Aplikace pro vyhodnocení rentability výroby a obchodu

Application for assessment of profitability production and bushiness

Kafka Pavel

Bakalářská práce
2008

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2007/2008

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Ing. Pavel KAFKA**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Aplikace pro vyhodnocení rentability výroby a obchodu**

Zásady pro vypracování:

1. **Analýza požadavků organizace.**
2. **Vytvoření a konfigurace databáze:**
   - **volba typu databáze (relační, analytická, objektová)**
   - **konfigurace databáze (nastavení, zabezpečení, záloha, obnova databáze)**
   - **návrh uložení dat (tabulky, indexy, uložené funkce a procedury, pohledy. . .)**
   - **aktualizace dat (načítání a aktualizace dat z IS)**
   - **kontrola integrity dat**
   - **software pro databázi (Microsoft SQL, ORACLE, MySQL, PostgreSQL)**
3. **Návrh a vytvoření uživatelského rozhraní.**
4. **Zhodnocení aplikace:**
   - **porovnání aplikace s aktuálním informačním systémem**
   - **náklady na vytvoření aplikace (software pro vývoj, náklady na vývoj aplikace)**
   - **provozní náklady (hardware, údržba aplikace)**
   - **výhody a nevýhody aplikace**

Rozsah práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **LACKO, L. – Business Inteligence v SQL Serveru 2005. Brno: Computer Press, 2003. s. 390.**
2. **LACKO, L. – Oracle : Správa, programování a použití databázového systému. Brno: Computer Press, 2003. s. 480.**
3. **SACK, J. – Velká kniha T-SQL & SQL Server 2005. Brno: Zoner Press, 2007. s. 864.**
4. **PRICE, J. – C – programování databází. Grada publishing, 2005. s. 624.**

Vedoucí bakalářské práce: **doc. Ing. Zdenka Prokopová, CSc.**
Ústav aplikované informatiky

Datum zadání bakalářské práce: **20. února 2008**

Termín odevzdání bakalářské práce: **5. května 2008**

Ve Zlíně dne 20. února 2008

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ Fakulta aplikované informatiky

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Bakalářská práce popisuje vznik aplikace pro doplnění informačního systému. Důvody vzniku aplikace, požadavky na její funkcionalitu a samotný vývoj. Tvorba aplikace probíhá ve vývojovém prostředí Microsoft Visual Studio 2005 s nadstavbovými nástroji firmy DevExpress. Databáze pro aplikaci je postavena na databázovém serveru Microsoft SQL Server 2005.

Klíčová slova: informační systém, SQL Server 2005, Visual studio 2005, DevExpress

ABSTRACT

The bachelor thesis is describing following subjects: the creation of the application of supplement information system, the reasons of its creation, the functionality requirements and development of the application. Creation of the application is proceeding at Microsoft Visual Studio 2005 by using of control suit, made by DevExpress. Application database is building on database server Microsoft SQL Server 2005.

Keywords: information system, SQL Server 2005, Visual studio 2005, DevExpress

Jednoduchá aplikace je fikce, která skončí analýzou problému.

Prohlašuji, že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně 15.5.2008 ……..

Podpis diplomanta

OBSAH

ÚVOD

Informační systémy slouží pro uložení, zpracování, vyhodnocení a poskytování informací a dat, které jsme nějakým způsobem získali. Data lze získat například při výrobě, při obchodních jednáních nebo už je máme uložena fyzicky v úplně jiném typu informačního systému. Informační systémy existovaly už před příchodem počítačů třeba v papírové podobě. Příkladem tohoto typu informačního systému může být kartotéka nebo účetní knihy. Pokud chcete získat data z tohoto typu informačního systému, nebude to hned a už vůbec ne přehledně. Tím se odlišuje většina dnešních informačních systémů založených na počítačích od svých předchůdců. Zkuste si představit, jak dnes vyhledáváte tisíc položek v kartotéce a zapisujete je třeba do tabulky. Pokud však namítnete, že to je dnes už nemožné, pak si vzpomeňte, kdy jste naposled vypisovali údaje „z počítače“, abyste je mohli použít k vámi zamýšlené operaci. Najednou to není až tak dávno. Pokud stejnou činnost opakujete vícekrát, pak je už jen na vaší trpělivosti, kdy požádáte informatiky nebo správce informačních systémů o nápravu tohoto stavu. Pak už záleží jen na počtu podobných žádostí a možnosti rozšíření informačního systému. Pokud je cesta přes informační systém složitá, musí se obejít pomocí aplikace, která dočasně nebo trvale nahradí některé funkce informačního systému. Právě vývoj aplikace tohoto typu je popsán v následující práci.

Bakalářská práce se zabývá doplněním stávajícího informačního systému podle požadavků z různých úrovní firmy, počínaje od výroby přes obchod až po manažery. Teoretická část naznačuje možnosti některých dnešních nástrojů. Práce však neslouží pro srovnání nástrojů, proto nejsou možnosti nástrojů nijak rozváděny ani porovnávány. V praxi je možnost výběru technologie a nástrojů značně limitována existující koncepcí firemního rozvoje. Výměna technologií bez dalších důsledků je možná pouze v okrajových záležitostech bez dopadu na chod firmy nebo podniku.

Praktická část prochází jednotlivými etapami vývoje aplikace, která doplňuje dočasně informační systém. Začíná analýzou požadavků jednotlivých typů uživatelů. Popisuje problémy při vývoji, jejich možnostmi řešení a zvolenými postupy.

I. TEORETICKÁ ČÁST

1 INFROMAČNÍ SYSTÉMY

1.1 Podnikové informační systémy

ERP slouží pro většinu činností firem a podniků. Zejména jde o prodej, nákup, výrobu, sklady, účetnictví, controlling, ale také majetek firmy a její zaměstnance. Informační systémy dnes již patří mezi základní vybavení firem všech velikostí. Výběr systémů je až nezvykle bohatý a o to těžší může být zvolit si ten pravý. Jejich kvalitu nelze posuzovat podle schopností prodejců daného systému, i když hodně o daném systému naznačuje. Čím více implementací, tím bude daný informační systém variabilnější a robustnější. Pro malé firmy s několika desítkami nebo stovkami dokladů bude k plné spokojenosti vyhovovat bez úprav kterýkoliv z nich. Ve větších firmách nebo podnicích si zavedení informačního systému vyžaduje úsilí a finanční prostředky, které je nutné vynaložit, aby daný software předvedl co nejvíce ze svých možností. Žádný informační systém však není a ani nebude dokonalý. Mají svá omezení, o nichž se při implementaci patrně dovíte velmi málo. Narazíte na ně teprve při ostrém provozu. Pak už je ale jejich odstranění výrazně náročnější a často i nemožné. Cílem této práce není sepsat a zkritizovat nedostatky informačního systému, ale hledat cesty k jejich odstranění a doplnění požadované funkčnosti buď přímo do informačního systému, nebo pomocí aplikace, která bude nadstavbou daného systému.

1.2 Požadavky na informační systémy

Informační systém v podniku nebo firmě nepředstavuje jen účetnictví. Pokud přidáme k účetnictví například internetový obchod, už nepůjde jen o digitalizovanou část agendy, ale budeme mít v systému informace o výrobcích a jejich prodeji, u kterých dokážeme zpětně analyzovat tok zboží různými cestami a tyto data využít k rozvoji firmy. Získáme systém, který můžeme zjistit více o firmě jejím fungování. Pro malou firmu už bychom mohli mluvit o ERP, pro větší firmu vyrábějící nějaký typ produktu však představuje uvedený příklad jen rozšíření účetnictví. Rozhodně pro ni není ERP systémem, ten by musel obsahovat alespoň výrobní postupy a nejlépe pokud by byli všechny části systému provázané. Pokud však chceme dokonalejší systém, měl by obsahovat komplexnější informace než jen výrobu, prodej a digitální vedení ekonomických agend. Takto postavený systém bude mít jinou strukturu pro výrobní organizaci ve strojírenství než v potravinářství. Pokud bychom chtěli vytvořit třeba jen pro každý obor jeden systém,

existovalo by jich několik desítek. Vývoj a údržba takového počtu systémů by byl nákladný, proto existuje pouze několik obecnějších systémů. Ty je však nutné přizpůsobit jednotlivým požadavkům firem. Právě rozdílnost požadavků způsobuje problémy při implementaci. Může se pak stát, že některé procesy jsou implementovány lépe a jiné hůře. Nevhodně implementované procesy je pak nutno postupem doby upravit, abychom uspokojili i uživatele pracující s touto částí systému. Pokud je možné upravit informační systém, je všechno v pořádku. Horší je situace, pokud informační systém tokový zásah nepovolí a je nutné vytvořit aplikaci pro část firmy. Jde o nesystémové řešení hlavně pro uživatele, kteří jsou nuceni pracovat s oběma programy. Informační systémy se však vyvíjí a rostou jejich možnosti, pak uvedená nesystémovost může být ukončena novou verzí informačního systému. Dočasné řešení jinou aplikací nemusí představovat jen problém, například pokud aplikace doplňující informační systém dokáže uspořit finanční prostředky nebo práci zaměstnanců.

