pokud nezadame vypoctak, procedura se ukonci
     GoTo konec
    End If


'zjisteni jaky je vypoctak :
If InStr(Cesta_vypoctak, "profile") <> 0 Then   'vypoctak pro profilovy drat
  Typ_vypoctaku = "profilovy"
  vyber_drazka.value = 1
  Form_Svazky.tl_drazkaT_GotFocus
 Else    'vypoctak pro kulaty drat
  Typ_vypoctaku = "kulaty"
  vyber_drazka.value = 2
  Form_Svazky.tl_drazkaS_GotFocus
End If
```

*Otevření z tabulky „Nastavení"*

```
 Set fso = CreateObject("Scripting.FileSystemObject")
 Set dbs = CurrentDb  'prirazeni k soucastne otevrene databazi
 Set rst = dbs.OpenRecordset("select * from Nastaveni")  'vybere hodnoty poli v nastaveni

     Cesta_vypoctak_profil = ""

     Cesta_vypoctak_profil = rst.Fields!Cesta_vypoctak_profil.value

     If fso.FileExists(Cesta_vypoctak_profil) = False Then
       MsgBox "Soubor s vypoctovym listem nebyl nalezen", vbOKOnly + vbCritical, "Nenalezeno"
       GoTo konec
     End If

     If Cesta_vypoctak_profil = "" Then  'pokud nenalezne vypoctak, procedura se ukonci
      GoTo konec
     End If
```

# Příloha 3 – Načtení hodnot elektrického provedení

```
 Set MyXL = GetObject(Cesta_vypoctak_profil)    ' priradi vypoctak promenne

 Dim Radek, Radek1, Radek2 As Long      'aktualni radek
 Radek = 1  'sem priradit 1.hodnotu
 Radek1 = 5
 Radek2 = 10

'cyklus pro nacitani poli
Do Until (MyXL.Sheets(1).Range("A" & Radek) = KonecnyRadek) _
          And (MyXL.Sheets(1).Range("A" & Radek1) = KonecnyRadek) _
          And (MyXL.Sheets(1).Range("A" & Radek2) = KonecnyRadek)

   hodnota = MyXL.Sheets(1).Range("A" & Radek)
   ReDim Preserve provedeniP_array(Radek)
   provedeniP_array(Radek) = hodnota

  Radek = Radek + 1
  Radek1 = Radek + 5
  Radek2 = Radek + 10

Loop

Radek_profil = Radek - 1
```

---

```
  Public Sub Fce_Reload()


   If Pole_list.ListCount <> 0 Then  'pokud neni combobox prazdny
    X = 0
    Do While X < Me.Pole_list.ListCount   'vymaze stare hodnoty ComboBoxu
      Pole_list.RemoveItem (X)
    Loop


  On Error Resume Next

    Pole_list.value = Null
    Me.Pole_list.ListIndex = 0
   End If

  On Error GoTo Err_Fce_Reload


  'podle typu dratu
   Select Case Me.Vyber_typ_dratu.value
     Case 1 'profilovy drat
         Typ_vypoctaku = "profilovy"
         vyber_drazka.value = 1  'zatrzeni typu drazky
         For i = 1 To Radek_profil
           Pole_list.AddItem provedeniP_array(i)
         Next i
         Cesta_vypoctak = Cesta_vypoctak_profil

     Case 2    'vypoctak pro kulaty drat
       Typ_vypoctaku = "kulaty"
       vyber_drazka.value = 2
       For i = 1 To Radek_kulaty
         Pole_list.AddItem provedeniK_array(i)
       Next i
       Cesta_vypoctak = Cesta_vypoctak_kulaty

     Case Else 'pokud neni zatrzeno, nedelej nic
       Exit Sub

   End Select
```

# Příloha 4 – Načtení dat z Excelu do polí formuláře

```
 Dim MyXL As Object     ' promenna pro Excel.