2 NÁVRH APLIKACE

2.1 Volba typu databáze

Databáze již slouží pro shromaždování dat řadu let a za tu dobu se vytvořilo několik způsobů, jak data uložit. Jeden z nich je objektová databáze, její fungování je obdobné jako u objektových programů. Data jsou prakticky objekty, na kterých je postavena logika programu. Objekty se netvoří až skládáním dat v aplikaci, ale jsou jimi již v databázi. Existují objektové databáze, které umí snadno převést objekty v aplikaci na objekty v databázi. Bylo by velmi zajímavé je vyzkoušet. Proti nim však je malé povědomí o těchto databázích. A dalším je nutnost naučit se s tímto typem software pracovat. Osobně jsem tento typ databáze ještě nikdy nepoužil. A stejně na tom jsou všichni zbývající spolupracovníci. Objektová databáze tím pádem není v našem případě reálným prostředkem pro využití. Proti jsou i další fakta. Pro nasazení jsou důležitými faktory i rychlost, cena vývoje a údržby. Pokud se budeme zabývat objektovými databázemi, pak se práce na požadovaných úpravách protáhne o několik měsíců. Tím může dojít k situaci, že se aplikace nebude používat po takovou dobu, aby cena za vývoj převýšila zisky plynoucí z jejího používání.

Další z možností, která byla na začátku zvažována, je analytická databáze. Představa o jejím nasazení se zakládala na požadavcích managementu. Výhoda těchto databází se projeví při velkém rozsahu dat. Vzhledem k tomu, jaká funkčnost je nakonec vyžadována, není tento typ databáze vhodný pro zhruba polovinu z požadovaných dat. Vzhledem k možnostem a výkonu dnešních počítačů je však poměrně jednoduché získat data z relační databáze i pro analytické funkce. Objem dat není takového rozsahu, abychom nutně museli používat analytickou databázi. Avšak pro některá data by byla velmi výhodná. Ve prospěch analytické databáze jsou i zkušenosti pracovníků se zpracováním dat pomocí tohoto typu databáze. Z tohoto důvodu bylo rozhodnuto, že část dat týkající se obchodu bude zpracována podle zásad pro analytické databáze.

Poslední z původně navrhovaných modelů je databáze relační. I když ji uvádím jako poslední, v současnosti je to asi nejvíce využívaný typ databáze. Její principy jsou jednoduché a s její praktickou aplikací nemají problémy ani lidé, kteří k databázím nemají příliš blízko. Pro potřeby aplikace bude tento typ použit k uložení dat o výrobcích a výrobních postupech. Rozsah dat je z dnešního pohledu malý. Jedná se o desítky,

maximálně o stovky tisíc záznamů. Při současných cenách hardware je zpracování takto uložených dat levnou a rychlou záležitostí.

Podle rozhodnutí budou tedy data uložena ve dvou typech databází. Výhodně se využije analytická databáze pro části projektu, kde není nutné pracovat s podrobnými informacemi, ale naopak již s daty agregovanými určitým způsobem. Další část dat bude uložena v relační databázi. Normalizace tabulek bude provedena podle potřeby. Důvodem k tomuto rozhodnutí je celkový rozsah projektu. Jedná se o jednodušší aplikaci, která se bude spíše zmenšovat vzhledem k předpokládanému rozšiřování funkčnosti informačního systému.

2.2 Objektové programování

Posledních dvacet let je jiný způsob programování než objektový považován za zastaralý a nevhodný. Zřejmě by způsob programování aplikace měl být jednoznačně určen tímto trendem. Sám již takový optimista nejsem, předpokládám, že část aplikací není psána objektově, ale procedurálně s využitím objektů, které poskytuje daný jazyk a platforma. Důvod proč tomu tak je, je obrovský rozsah možností dnešních databázových systémů. K tomu přičtěme bohatou nabídku prvků pro vývoj aplikací. Není důvod psát si další kód duplicitně k těmto nástrojům. Tento postup je patrně zvolen v naprosté většině menších aplikací. Jen pro některé části malých aplikací a pro velké projekty je nutné napsat kód objektově. I když se zdá použití jiných technik než objektového programování ne moc šťastné rozhodnutí, bývá někdy využití procedurálních funkcí dostatečné. Nejedná se většinou o aplikace, které by měly být základem pro další vývoj. Aplikace a programy, které dodržují zásady objektového programování, však jsou jednoduše rozšiřitelné a mohou být základem pro další rozvoj. Důvodem mohou být dosud neznámé požadavky, které vzniknou v následujících obdobích. Přepisování části aplikací je pak ztrátou času, ale není neobvyklé. Spíše tvoří pravidlo. Prakticky všechny současné a běžně využívané programy, prošli fází přepsání kódu.

2.3 Klikací programování

Klikací nástroje mají zajímavý vzhled a budí dojem jednoduchosti, nicméně pro mě jsou hůře ovladatelné a proto pomalejší při práci než napsat několik řádků skriptu. Navíc je pro mě skript skrytý za obrázky špatně čitelný. Možná, pokud se naučíte programovat klikáním, zdrojový text nepotřebujete vůbec zobrazovat. Třeba jde jen o otázku zvyku než opodstatněných argumentů. Nejlépe bych dokumentoval uvedené argumenty na příkladu dotazu generovaného po klikání a napsaného ručně podle pravidel, které jsme si určili. I když podle norem má být výkonnější dotaz, který spojuje tabulky pomocí klausule *JOIN*, přesto se mi zdá méně čitelný. Navíc jde o velmi jednoduchý příklad, v případě složitějších dotazů může generovaný kód působit dojmem složitosti. Další ukázky jsou v příloze.

```
--generovaný dotaz
SELECT    Zbozi.Nazev, Zbozi.Zkratka, Zbozi.MernaJed, ZboziDruh.Nazev
          AS Expr1, ZboziKod.Nazev AS Expr2, ZboziSkupina.Nazev AS
          Expr3

FROM      Zbozi INNER JOIN
                ZboziDruh ON Zbozi.ID_Druh = Zbozi_Druh.ID INNER JOIN
                ZboziKod ON Zbozi.ID_Kod = Zbozi_Kod.ID INNER JOIN
                ZboziSkupina ON Zbozi.ID_Skupina = Zbozi_Skupina.ID

--psaný dotaz
SELECT zb.Nazev, zb.Zkratka, zb.MernaJed, d.Nazev AS Druh,
       k.Nazev AS Kod, s.Nazev AS Skupina
FROM   Zbozi zb, ZboziDruh d, ZboziKod k, ZboziSkupina s
       --propojení tabulek
WHERE  zb.ID_Druh = d.ID
       AND zb.ID_Kod = k.ID
       AND zb.ID_Skupina = s.ID
       --omezení
```

Podobná situace platí i v programovacích nástrojích. Klikání na rozbalovací nabídky vypadá elegantně, ale vyžaduje docela dobrou znalost hierarchie požadovaných položek. Pokud však použijeme jen klasické psaní podle nápovědy nebo internetu, pak mi přijde často i rychlejší kopírování kusů kódu z internetu a jejich úprava pro vlastní potřeby. Existují však situace, na které klikací nástroje neumí reagovat a je nutné dopnit jejich funkčnost psaným kódem. Výhodou klikacích nástrojů se stává uložení daného prvku do xml souboru načítaného až při běhu programu, velmi jednoduše pak změníte vzhled formuláře bez nutnosti přeložit celý nebo část kódu znova. Pokud je kód napsán ručně, tak o tuto možnost přicházíte. Nejde o čas, který strávíte znovuvytvořením programu, ten je u malých projektů zanedbatelný. Podstatě je horší samotná úprava kódu a zavlečení

nových chyb do aplikace. Výrobce programu tak staví programátora před danou věc, část, která jde naklikat a ukládá se mimo zkompilovaný soubor, vytvořit touto metodou. I když se nemusí všem a vždy líbit. Škoda, že moderní nástroje nedokáží z kódu vytahat informace a ty pak uložit mimo kompilovanou část.

2.4 Cizí software

V době kdy můžete na internetu nalézt miliony řádků zdrojových kódů, nemusí být nejefektivnější psaní vlastních programů. Na internetu můžete nalézt poměrně složité programy nebo zdrojové kódy, které jsou již zbaveny chyb. Dokážou tedy ušetřit spoustu vaší práce. Je však nutné zvážit, zda je pro firmu efektivní použití již existujících řešení. Pokud jde o placený software, dá se poměrně dobře odhadnout, jestli se náklady vynaložené na jeho nákup vrátí. Horší je postup v případě open source kódu. Ten je většinou k dispozici pod GNU licencí. Pokud se rozhodnete jej využít, pak musíte uvolnit pod touto licencí i vlastní produkt založený na tomto kódu. V některých případech problém s uvolněním není. Může však vzniknout kód, který není vhodné vypustit do okolního světa. Tím pak padá možnost využívat tento open source. Využití cizího kódu je vhodné zvážit na počátku řešení.

2.5 Software pro databázi

Původní návrhy na databázové stroje byly tři. Oracle 10g, MySQL 5.0 a Microsoft SQL Server 2005. Oracle pokud nepotřebujete uložit více než 4 GB dat je zdarma. Bohužel tento limit je nastaven tak, aby byl vhodný jen pro testování nebo jednoduchý webový server. Nákup licence pro nadstavbu je mimo reálné náklady na tento systém.

Dalším rozšířeným databázovým strojem je MySQL server. Začíná se podobat velkým databázím, poskytuje již značné množství vestavěných funkcí. Výhodou je cena tohoto software. Nevýhodou je podpora práce s daty. Existují již nástroje, které dokážou MySQL dotáhnout na vysokou úroveň tak, že práce je pak snadná a rychlá, ale tyto nástroje nejsou zdarma. Prakticky zde platí, pokud je nástroj dobrý, je drahý. Levný neposkytuje takový komfort a dost často obsahuje chyby v programu.

Databázový software Microsoft SQL server 2005 má před ostatními ve firmě jednu velkou výhodu. Je již zakoupen a není nutné jej pořizovat. Funkčností se blíží Oracle 10g a náklady na pořízení jsou pro nás nulové. Je však zakoupena pouze jedna licence. Proto musí být databáze pro aplikaci na serveru, kde již běží informační systém. Výhodou

serveru je jeho výkonová rezerva. Současný informační systém nedokáže využít potenciálu tohoto serveru.

Ovšem databáze pro aplikaci si ukousne část diskové kapacity. To může být problémem v budoucnu. Protože databáze jen porostou a se snižováním objemu dat nemůžeme zatím počítat ani v našem případě. Po naplnění daty se může dostat produkční databáze informačního systému na hranici požadovaných hodnot výrobcem tohoto systému. Řešení problému však není nákladné. Případné rozšíření diskové kapacity při současných cenách hardware nezatíží celkové náklady na vývoj aplikace velkou částkou.

Volba databázového systému půjde cestou nejmenšího odporu. Ze všech databázových systémů nám nejlevněji vychází MS SQL server. Nikoliv však cenou, ale náklady, které budeme muset vynaložit na budoucí aplikaci.

2.6 Software pro vývoj aplikace

Pro vývoj aplikace bylo již předem vybráno Visual Studio Profesional 2005 od firmy Microsoft. Nasazení jiného operačního systému není minimálně ve střednědobém horizontu reálné, není tedy nutné, aby aplikace byla multiplatformní. Pro další vývoj a údržbu se počítá s přechodem na Visual Studio Profesional 2008. Možností jak urychlit vývoj, je nasazení upravených ovládacích prvků jiných stran. Tyto by měly více odpovídat našim požadavkům než prvky, které obsahuje samotné vývojové prostředí.

Z rozsáhlé nabídky byl zvolen produkt firmy Developer Express Inc, dostupný v trial verzi na stránkách této firmy[999]. Úprava všech standardních ovládacích prvků je dovedena až na hranici zbytečnosti. Tím nechci uvedený produkt kritizovat, právě naopak. Nabízí prvky pro desktopové aplikace a v dnešní době neodmyslitelně i pro webové aplikace. Rozsah možností prvků je z dnešního pohledu neomezený. Ovládací prvky, které potřebujete nebo si myslíte, že budete potřebovat, není nutné programovat. Uvedený produkt je již obsahuje v takovém provedení a kvalitě, že již není nutné přemýšlet nad úpravami standardních prvků.

Bohatou skupinou jsou zejména ovládací prvky pro plánování a spolupráci s Exchange serverem. Jejich možnosti jsou daleko větší, než jaké v současnosti představuje Microsoft Outlook ve verzi 2007.

Každého uživatele potěší, pokud si může nastavit prostředí k obrazu svému. Tato nadstavba Visual Studia k tomuto řešení přímo vybízí. Ovládací prvky již uvedené

vlastnosti mají implementovány. Další možnosti si uvědomíte až při vlastním odzkoušení uvedené nadstavby.

Patrně nejdříve, ještě předtím než si jakýkoliv produkt prohlédneme, si zjistíme cenu. Záměrně ji však uvádím až na konci. Je větší než cena Visual studia ve verzi Profesional. Ovšem při kurzu dolaru, který se nyní a pohybuje se kolem 16 korun, výsledná cena představuje částku kolem 25 tisíc korun. Uvedená cena je za nadstavbu Profesional včetně zdrojového kódu. Na první pohled je částka vysoká. Pokud si však spočítáme náklady na úpravu jen několika málo standardních prvků, vyjde nám částka podstatně vyšší. Není nutné dlouho přemýšlet ani počítat, jestli se uvedená nadstavba vyplatí nebo ne.

2.7 Programovací jazyk C#

Vlastní kód aplikace, budeme psát v jazyku C#. Jiný jazyk nepřichází do úvahy. Bohužel jsem jediný, který uvedený jazyk ještě nepoužíval. Předpokládám však, podle některých publikací, avizovanou podobnost s jazyky Java, C++ a Visual Basic, se kterými jsem v minulosti pracoval. Alespoň si vyzkouším nejvíce proklamovaný jazyk poslední doby.

II. PRAKTICKÁ ČÁST

3 ANALÝZA POŽADAVKŮ

3.1 Informační systém

Ve firmě, pro kterou pracuji, je nasazen informační systém, který řeší specifické požadavky zákazníků pomocí specializovaných funkcí. Informační systém tak může převzít činnosti, které jsou v daném prostředí rutinní a je možné jejich postupy algoritmizovat. Taktéž poskytuje prostředí pro vlastní rozšíření systému. Data jsou uchována v databázi Microsoft SQL Server 2005.

3.2 Požadavky na novou funkčnost

3.2.1 Výroba

V informačním systému jsou vedeny informace o technologickém postupu výroby jednotlivých výrobků, o aktuálním stavu rozpracovanosti na jednotlivých výrobních zakázkách a o nových požadavcích od obchodníků. V současné implementaci chybí plánování výroby v reálném čase a realizace nových obchodů v uskutečnitelných dodacích lhůtách. Současný systém pro tuto funkčnost sice obsahuje některé obecné nástroje, ale pro konkrétní implementace jsou řešeny na míru dodavatelem informačního systému, případně vlastními silami firmy. Vedoucí výrobních linek požadují naplánování výroby tak, aby konečný výrobek byl expedován k zákazníkovi co nejdříve po provedení poslední operace. Tím by odpadla nutnost výrobky skladovat, nebo naopak posunovat termín dokončení zakázky.

3.2.2 Obchod

Informační systém zabezpečuje evidenci a vzájemné propojení objednávek, zakázek, dodacích listů, faktur a dalších dokladů důležitých pro oblast prodeje výrobků a nákupu materiálu. Pro prodejce je důležité mít ihned informace o daném odběrateli, jeho historii prodeje, nastavení cen nebo finanční závazky vůči společnosti. Vše systém poskytuje, bohužel, data se generují z OLAPu a jsou k dispozici s několikaminutovým zpožděním. Což může vést ke komplikacím při sjednávání zakázky. Systém je tedy vybaven všemi potřebnými daty, jenže jsou předkládána uživateli s nedostatečnou rychlostí.

Dalším krokem je spolupráce obchodu s výrobou a reálné dodací lhůty v uzavíraných objednávkách.

3.2.3 Management

Potřeby řídících pracovníků jsou shodné snad ve všech organizacích, představují vyhodnocení obchodu, efektivity výroby, rozhodování o budoucím směrování firmy a jejím výrobním sortimentu. Pro výpočet předběžných kalkulací výroby obsahuje informační systém základní podporu, nedokáže však jednoduše měnit vstupní podmínky pro skupinu dat nebo daný výrobek a bez složitých procedur ihned poskytovat výsledky včetně základní analýzy problému.

Jiný typ dat lze získat z průběhu výroby. Data jsou uložena v informačním systému. Pro jejich vyhodnocení je nezbytné v aktuální verzi informačního systému provést řadu složitých operací. Další oblastí je zpracování přehledů prodeje a analýz nad datovými krychlemi. Informační systém tyto analýzy plně podporuje, výhrady opět směřují k rychlosti daných úloh.

3.3 Cíl projektu

Po analýze požadavků jednotlivých typů uživatelů, doplnit potřebné informace přímo do informačního systému, případně do aplikace, která bude jeho nadstavbou a bude plně využívat data shromážděná informačním systémem z různých stupňů výroby. Rozšíření informačního systému, případně nadstavbová aplikace, mají poskytovat uživatelský komfort při zpracování a analýze dat. Předpokládá se, že daná rozšíření budou využívána do doby, kdy podobnou nebo shodnou funkčnost poskytne informační systém ve svém základu. Tato doba se může podle odhadů pohybovat od jednoho do pěti let.

3.4 Požadavky na rozšíření

3.4.1 Výroba

- plánování výroby
- plánování nákupu materiálu
- on-line přenos výsledků plánování na jednotlivá pracoviště
- zpětná vazba s informacemi o průběhu výroby
- vyhodnocení, nastavení a optimalizace kapacit výroby
- reálné termíny pro nové objednávky

3.4.2 Obchod

- okamžité informace o odběrateli
- aktuální nastavení cen odběratele

- závazky odběratele, závazky vůči odběrateli
- přehled zakázek ve výrobě, nebo uzavřené zakázky
- reálné dodací lhůty pro nové objednávky
- přehled historie prodeje odběrateli