 Set MyXL = GetObject(Cesta_vypoctak) 'cesta k excelu se nacita z inputboxu od uzivatele


 Radek = Form_Svazky.Pole_list.ListIndex + 1 'jaky radek beru z vypoctaku

 If Radek = 0 Then Exit Sub
```

```
'nacitani z excelu do poli formulare - drazka - pomocne hodnoty promennych

If vyber_drazka.value = "1" Then  'pokud je drazka typu T
 Set S_hloubka_drazky2 = MyXL.worksheets(1).cells(Radek, 25)   'hloubka drazky je pro oba typy stejna
 Set S_A2 = MyXL.worksheets(1).cells(Radek, 21) 'sirka drazky
 Set S_Avz2 = MyXL.worksheets(1).cells(Radek, 21) 'prirazeni sirky drazky
 S_Avz3 = S_Avz2 + 3.2  'sirka drazky zajisteni
 Set S_Vyska_drazky_zajisteni2 = MyXL.worksheets(1).cells(Radek, 22)
 Set S_Vyska_drazky_od_vn_prum2 = MyXL.worksheets(1).cells(Radek, 23)

 Me.Text_hloubka_drazky.value = S_hloubka_drazky2
 Me.Text_drazka_k.value = S_Vyska_drazky_od_vn_prum2
 Me.Text_drazka_h0.value = S_Vyska_drazky_zajisteni2
 Me.Text_drazka_b0 = S_A2
 Me.Text_drazka_a = S_Avz3

Else   'pukud je drazka S
 Set S_hloubka_drazky2 = MyXL.worksheets(1).cells(Radek, 24)   'hloubka drazky je pro oba typy stejna
 Set S_dr_sirka = MyXL.worksheets(1).cells(Radek, 20) 'b1-sirka drazky
 Set S_dr_sirka_dna = MyXL.worksheets(1).cells(Radek, 21) 'b2-sirka dna drazky
 Set S_dr_Skrcku = MyXL.worksheets(1).cells(Radek, 22) 'b0 - sirka krcku
 Set S_dr_Lkrcku = MyXL.worksheets(1).cells(Radek, 23) 'h0 - delka krcku


 Me.Text_hloubka_drazky.value = S_hloubka_drazky2
 Me.Text_drazka_k.value = 0  'implicitne bude delka prechodu rovna 0
 Me.Text_drazka_b1.value = S_dr_sirka
 Me.Text_drazka_b2.value = S_dr_sirka_dna
 Me.Text_drazka_b0.value = S_dr_Skrcku
 Me.Text_drazka_h0.value = S_dr_Lkrcku


End If
```

# Příloha 5 – Upozornění na existenci v databázi

```
'pri zmene upozorni pokud je v databazi
Prom_cislo_provedeni = Pole_list.value

Set dbs = CurrentDb  'prirazeni k soucastne otevrene databazi
Set rst2 = dbs.OpenRecordset("select * from Svazky ") 'otevre 1. zaznam


Do While Not rst2.EOF 'dokud je dalsi zaznam

  If rst2.Fields!Cislo_provedeni.value = Prom_cislo_provedeni Then 'pokud se shoduje cislo s databazi

    If MsgBox("Svazek se stejnym cislem provedeni jiz existuje, chcete ho nacist do tohoto formulare?", _
             vbYesNo + vbExclamation, "Upozorneni : svazek existuje v databazi") = vbYes Then
      existence_svazku = 1
      idd = rst2.Fields!ID.value 'zjisti id radku
      Call Modul_funkce.nacti_vzor(idd) 'spusti nacitaci funkci
      Form_Svazky.SetFocus
      Exit Sub

    Else
    Exit Sub

    End If
  End If

rst2.MoveNext

Loop

rst2.Close
dbs.Close
```

# Příloha 6 – Ukázky částí procedury „nacti_excel“

```
    Set MyXL = GetObject(Cesta_vypoctak)

     Radek = Form_Svazky.Pole_list.ListIndex + 1  'jaky radek beru z vypoctaku

'rozlisuji nacitani z excelu podle dratu na profilovy a kruhovy vypoctak : kazdy ma jinak sloupce
 If Typ_vypoctaku = "profilovy" Then

    Set R_deleni1 = MyXL.worksheets(1).cells(Radek, 81) 'deleni tlumicich tyci

      PLomitka = InStr(R_deleni1, "/") 'pozice lomitka
      Citatel = Mid(R_deleni1, 1, PLomitka - 1)
      Jmenovatel = Mid(R_deleni1, PLomitka + 1)

      R_deleni = Val(Jmenovatel)              'Val(Citatel) / Val(Jmenovatel)