3.4.3 Management

- kalkulace s hromadnými změnami vstupních parametrů
- výpočty kalkulací a porovnání s aktuální variantou
- výpočty nákladů na výrobu
- analýza problémových částí výroby
- optimalizace výrobních kapacit
- rychlé výsledky z OLAP analýz
- vyhodnocení prodeje a jednotlivých obchodníků

vyhledání nevýhodných zakázek

- vyhledání nepříznivých trendů
- regionální analýzy

3.5 Požadavky na obsah sestav

3.5.1 Výroba

plánování výroby

- počátek výrobního procesu
- jednotlivé operace včetně času
- informace o předchozí operaci
- informace o následující operaci
- informace o skluzu zakázky
- informace budou on-line zobrazeny ve výrobě

plánování nákupu materiálu

- požadavky na objem materiálu
- nejpozdější termín nákupu
- přehled posledních nákupů daného materiálu včetně cen
- přehled spotřeby za zvolené období

zpětná vazba s informacemi o průběhu výroby

- rozdíl mezi skutečným trváním operace a plánovanou dobou
- neplánované náklady na výrobu
- vyhodnocení kvality výroby

vyhodnocení, nastavení a optimalizace kapacit výroby

- naplnění kapacit pracovišť
- úprava kapacit podle výroby
- rozšíření nebo snížení kapacit výroby
- vyhledání pracoviště s nejvyšší zmetkovostí

reálné termíny pro nové objednávky

- nejbližší termín pro dodání výrobku
- kontrola termínů nových zakázek

3.5.2 Obchod

okamžité informace o odběrateli

- základní data
- sjednané schůzky
- poslední obchody
- nabídky a akce pro daného odběratele

aktuální nastavení cen odběratele

- přehled nejvíce odebíraných výrobků s posledními cenami
- upozornění na možné slevy
- upozornění na nabídkovou akci
- varování při nízké rentabilitě obchodu

závazky odběratele, závazky vůči odběrateli

- přehled závazků odběratele
- upozornění na prodlení závazků
- přehled budoucích závazků s uzavřených zakázek
- přehled závazků dodavatele

přehled zakázek ve výrobě, nebo uzavřené zakázky

- přehled zakázek před zahájením výroby
- přehled zakázek ve výrobě, stav rozpracovanosti a termín dokončení
- upozornění na skluzy v dodávkách
- přehled reklamací a jejich vyřízení

reálné dodací lhůty pro nové objednávky

- shodné s požadavky výroby

přehled historie prodeje odběrateli

- souhrn po jednotlivých obdobích
- souhrn podle skupin výrobků
- upozornění na klesající trendy

3.5.3 Management

kalkulace s hromadnými změnami vstupních parametrů, výpočty kalkulací a porovnání s aktuální variantou

- možnost změny jednoho nebo více parametrů
- jednoduchá analýza celkového dopadu změny parametrů
- jednoduchá analýza dopadu jednotlivých parametrů

výpočty nákladů na výrobu, analýza problémových částí výroby, optimalizace výrobních kapacit

- částečně se překrývá s požadavky výroby
- rozšíření přehledu režijních nákladů
- souhrnné pohledy na výrobní linky
- náklady na dopravu výrobků

rychlé výsledky z OLAP analýz, vyhodnocení prodeje a jednotlivých obchodníků, vyhledání nevýhodných zakázek

- přehled nejvýhodnějších zakázek
- přehled ztrátových zakázek
- přehled ziskovosti jednotlivých prodejců
- výhled do blízké budoucnosti

vyhledání nepříznivých trendů, regionální analýzy

- přehled výrobků s klesajícím prodejem
- přehled regionů s klesajícím prodejem
- dostupné informace o konkurenčních firmách v daném regionu
- přehled dostupných informací a cen o konkurenčních výrobcích

3.6 Požadavky na rychlost zpracování sestav

Současný informační systém obsahuje řadu požadovaných sestav, ale nedokáže je poskytovat v dostatečné rychlosti. Tyto jsou pak terčem kritiky ze strany uživatelů a vnáší do prostředí firmy averzi vůči informačnímu systému.

Sestavy je možné přepsat a optimalizovat pomocí vestavěného vývojového prostředí. Dodavatel informačního systému neručí za další provoz takto upravených sestav. Údržba těchto speciálních částí systému je pak plně v režii firmy, která si úpravy provedla. V případě změny v informačním systému může dojít k nefunkčnosti dané speciální úpravy.

Cílem úprav je zvýšit informativní hodnotu sestav a zkrátit dobu jejich zpracování. Protože nebudou sloužit mnoha firmám v různých oborech, nemusí být upravené sestavy obecné. Výsledkem bude sestava odpovídající požadavkům uživatelů dané firmy.

Alternativou úpravy sestav je vybudování vlastní aplikace, ve které bude struktura informací a rychlost přizpůsobena požadavkům uživatelů. Vzhledem k předpokládané omezené době funkčnosti aplikace je možné akceptovat i některé nestandardní postupy pro tvorbu a vývoj software.

3.7 Aktuální informační systém

Informace o struktuře a chování systému nejsou k dispozici. Následující předpoklady o stavbě a chování informačního systému jsou výsledkem vlastních úvah. Nemusí odpovídat skutečnému stavu a nejsou prezentovány s úmyslem poškodit dodavatele informačního systému.

Informační systém je naprogramován v jazyce Delphi. Je ryze objektovým programem, ve kterém jsou data striktně oddělena od logiky programu. Data jsou načtena do objektů na základě požadavků uživatelů. Objekty se kromě vlastností v naprosté většině případů

skládají z podřízených objektů. Podřízené objekty mohou být složeny z dalších podřízených objektů. Tento model je vynikající z pohledu programátora a analytika. Má jistě obrovské možnosti i díky obecnosti některých objektů. Uživatel takového systému se ale dostane poměrně běžně do situace, kdy je nutné procházet velký počet podřízených objektů v několika úrovních. Výsledkem je čekání, než program dokončí zpracování dat.

Informační systém může ukládat data do databáze firmy Microsoft nebo Oracle. V našem případě se jedná o databázi SQL Server 2005 firmy Microsoft. Databáze není objektového typu, ale podle struktury tabulek a údajů jde nejspíše o objektové uložení dat. Důvodem je pravděpodobně jednoduché navázání na objekty samotného informačního systému. Toto uspořádání dat však přináší i některé nevýhody. Logika a interpretace dat je záležitostí samotných objektů. Tudíž data uložená do databáze mohou obsahovat chybné informace, které jsou uvedeny na pravou míru až logikou programu. V databázi lze nalézt spoustu duplicitních údajů, chybějící data a v některých případech i jejich nekonzistentnost. Tato nekonzistence se projevuje např. chybnými hodnotami (např. součty) v hlavičkách dokladů, které neodpovídají hodnotám v položkách.

Nekonzistencí zde nemám na mysli chybné odkazy na neexistující záznamy jako v relačních databázích. V případě duplikujících se dat naleznete záznamy, které nemají odpovídající duplikát. Jak tato data interpretovat, poznáte jedině z vnějšího chování informačního systému.

Pokud bychom chtěli vytvořit rychlejší sestavy v rámci informačního systému, pak je nutné data načítat nikoliv procházením objektů, ale tvorbou SQL dotazů a získávat je přímým dotazem do databáze. Data uložená ve výše popsané podobě nám budou komplikovat tvorbu SQL dotazů nebo budou vyžadovat následné úpravy pomocí programu. Což v případě rozsáhlejších dat povede ke zpomalení vyhodnocení nebo pomalejšímu vygenerování sestavy.

Lepší situace nebude ani v případě tvorby vlastní aplikace založené na relační databázi. Data bude nutné očistit a uložit v jiném typu tabulek.

4 NÁVRH APLIKACE

4.1 Konfigurace databáze

Nastavení databáze se převezme z produkční verze informačního systému. Sníženy budou pouze limity pro velikosti datového souboru a velikost transakčního logu na úroveň, která nebude omezovat informační systém.

Objem dat uložených v databázi aplikace by neměl podle odhadu překročit 10GB. Představuje několikrát menší objem dat, než má v současné době produkční databáze informačního systému. Záloha databáze bude záležitostí správce sítě.

Přístup do databáze bude autentifikován přes systém windows. Přístup jednotlivých uživatelů bude dále omezen na jednotlivých tabulkách, pohledech, procedurách a funkcích.

4.2 Návrh uložení dat

Data v databázi aplikace nebudou kopírovat databázi informačního systému, ale naprostá většina z nich bude přetransformována do nových tabulek.

Z pohledu aplikace například není důvod uchovávat číselníky v jedné tabulce. Nebudou tak obecné, abychom je museli ukládat v obecném tvaru a následně přetypovávat pro naše potřeby.

Některá data jsou z důvodů obecnosti informačního systému uložena v databázi v řádcích. Pokud bychom je stejně uložili i v databázi pro aplikaci, pak ztratíme v určitých částech aplikace určitou dobu překlápěním mezi řádky a sloupci nebo psaním objektů, které tato data budou správně interpretovat.

Tabulky informačního systému jsou pro naše potřeby nevhodně indexovány. Indexy jsou poplatné objektovému návrhu informačního systému. Pro relační databáze nebo analytické funkce budeme muset vytvořit nové indexy založené na primárních a cizích klíčích a v analytické části také na datumových položkách. Počet indexů omezovat nebudeme, ale nepředpokládá se více než 4 až pět indexovaných sloupců na tabulku.

Pokud bude nutné v aplikaci počítat některá pole platná pro všechny řádky tabulky, budou uchována jako počítaná pole tabulky. Pro výpočty, které budou použity v databázi, se budou striktně využívat funkce nebo uložené procedury. Důvodem je snížení zátěže serveru i přes jeho výkonnostní rezervu. Pokud budeme požadovat v aplikaci zobrazení dat

z více tabulek, bude pro tento účel vytvořen pohled nebo tabulková funkce. Výhodou tabulkových funkcí jsou parametry, které dokáže funkce přijímat a zpracovat. Pokud bude nutné zobrazovat data podle uživatele, je její použití jasné, rychlé a pohodlné.

Pro analytickou část programu budou data uložena v takovém tvaru a stupni denormalizace, aby jejich načtení bylo rychlé a pohodlné z prostředí Visual Studia. U těchto dat nebude z logických důvodů docházek k aktualizaci, proto denormalizace není na škodu. Pokud některá část aplikace bude používat vkládání nových dat jejich úpravu nebo smazání a bude požadována jejich historizace, budou data ukládána ve stejném typu tabulky avšak s jiným názvem. Patrně s příponou „hist“ nebo „min“ za jménem tabulky. Nebude se tím komplikovat návrh tabulek, ve kterých budou najednou aktuální data a data z minulosti. U těchto tabulek však bude nutné vypnout některá integritní omezení.

4.3 Aktualizace dat

Data zastarávají prakticky okamžikem načtení ze zdroje. Po určité době klesne jejich informační hodnota a přestanou být zajímavá. Bude proto nutné vytvořit systém pro jejich aktualizaci. Frekvence aktualizace není jednoznačně daná. Některý typ dat stačí aktualizovat 1x za měsíc, případně ještě méně často. Mezi taková data patří například základní číselníky. Jiná data mohou být neaktuální již po několika málo minutách (okamžitý stav rozpracovanosti na dílně). Dalším důležitým aspektem aktualizace je samotná délka aktualizace dat. Pokud trvá několik málo sekund, pak není problém v jejím opakování třeba každých 5 minut. V případě, že bude trvat řádově desítky vteřin nebo minut, nebude možné její časté opakované spouštění v době největšího vytížení databází. Bude proto nutné nastavení aktualizace podle potřeb jednotlivých částí aplikace.

4.4 Kontrola integrity

Před zahájením tvorby databáze pro nadstavbu, bude provedeno několik kontrol, které budou mít za úkol nalézt porušení integrity v aktuální databázi informačního systému. Vzhledem ke způsobu uložení dat půjde o důležitou součást příprav pro tvorbu databáze nadstavby. Data budou muset být upravena v aktuálním informačním systému, případně se budou muset při importech a aktualizacích důsledně čistit a opravovat. Data v databázi aplikace budou kontrolována pomocí cizích klíčů a omezeními na sloupce. Chyba ve vstupních datech bude mít za následek vytvoření výjimky. O jejím dalším zpracování se rozhodne podle typu chyby.

4.5 Zásady pro psaní zdrojového kódu

V mnoha knihách se lze dočíst jak pojmenovávat proměnné a objekty. Pokud si však stáhnete a otevřete zdrojový kód z internetu, a to být nejen u malých projektů, je napsaný pokaždé s jiným způsobem formátování. Každý si píše kód podle svých představ a zvyklostí. Nelze říci, že kód napsaný daným stylem je dobrý nebo špatný. Části programátorů vyhovuje, jiné minimálně nepotěší. I pokud se sejdou pracovníci s prakticky shodnými styly, vždy se dají najít drobné rozdíly. Aby se předešlo vzájemnému přeformátovávání, je vhodné hned na začátku určit pravidla psaní kódu.

4.5.1 Obecné pravidla

Pro pojmenování všech druhů objektů se používá jednotné číslo. Tvorba názvů se řídí pascalovskou konvencí (PascalovaKonvence) nebo camelovou konvencí (camelovaKonvemce). Podtržítka se v názvech objektů nepoužívají. Použití podtržítek je opodstatněné pouze v názvech kde se vyskytují jen velká písmena (konstanty JEN_U_VELKYCH). Anglické názvy se používají ve zdrojových kódech programu pro pojmenování všech druhů objektů. Pro pojmenování objektů v databázích se využívá jen českých slov. V databázi se anglické názvy používají pouze pro pojmenování funkcí a procedur. Konstanty a parametry jsou uloženy v tabulce v databázi, ze které se do programu načítají pomocí objektu. Ukládání těchto hodnot do kódu se nevyužívá. Jednoznačné zkratky používané v rámci programu jsou dokumentovány v tabulce databáze. Jedná se o názvy používané ve výrobě a obchodu, které jsou použity k pojmenování objektů v databázi a programu.

4.5.2 Jména objektů v programu

Přehled základních pravidel, jimiž se řídí psaní kódu v editorech vývojových prostředí.

třídy	*PascalovaKonvence*
konstanty, statické proměnné	*camelovaKonvence*
procedury, funkce	*PascalovaKonvence*
lokální proměnné	*camelovaKonvence*
parametry (private)	*camelovaKonvence*
rozhraní, (public)	***I****PascalovaKonvence*

třídy atributů	*PascalovaKonvence**Atribute***
metody	*PascalovaKonvence*
události	*PascalovaKonvence**EventArgs***
třídy výjimek	*PascalovaKonvence**Exception***

Formátování zdrojového kódu se řídí následujícími technikami a pravidly

Odsazování	TAB
Maximální délka řádku	120 znaků (podle rozlišení monitorů)
Cykly, podmínky (umístnění závorek)	*řádek kódu*
{	
řádek kódu	
}	
Oddělení funkcí	///XML popis
Komentáře	blok se popisuje před jeho kódem,
	do řádků jen krátký popis

4.5.3 Konvence pro databázi

V SQL dotazech je klíčové slovo zvýrazněno velkými písmeny, názvy polí a dalších objektů v databázi píšeme shodně s jejich jmény v DB. Komentáře v SQL dotazu se uvedou před samotným dotazem.

Pokud je použité fiktivní ID, pak se ponechá generovaná hodnota ID typu číslo se zvyšováním o hodnotu 1. Počet a velikost indexů není neomezen. Integrita databáze se udržuje pomocí cizích klíčů, pro vlastní tabulky se integrita vyžaduje bez výjimky. Pro importovaná data je možné integritní omezení vypnout. Kalkulované pole jsou ukládána v databázi jako sloupec a mají název začínající malým písmenem cf (cfNazev).

schéma	velmi krátký název
tabulky	plné názvy PascalovaKonvence
pohledy	začíná malým v (vPascalovaKonvence)
indexy	IX_POLE (standart MS SQL)

vazby	FK_TAB1_TAB1 (standart MS SQL)
sloupce	PascalovaKonvence
ID sloupce	NazevTabulkyID
funkce	začíná malým f (fPascalovaKonvence)
procedury	začíná malým p (pPascalovaKonvence)
pro přidávání záznamů	pInsPascalovaKonvence
pro mazání zánamů	pDelPascalovaKonvence
pro změnu záznamů	pUpdPascalovaKonvence

Formátování SQL dotazu se řídí podle následujícího vzoru

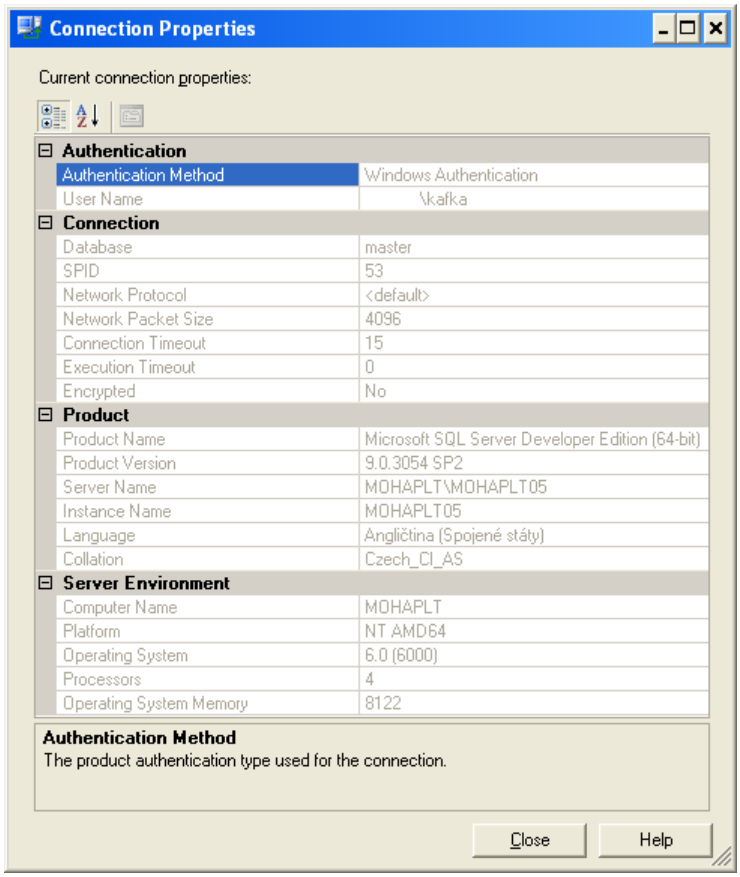
```
SELECT      funkce, sloupce
FROM        tabulky
WHERE       podmínky AND podmínky
            AND podmínky AND podmínky
            AND vnořenýDotaz   SELECT    sloupce
                               FROM      tabulky
                                  :
                                  :
GROUP BY
      :
      :
```