    Set R_tlum_tyc_D = MyXL.worksheets(1).cells(Radek, 78) 'prumer tlumicich tyci
    Set R_tlum_tyc_n = MyXL.worksheets(1).cells(Radek, 77) 'pocet tlumicich tyci
    Set R_vzdal_tlum_tyci = MyXL.worksheets(1).cells(Radek, 79) 'vzdalenost tlumicich tyci
```

---

```
'pro vypocet vnitrniho prumeru stahovaciho kruhu
S_pridavek_stah_kruhu = 2 * (Me.Text_pridavek_stah_kruhu.value) 'nasobim 2 protoze beru na cely prumer

'nacteni opernych tyci
delka_ot = Val(Me.TextLS.value) + Val(Me.TextPresah_ot.value)  'delka operne tyce

sirka_ot1 = Me.sirka_ot1.value 'sirka operne tyce 1
sirka_ot2 = Me.sirka_ot2.value
sirka_ot3 = Me.sirka_ot3.value
```

---

```
'Generovani dat podle prumeru hridele
 PrHr = PrumHrideleSeznam.value
 Select Case PrHr

   Case "241"
     hr_cela = 255 'Obrobeni cela
     hr_dplech = 240 'Prumer na plechu pred obrobenim
     hr_dsvor = 290 'Roztecna kruznice pro svornik
     dsvor = 20 'prumer svorniku
```

# Příloha 7 – Funkce tlačítka „Generuj řídící soubor“

```
Public Sub cmb_Generuj_Click() 'generuje data do souboru *.exp

Set fso = CreateObject("Scripting.FileSystemObject") 'vytvori objekt

Call nacti_excel


'nacteni cesty pro ulozeni od uzivatele
'Zapisovani parametru do souboru s expresnama

cesta = Application.CurrentProject.Path
Jmeno_souboru = InputBox("Soubory budou ulozeny do adresare: " & Chr(13) & cesta & "\expresny\" _
                & Chr(13) & Chr(13) & "Zadejte název souboru pro expresny (bez pripony)", , "svazek")

Cesta_generuj_exp = cesta + "\expresny\" + Jmeno_souboru + ".exp"

If Jmeno_souboru = "" Then Exit Sub

Do While fso.FileExists(Cesta_generuj_exp) = True 'pokud soubor existuje
  If MsgBox("Soubor existuje. Chcete ho nahradit?", vbYesNo + vbExclamation, "Soubor existuje") = vbYes Then
    Kill Cesta_generuj_exp
  Else
    Jmeno_souboru = InputBox("Soubory budou ulozeny do adresare: " & Chr(13) & cesta & "\expresny\" _
                    & Chr(13) & Chr(13) & "Zadejte název souboru pro expresny (bez pripony)", , "svazek")

    Cesta_generuj_exp = cesta + "\expresny\" + Jmeno_souboru + ".exp"
  End If
Loop

 If Not fso.FolderExists(cesta & "\expresny\") Then
    MkDir cesta & "\expresny"  'pokud neni adresar vytvori ho to
 End If
```

---

```
Open Cesta_generuj_exp For Output As #1    ' Otevreni souboru pro zapis

Print #1, "B1870_D04=" + Str(promB2896_D01 - promB1870_M00) + " //cisty prumer rotoroveho plechu "
Print #1, "B1870_D05=" + Str(promB2896_D01 - (2 * R_vzd_mezera_2)) + " //cisty prumer statoroveho plechu"

Print #1, "B1870_N02=" + Str(R_pol_N) + "//pocet polu"
Print #1, "B1870_O=" + Str(R_deleni) + "//deleni tlumicich tyci"

If vyber_drazka.value = "1" Then 'pokud je drazka T
 Print #1, "B2896_D00=" + Str(promB2896_D00) + " //Prumer v drazkach"
Else  'pokud je drazka typu S
  'jestli je drazka S tak pricti do prumeru i delku krcku a prechodu
  Print #1, "B2896_D00=" + Str(promB2896_D00 + 2 * (S_dr_Lkrcku + S_dr_Lprechodu)) + " //Prumer v drazkach"
End If


Print #1, "B2897_D01=" + Str(S_Dv - 9) + " //Prumer stah. kruhu velky" ' -9 mm od vnejsiho prumeru stat plechu
Print #1, "B2897_T= 20 //Sila stah kruhu"

Close #1

MsgBox "Vygenerovano, soubor se automaticky ulozil do databaze"