5 UŽIVATELSKÉ ROZHRANÍ

5.1 Databáze aplikace

5.1.1 Databázový server

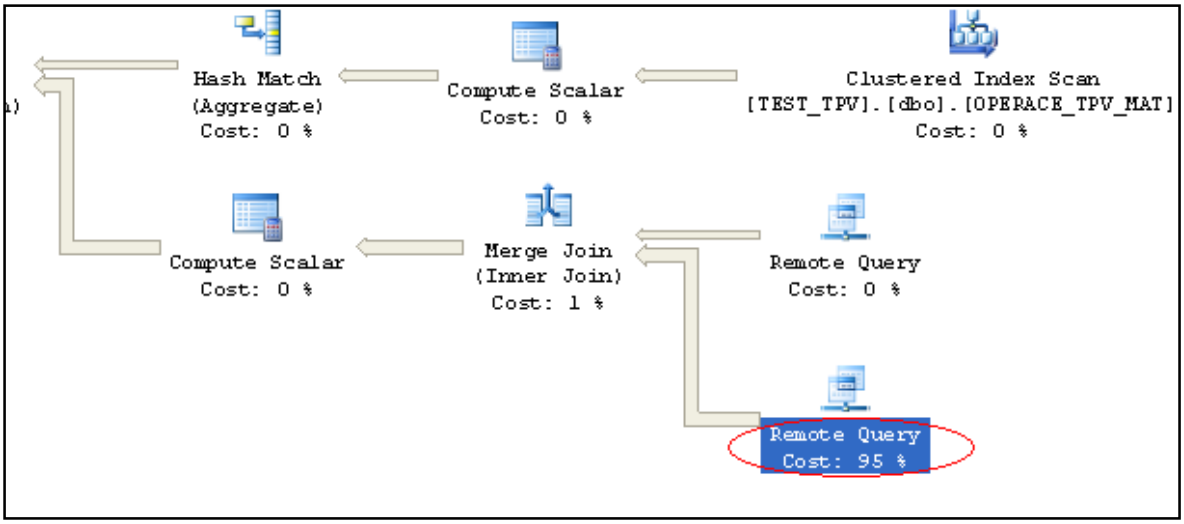
Pro vytvoření první testovací databáze byl použit počítač s operačním systémem Microsoft XP SP2, na který byla nainstalována zkušební software SQL Server 2005. Tuto verzi je možné používat 180 dní. Nastavení nebylo výrazně upravováno, parametry byly ponechány většinou původní. Pro testovací účely je většina nastavení postačující a údaje o počítači a SQL serveru jsou na následujícím obrázku. Tato instalace umožňuje použití analytických, integračních a reportovacích služeb.



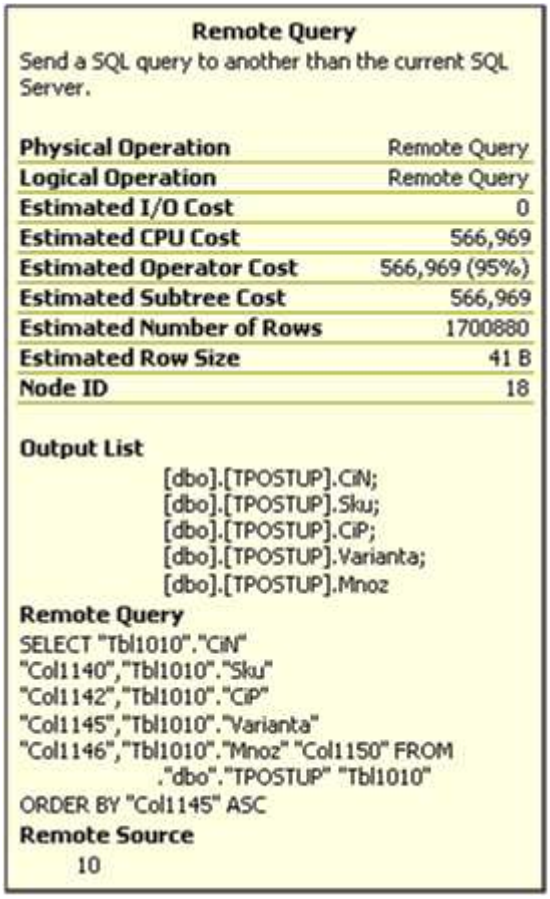
Obrázek 1. Propojení na databázi

5.1.2 Propojení databáze na informační systém

Pomocí linkovaného serveru se napojíme na produkční databázi informačního systému. Tato databáze bude využita pouze jako zdroj dat bez dalšího využívání jejich služeb nebo poskytovaných funkčností. Důvodem je co nejnižší, případně žádné zamykání dat. Veškeré úpravy dat a manipulaci s nimi budeme provádět až ve vlastní databázi. Distribuované dotazy tedy používat budeme jen v omezené míře. Je zakázáno používat jiných typů dotazů než SELECT. Dotazy do produkční databáze budou pokud možno optimalizovány nebo jejich vhodnost posouzena pomocí funkce Estimated Execution Plan, kterou poskytuje Microsoft SQL Server Management studio.



Obrázek 2. Grafický plán vykonání dotazu



Obrázek 3. Přehledné informace k operaci

Například výše uvedené obrázky naznačují problém v dotazu do vzdálené databáze, který trval několik desítek sekund. Takto dlouho trvající dotazy mají omezené použití při aktualizaci dat. Snahou je tyto dotazy upravit nebo rozdělit na několik menších.

5.1.3 Procedury pro aktualizaci

Do aplikace potřebujeme dostat velkou část dat produkční databáze. Cest jak toho dosáhnout je více, MS SQL Server například obsahuje integrační služby, které jsou již značně automatizovány a ovládají se již víceméně pomocí grafického rozhraní. Ne každému tento postup musí vyhovovat. Pro aktualizaci části databáze, kterou jsem zpracoval osobně, jsem použil uložených procedur. Jednotlivé tabulky se aktualizují pomocí uložených procedur. Procedury mají většinou tři části. První část procedury zajistí doplnění daty, které ještě v tabulce nemáme. Tato data vznikla v době mezi předchozí a právě probíhající aktualizací. Představují nové záznamy nebo změny stavů položek. Ve druhé části aktualizace je provedena kontrola na zánik nebo zneplatnění záznamů

v informačním systému. Tato data jsou pak v databázi aplikace smazána nebo historizována. Podle typu tabulky jsou pak data úplně smazány, nebo v některých případech je doplněn datum platnosti, anebo jsou upraveny, ale původní verze dat je nakopírována do tabulky historie. Kopírování je automatizováno pomocí spouště, která reaguje na úpravu případně mazání dat. Nejnáročnější částí aktualizace je porovnání dat mezi databází aplikace a informačního systému. Jde v podstatě o překontrolování, zda jsou údaje vedené v databázi aplikace shodné s daty v informačním systému. Rozdílná data jsou pak upravena v databázi aplikace. Tuto část aktualizace nemají všechny tabulky. Pokud u tabulky nepotřebujeme znát vývoj, pak tato aktualizace není nutná.

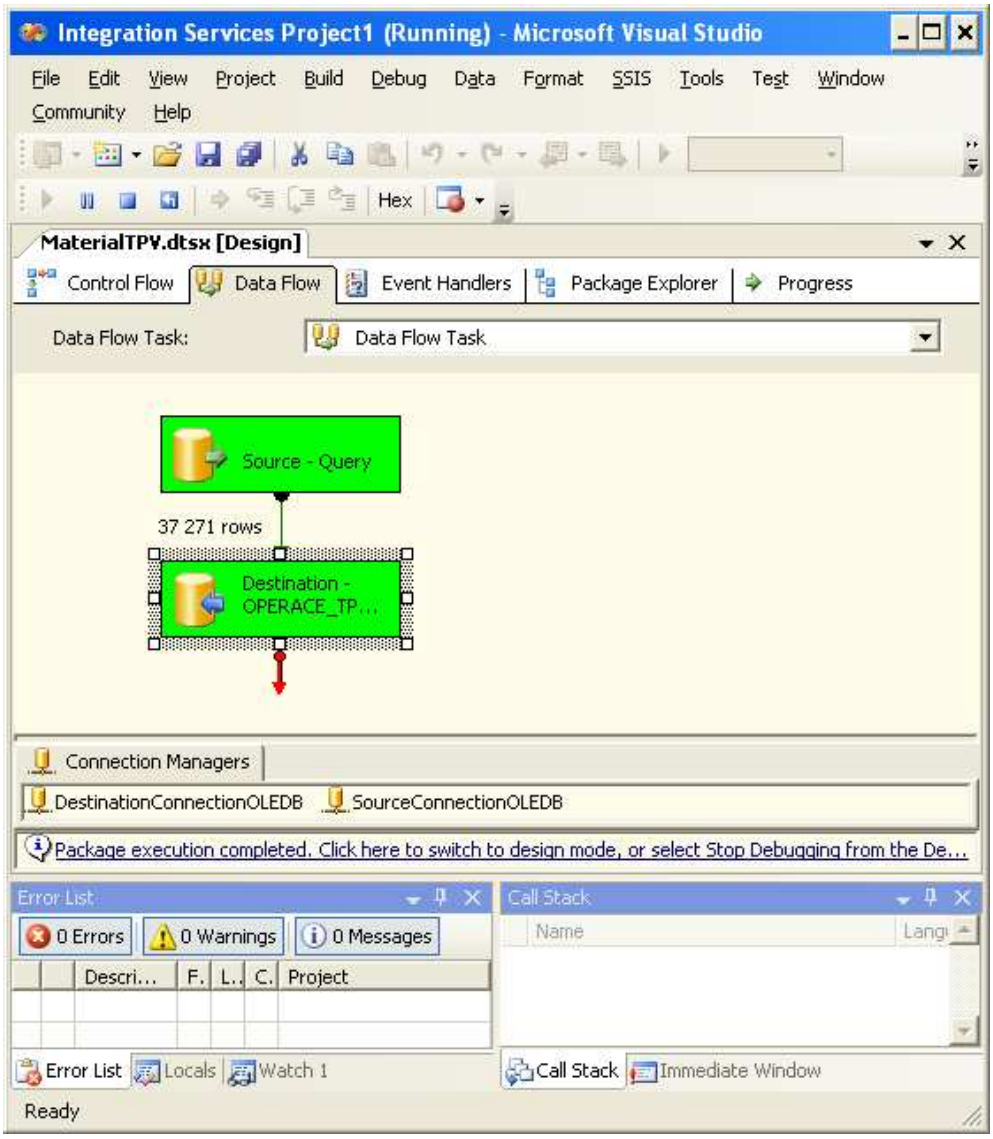
Pro všechny typy aktualizací je využito nových funkcí SQL Serveru. Společný tabulkový výraz vytvoří v paměti zhruba něco jako pohled nebo dočasnou tabulku. Životnost takto vytvořené tabulky je jeden následující dotaz po jejím vytvoření. V dotazu se potom můžeme obracet na tabulkový výraz jako by to byla tabulka. Další novinkou a myslím ještě zajímavější, je použití operandu EXCEPT. Překvapivě rychle dokáže porovnávat dvě sady výsledků proti sobě. Výsledkem porovnání jsou potom řádky, které se nacházejí pouze v levé části operandu. Pomocí těchto příkazů je aktualizována většina tabulek a dat.

Využitím příkazů, které jsou specifické pouze pro tento typ serveru je vyloučena možnost využití jiného databázového stroje.

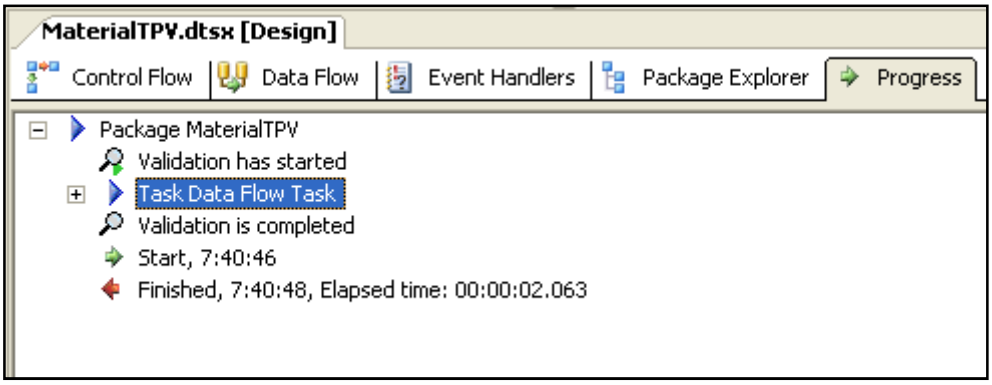
Příklady použití uložených procedur jsou uvedeny v přílohách 1 až 5.

5.1.4 Integrační služby

Integrační služby slouží především k transformaci dat z jiných zdrojů do prostředí serveru SQL 2005. Je však možné vytvářet transformační balíčky i uvnitř databázového stroje MS SQL serveru. Návrhy jsou plně vizuální, ale složitější změny je nutné psát. U jednoduchých si vystačíte s taháním obrázků. Výsledný návrh kopírování nebo transformace dat je pak zabalený do ikon. Rychlost je minimálně srovnatelná s rychlostí uložených procedur. Při transformaci můžete data podrobit složitým úpravám. Ve výsledku není vidět složitost jednotlivých etap, ale jen přehledný obrázek transformace dat. Pro ukázku je zde naprosto jednoduchý příklad, který zvládá aktualizaci dat stejně jako předchozí ukázka uložené procedury. Obrázek je uveden po spuštění s počtem načtených záznamů ve zdrojové databázi. Rychlost aktualizace je uvedena na dalším obrázku. Je vyšší než očekávaná od klikacích programů.



Obrázek 4. Import dat pomocí integračních služeb



Obrázek 5. Výsledek importu pomocí integrační služby

5.1.5 Aktualizace dat

Databáze aplikace je v tabulkách docela striktně omezena cizími klíči. Při aktualizaci je nutné zachovat pořadí upravovaných tabulek. Pokud jej nezachováme, dojde k chybě a data nebudou v pořádku. Vlastní pořadí aktualizace lze jednoduše vyčíst z návrhu databáze. Dodržení pořadí aktualizací však samo o sobě nezaručí bezchybné provedení této akce. Pokud například budeme odstraňovat z tabulky záznamy, může na ně odkazovat jiná tabulka. Záznam pak není možné odstranit a aktualizace skončí s chybou. Tuto možnost lze vyřešit s použitím některých vlastností cizích klíčů. Pak ale předáte část úprav do pravomoci SQL serveru a ztratíte nad ní kontrolu. Tyto kontrolu a úpravy mají jednoznačně definované chování, ale v případě chyby v informačním systému dojde ke znehodnocení dat poskytovaných aplikací. Navíc tento krok ještě komplikuje historizace některých záznamů. Pokud dojde k historizaci dat uvnitř jedné tabulky například se upraví datum platnosti, pak se situace zjednoduší. Data není nutné mazat a nedochází ke ztrátě integrity dat. Pokud ovšem data přesunuji do historických tabulek, situace je komplikovanější. Jsou možné dva případy. V prvním případě dojde ke kopii dat do historických tabulek a úpravě existujícího záznamu. Zde je situace jednoduchá a tato úprava naopak nepřináší žádné komplikace. Druhá možnost představuje vlastní smazání řádku v tabulce a jeho přesun do historie již tak primitivní není. Pokud na tento záznam odkazují záznamy z jiné tabulky, nelze ho jednoduše smazat. Musíme proto rozdělit aktualizaci na část, která vkládá nové data, a část, která maže existující data v tabulkách. První část dodržuje pořadí aktualizací podle schématu databáze a druhá část pak pokračuje mazáním dat v obráceném pořadí, než proběhlo vkládání dat. U tohoto způsobu nám výskyt chyby ukazuje na nedostatek ve zdrojových datech.

Jinou možností bylo data načítat vždy v aktuálním stavu a zastaralá data jednoduše smazat. Pro účely aplikace by tento způsob vedení dat postačoval, jednoduše by se navrhnul a realizoval pomocí integračních služeb. Nepřinesl by komplikace při změnách struktury tabulek ve zdrojové databázi, protože úprava v integračních službách je rychlá a jednoduchá. Ztratil by však možnost jednoduché historizace údajů, z nichž některé nejsou vedeny v databázi informačního systému. Přitom se může jednat o zajímavá data, která by umožnila přinejmenším zajímavé analýzy.

5.1.6 Aktuální a historická data

Pokud vedeme aktuální a historická data v jedné tabulce, pak pracovat s touto tabulkou není jednoduché a naopak docela komplikuje tvorbu dotazů do databáze. Podobná situace je i v případě dvou oddělených tabulek pro historická a aktuální data. Proto se zde nabízí jiný postup, než tvořit komplikované dotazy.

Pokud vedeme odděleně záznamy v historických i v aktuálních tabulkách, do pohledu je sloučíme. Odpadá spousta práce s komplikovanou údržbou dotazů a pohled bude navíc využitý pro celou aplikaci. Jiný typ pohledu pak vytvoříme pro tabulku obsahující oba typy dat. Pohled bude obsahovat jen aktuální data. Dotazy, které pak směřují na tuto část dat, nebudou komplikované a podstatně se zvýší jejich přehlednost. Za tuto výhodu však zaplatíme o něco delší dobou zpracování takto postaveného dotazu. Vzhledem k objemu dat však půjde maximálně o desetiny sekund až sekundy, a to je i pro rychlé zpracování dat přijatelné zdržení.

Pohledy však nemusí být vždy tím nejpohodlnějším řešením. Lze použít i uživatelsky definované tabulkové funkce nebo procedury. Tyto však nejsou tak obecně využitelné jako pohledy.

5.1.7 Omezení přístupu k datům

Všechna data mají určitou hodnotu. Některá data mají jen hodnotu práce vynaloženou při jejich získávání nebo ukládání. Například z pohledu firmy je to řádek s informacemi o prodeji výrobku odběrateli. Z pohledu konkurenčních firem má už i jeden řádek určitou cenu. Pokud však máme těchto informací více, je jejich hodnota podstatně vyšší a nejen pro konkurenční firmy. Podle hodnoty a vypovídacích schopností dat umožňujeme přístup jednotlivých uživatelů k jejich obsahu případně k jejich změnám. Není vhodné, aby každý měl přístup ke všemu. Pro odepření přístupu existují dva důvody, jeden z důvodů ceny informací a druhý z důvodu poškození dat.

Uživatelé po přihlášení jsou rozděleni do tří skupin. Každé skupině odpovídá jedna role. Uživatelské role se však mohou vzájemně kombinovat. Po určité době provozu bude přistoupeno k úpravám rolí případně k jejich menší diferenciaci. Pro začátek se počítá s rolemi manager, prodejce a technolog. Každá role odpovídá pracovní náplni dané pracovníkem zařazeným do této skupiny práv.

Role manager má přístup ke všem informacím v aplikaci bez omezení. Může upravovat hodnoty, které jsou editovatelné.

Role prodejce umožní zobrazit a editovat záznamy týkají se obchodu a odběratelů. Nemá již možnost zasahovat do technologických informací a nejsou mu poskytnuty vybrané analytické informace.