'otevre vygenerovany soubor v poznamkovem bloku
 Set ws = CreateObject("Wscript.Shell")
 ws.Run ("notepad " & Cesta_generuj_exp)

Call cmb_Ulozit_Click  'zavola funkci pro ulozeni do databaze
```

# Příloha 8 – Funkce tlačítka „Uložit do databáze“

```
Set dbs = CurrentDb  'prirazeni k soucastne otevrene databazi
Set rst3 = dbs.OpenRecordset("select * from Svazky ")  'otevrit  1. zaznam

Prom_cislo_provedeni = Pole_list.value


Do While Not rst3.EOF 'dokud je dalsi zaznam

  If rst3.Fields!Cislo_provedeni.value = Prom_cislo_provedeni Then 'pokud se shoduje cislo s c. v databazi
   existence_svazku = 1
  End If

 rst3.MoveNext 'presun na dalsi zaznam

Loop


If existence_svazku = 1 Then 'pokud je v databazi svazek se stejnym cislem el. provedeni
 On Error Resume Next
  prom_model = Text_model.value
  Prom_cislo_provedeni = "'" + Pole_list.value + "'"   'pridavam apostrofy kvuli syntaxi dotazu

  'otevre zaznamy se stejnym elektrickym provedenim
  Set rst2 = dbs.OpenRecordset("select * from Svazky where Cislo_provedeni=" & Prom_cislo_provedeni)
   a = rst2.Fields!ID.value

  Do While Not rst2.EOF 'dokud je dalsi zaznam
   a = rst2.Fields!ID.value

    If rst2.Fields!Model.value = prom_model Then 'pokud je v zaznamu stejny model
      If MsgBox("Svazek se stejnym CISLEM PROVEDENI i stejnym MODELEM jiz existuje. Chcete ho prepsat?", _
            vbYesNo + vbInformation, "Upozorneni : stejny svazek existuje v databazi") = vbYes Then
        id_model = rst2.Fields!ID.value
        Set rst = dbs.OpenRecordset("select * from Svazky where ID=" & id_model) 'otevre stejny zaznam
        rst.edit 'otevre zaznam pro upravy
        GoTo konec_do
      Else
       Exit Sub
      End If

    Else
      stejny = 0
    End If

   rst2.MoveNext

  Loop

If stejny = 0 Then  'pokud to nenajde stejny model, vytvori novy zaznam
   Set rst = dbs.OpenRecordset("Svazky")  'otevrit tabulku
   rst.AddNew  'pridat novy zaznam
End If

konec_do:
  rst2.Close

Else  'existence_svazku <> 0 - v DB neni svazek se stejnym cisle provedeni

 Set rst = dbs.OpenRecordset("Svazky")  'otevrit tabulku
 rst.AddNew  'pridat novy zaznam

End If
```

```
Call nacti_excel 'zavola proceduru pro pocitani promennych

'naplneni tabulky hodnotami promennych
'komentare k jednotlivym promennym jsou ve funkci cmb_Generuj_Click() nebo v deklaracni casti
rst![Ulozeni] = "ulozenim pres formular"
rst![Model] = Model
rst![Cislo_provedeni] = Pole_list.value

rst![B1559_D] = promB1559_D
rst![B1559_DO1] = promB1559_DO1
rst![B1559_N] = R_tlum_tyc_n
rst![B1559_U] = ((promB1559_N - 1) / 2) * (360 / R_deleni)
rst![B1559_Y] = R_vzdal_tlum_tyci
rst![B1840_EOO] = ot1_existence
rst![B1840_EO1] = ot2_existence
```

# Příloha 9 – Uložení dat z expressionu do databáze

*Cesta pomocného souboru a otevření soboru s expressiony pomocí File Dialogu*

```
Cesta_nacti_exp_sv = "H:\temp.txt" ' pomocny textak do ktereho se uklada vybrana cast soboru

'otevira expresen:
 With Application.FileDialog(msoFileDialogFilePicker)
     .Filters.Clear
     .Filters.Add "Expression text", "*.exp; *.txt", 1
     .Filters.Add "All files", "*.*", 2
     .Title = "Vyberte prosim soubor expressionu, ktery chcete ulozit do databaze"
     .Show

   Cesta_nacti_exp2_sv = ""

   For Each vrtSelectedItem In .SelectedItems
      Cesta_nacti_exp2_sv = vrtSelectedItem 'vypise cestu ke zvolenemu souboru
   Next vrtSelectedItem

 End With
```

*Otevření souboru pro vstup, vytvoření pomocného souboru a načtení názvu modelu*

```
Open Cesta_nacti_exp2_sv For Input As #1    ' Otevrit soubor pro vstup

Set fso = CreateObject("Scripting.FileSystemObject") 'vytvori fso
Set TF = fso.CreateTextFile(Cesta_nacti_exp_sv) 'vytvori pomocny textovy soubor pro ukladani


  Do While Najdi_nazev = 0  'opakuj dokud se "Date" nenachazi v radku (za Date je cesta souboru)
    Line Input #1, strline
    Najdi_nazev = InStr(strline, "Date")
  Loop

  Line Input #1, strline  'nacte radek s cestou

  Text = 1
  Do While Not Text = 0
    poz = InStr(strline, "\")  'hleda lomitko v textu
    Nazev = Mid(strline, poz + 1)  'nacte vse za lomitkem
    Text = InStr(Nazev, "\")  'hleda lomitko v textu nazvu
    strline = Nazev
  Loop

  'MsgBox "Nazev souboru je: " & nazev

  Cislo_modelu = Left(Nazev, 7)
  Hledany_model_sv = "!" + Cislo_modelu + "_1900_WAVE_Svazky_prt"
  Hledany_model_sv_bez = Mid(Hledany_model_sv, 2) 'bez vykricniku
```

*Zjištění jestli model je v databázi*

```
Set dbs = CurrentDb  'prirazeni k soucastne otevrene databazi
Set rst3 = dbs.OpenRecordset("select * from Svazky ") 'otevre 1. zaznam

Do While Not rst3.EOF 'dokud je dalsi zaznam
    If rst3.Fields!Model.value = Hledany_model_sv_bez Then 'pokud model je uz v databazi
      exist = 1
      id_exist = rst3.Fields!ID.value
      GoTo konec_exist
    Else
      exist = 0
    End If

  rst3.MoveNext
 Loop
```

*Vyhledávání expressionů pro svazky, zapisování do pomocného souboru (ukončovací Loop pro While je až na konci procedury)*

```
zacatek:
  Do While Not EOF(1)
    Line Input #1, strline
    Najdi_model = InStr(strline, Hledany_model_sv)
    If Najdi_model = 1 Then GoTo pokracuj
  Loop

pokracuj:
  If Najdi_model = 0 Then
     Hledany_model_sv = InputBox("Nenasel jsem tuto komponentu : " & Hledany_model_sv _
                   & Chr(13) & "Zadejte prosim nazev pouzite komponenty!", "Nenalezeno", Hledany_model_sv)
     If Hledany_model_sv = "" Then Exit Sub
      Seek #1, 1 'vrati se na 1.znak v dokumentu
     GoTo zacatek
  Else


  'Line Input #1, strline  'preskakuji radek
  TF.Writeline (Hledany_model_sv)

  Do While Not konec_modelu = "!"
    Line Input #1, strline
    TF.Writeline (strline)
    konec_modelu = Left(strline, 1)
  Loop

End If
```

*Vyhledání parametru, komentáře a hodnoty parametru v řádku*

```
Do While Not EOF(1)     ' Opakuj dokud neni konec souboru
   Input #1, strline


     PEqu = InStr(strline, "=") 'pozice kde je rovnase v radku

     If PEqu > 0 Then
       PKom = InStr(strline, "//") 'pozice kde zacina komentar
       CoExpr = Mid(strline, 1, PEqu - 1) 'parametr

       If PKom > 0 Then
         CoValue = Trim(Mid(strline, PEqu + 1, PKom - PEqu - 1))    'hodnota parametru
         CoKom = Mid(strline, PKom + 2)
         PHodn = InStr(CoKom, "//")
         PHodnoty = PKom + PHodn + 2 'pozice kde zacina hodnota za //
         CoHodn = Val(Mid(strline, PHodnoty + 1))
       Else
           CoValue = Trim(Mid(strline, PEqu + 1))
           CoKom = "komentar nedefinovany"
       End If

     End If
```

*Porovnávání hodnot expressionů a databáze –když  hodnota v databázi je číslo a expression je text*

```
''pokud je hodnota_db cislo a hodnota expressionu text
If IsNumeric(hodnota_db) = True And IsNumeric(CoValue) = False Then
  If hodnota_db = Val(Trim(CoValue)) Then 'pokud je hodnota parametru stejna jako v databazi:
   Else
      If MsgBox("Hodnota parametru " & Chr(13) & CoExpr & Chr(13) & _
               "z expressionu ( který znamená " & CoKom & ") je : " & Chr(13) & CoValue & Chr(13) _
               & "Hodnota tohoto parametru v databazi je:  " & hodnota_db & Chr(13) & Chr(13) & _
               "Chcete hodnotu v databazi nahradit novou?", vbYesNo + vbDefaultButton2 + vbQuestion, _
               "Nahradit hodnotu parametru?") = vbYes Then
        'hodnota se nahradi
        Else
        CoValue = Trim(Str(hodnota_db))
      End If
  End If

End If ' jestli je hodnota_db numeric a CoValue string
```

*Pokud není model v databázi, vytvoří nový záznam a přiřadí ID*

```
Else  '****************************pokud neni model v databazi

    Set rst = dbs.OpenRecordset("Svazky")  'otevrit tabulku
    rst.AddNew  'pridat novy zaznam
    id_novy = rst.Fields!ID.value
    rst.Update
```

*Otevření záznamu a roztřídění do datové tabulky*

```
Set dbs = CurrentDb  'prirazeni k soucastne otevrene databazi

If exist = 1 Then 'pokud model existuje v databazi
   'otevrit tabulku na pozadovanem zaznamu
  Set rst = dbs.OpenRecordset("select * from Svazky where ID=" & id_exist)
  rst.edit
 Else
  'otevrit tabulku na novem zaznamu
  Set rst = dbs.OpenRecordset("select * from Svazky where ID=" & id_novy)
  rst.edit
End If

'davam CoValue kdyz je promenna textova (pole ve databzi je typu text jinak CoVal

'------------------------------- select pro rozpoznani parametru z expresnu a ukladani do databaze

        Select Case CoExpr
         Case "B1400_N"
         'CoVal = Val(CoValue)
         rst![B1400_N] = CoValue

         Case "B1400_S00"
         'CoVal = Val(CoValue)
         rst![B1400_S00] = CoValue

               .

               .

               .

         Case "B2897_D01"
         CoVal = Val(CoValue)
         rst![B2897_D01] = CoVal

         Case "B2897_T"
         CoVal = Val(CoValue)
         rst![B2897_T] = CoVal


        End Select
'-----------------------------------konec selectu
   rst![Ulozeni] = "nacteno z *.exp"
   rst![Model] = Hledany_model_sv_bez
   rst![Cislo_provedeni] = "neni - nacteno z *.exp"


rst.Update
rst.Close
```

# Příloha 10 – Část procedury nacti_vzor(idd)

```
Public Sub nacti_vzor(idd)
'nacita data - vzor -  z databaze do formulare (Fomr_Svazky_vzor)

Set dbs = CurrentDb  'prirazeni k soucastne otevrene databazi
'otevrit tabulku na pozadovanem zaznamu
Set rst = dbs.OpenRecordset("select * from Svazky where ID=" & idd)

'otevirani prislusnych vypoctaku podle typu drazky
If rst.Fields!Tvar_drazky.value = "T" Then 'pokud bude drazka tvaru T
  Form_Svazky.Vyber_typ_dratu.value = 1 'profilovy
  Cesta_vypoctak = Form_Svazky.Cesta_vypoctak_profil
End If
If rst.Fields!Tvar_drazky.value = "S" Then 'pokud bude drazka tvaru S
  Form_Svazky.Vyber_typ_dratu.value = 2 'kruhovy
  Cesta_vypoctak = Form_Svazky.Cesta_vypoctak_kulaty
End If

Call Form_Svazky.Fce_Reload

'priradi hodnotu comboboxu
Form_Svazky.Pole_list.value = rst.Fields!Cislo_provedeni.value
Call Form_Svazky.Nacti_data  'zavola proceduru nacitaci excel

'upravuje hodnoty formulare podle databaze
Form_Svazky.Text_model.value = rst.Fields!Model.value 'model
Form_Svazky.TextLS.value = rst.Fields!B2800_L.value 'delka stat. svazku
Form_Svazky.TextNK.value = rst.Fields!B2800_NO3.value 'pocet kanalu
Form_Svazky.TextLK.value = rst.Fields!B2800_X.value 'velikost kanalu
Form_Svazky.TextNkrpl.value = rst.Fields!B2800_NO4.value 'pocet krajnich plechu
Form_Svazky.TextSvazek1.value = rst.Fields!B2800_NO5.value 'delka svazku 1
Form_Svazky.TextSvazek2.value = rst.Fields!B2800_NO6.value 'delka svazku 2
Form_Svazky.TextSvazek3.value = rst.Fields!B2800_NO7.value 'delka svazku 3
```

# Příloha 11 – Vyhledávání v databázi

*Funkce tlačítka Vyhledat – nastavení proměnných a vyhledávání podle čísla modelu*

```
Private Sub cmb_Vyhledat_Click()
i = 0 'pocet zaznamu
akt = 0 'poradove cislo aktualniho zaznamu
Set dbs = CurrentDb  'prirazeni k soucastne otevrene databazi
'srovnavani s databazi podle poli vedle zatrzitek
prom_model_a = "'" + Me.Text_model.value + "'"   'pridavam apostrofy kvuli syntaxi dotazu
prom_c_provedeni_a = "'" + Me.Text_c_provedeni.value + "'"

prom_model = Me.Text_model.value
prom_c_provedeni = Me.Text_c_provedeni.value


Set rst1 = dbs.OpenRecordset("select * from Svazky")

Select Case Me.vyber_vyhledat.value  'jaka hodnota vyhledavani je zatrzena

  Case 1 'podle modelu
    If IsNull(prom_model) = True Then GoTo nenalezeno
    Do While Not rst1.EOF
     If rst1.Fields!Model.value = prom_model Then
      prom_ID_vyhl = rst1.Fields!ID.value
      'naplneni pole s vyhledanymi zaznamy:
       i = i + 1 'poradi zaznamu
       uloz = i & "," & prom_ID_vyhl 'ukladany text do pole
       ReDim Preserve vyhl_array(i) 'definuje dalsi prvek pole
       vyhl_array(i) = uloz
     End If
     rst1.MoveNext
    Loop
    'otevre nalezene zaznamy
    Set rst2 = dbs.OpenRecordset("select * from Svazky where Model=" & prom_model_a)
```

*Celý kód tlačítka Další*

```
Public Sub cmb_Dalsi_Click() 'funkce tlacitka Dalsi
On Error GoTo Err_cmb_Dalsi_Click
akt = akt + 1 'cislo aktualniho zaznamu)

'pokud je ciselnik akt vetsi nez pocet zaznamu, nastav ho na pocet zaznamu a ukonci proceduru
If akt > i Then
  akt = i
  Exit Sub
End If

txt_pole = vyhl_array(akt)
poz_carka = InStr(txt_pole, ",")
zaznam = Mid(txt_pole, 1, Len(txt_pole) - poz_carka - 1)
ID_nove = Val(Mid(txt_pole, poz_carka + 1))


Set dbs = CurrentDb
Set rst_n = dbs.OpenRecordset("select * from Svazky where ID=" & ID_nove) 'otevre hledany zaznam

  Me.Text_model_vyhl.value = rst_n.Fields!Model.value 'model
  Me.Text_c_provedeni_vyhl.value = rst_n.Fields!Cislo_provedeni.value 'cislo provedeni
  Me.Text_delka_svazku_vyhl.value = rst_n.Fields!B2800_L.value 'delka svazku z vypoctoveho listu
  Me.Text_ID_vyhl.value = rst_n.Fields!ID.value 'ID


rst_n.MoveLast
rst_n.Close
dbs.Close

Me.Text_aktual.value = akt & "." 'kolikaty zaznam je zobrazen

Exit_cmb_Dalsi_Click:
    Exit Sub

Err_cmb_Dalsi_Click:
    MsgBox Err.Description
    Resume Exit_cmb_Dalsi_Click


End Sub
```

*Tlačítko předchozí – úvod kódu*

```
Private Sub cmb_Predchozi_Click() 'funkce tlacitka Predchozi
On Error GoTo Err_cmb_Predchozi_Click

akt = akt - 1 'cislo aktualniho zaznamu

'pokud je ciselnik akt mensi nez pocet zaznamu, nastav ho na prvni zaznam a ukonci proceduru
If akt < 1 Then
  akt = 1
  Exit Sub
End If
```