Role technolog povolí přístup k technologickým informacím, základním informacím o stavu skladů, prodeji a plánování výroby.

SQL server umožňuje omezení práv na schémata tabulky a jejich sloupce. Neumožní vám nastavit omezení na řádky. Abychom dosáhli různého omezení na řádky stejné tabulky, musíme vytvořit pohledy, které uživateli dokážeme zobrazit jen pro čtení anebo i pro změnu. Důvodem proč tento přístup požadujeme, je rozlišení více uživatelů z více společností. Pokud nastaví uživatel jedné společnosti data určitým způsobem, nechce, aby toto nastavení upravoval někdo jiný než uživatel stejné společnosti. Vzhledem k součinnosti několika společností, ale nemá zájem o utajování těchto skutečností. Je naopak v zájmu ostatních společností tyto informace sdílet a poskytovat si je vzájemně mezi sebou.

5.2 Uživatelské rozhraní

Uživatelské rozhraní je pro úspěšné nasazení aplikace velmi důležitým článkem. Pokud uživatel nebude mít možnost jednoduchého ovládání, pak aplikaci nebude používat a nebude ani úspěšná. Ikdyž bude mít výborné zpracování ostatních částí, aplikace bez uživatelů, je jen ztráta času.

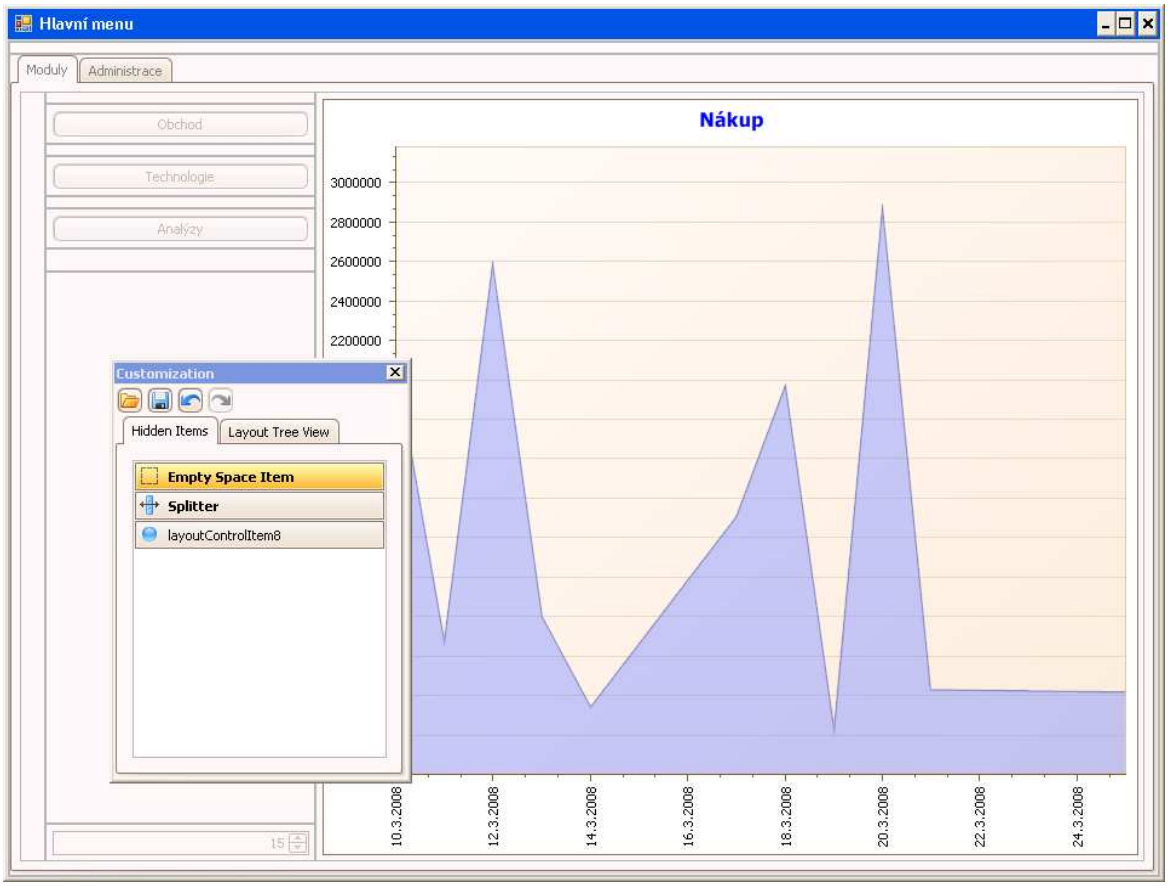
Rozhraní budou dominovat ovládací prvky, které lze jednoduše ovládat pomocí myši. Osobně tento styl aplikace nemám v oblibě. Preferoval bych jednoduché ovládání pomocí klávesnice s minimem používání myši. Naprostá většina uživatelů však klávesnici pro práci používá jen jako psací stroj. Tím je také limitováno její použití, co by pomůcky pro ovládání aplikací.

Aplikace se bude svým vzhledem a ovládáním blížit MS Office ve verzi 2007. Tento software je rozšířený a většina uživatelů s jeho ovládáním nemá problémy. Například velká část aplikace věnovaná obchodu se bude podobat Outlooku 2007. Plánování schůzek bude shodné s prací v Outlooku 2007, ale bude podstatně variabilnější.

Hlavní okno aplikace bude sloužit pouze pro spouštění dalších modulů, zbytek plochy okna může být vyplněn obecně dostupnými statistikami případně jen pozadím s obrázkem. Ve skutečnosti půjde o aplikaci, která nastaví práva a spustí modul již s omezením pro daného pracovníka. Přidání dalšího modulu pak nepřináší problémy do ostatních částí. Způsobí jen nepatrnou úpravu hlavního okna a nutnost odladění nově přidaného modulu.

Vzhled aplikace bude libovolně nastavitelný podle požadavků uživatele, všechny prvky lze nastavit podle potřeb uživatele. Jde o standardní možnost aplikací z Visual studia. Tuto vlastnost budou využívat hlavně jednotlivé moduly. Uživatel podle svého zaměření nebo okamžité situace zvolí, který prvek formuláře bude dominantní.

Obrázek 6. Jedna z možných úprav spouštěcího formuláře



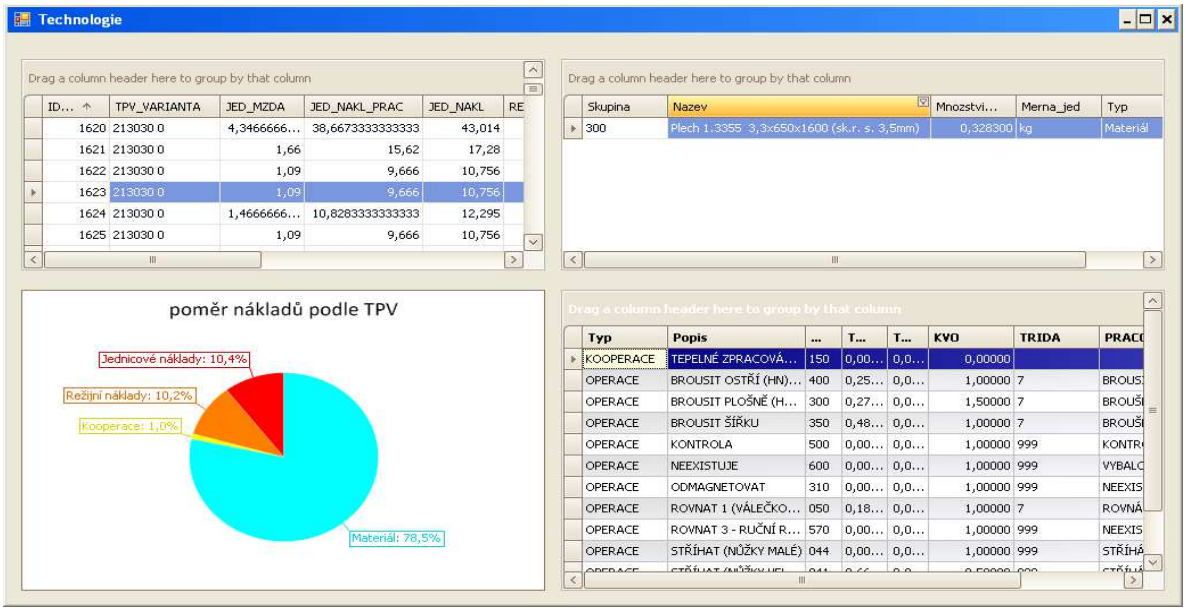
Obrázek 7. Možnost úpravy vzhledu za běhu aplikace

5.2.1 Formuláře

Aplikace bude pro začátek složena z pěti formulářů. Z hlavního formuláře budou pak dále spouštěny jednotlivé části aplikace. Hlavní formulář ponese i zabezpečení celé aplikace. Pokud to nebude nutné, bude celá aplikace pracovat bez přihlášení. Zabezpečení bude nastaveno podle jména uživatele přihlášeného v doméně. Aplikace bude mít za úkol oddělit uživatele jednotlivých společností a předložit jim odpovídající data a postupy. Postupy se mohou lišit mezi jednotlivými společnostmi nebo i výrobními linkami uvnitř společnosti.

Hlavním úkolem formuláře obchod bude sledování schůzek s obchodními partnery, přehled jednání a nabídek Výsledky jednání budou uloženy opět v aplikaci. Nabídky, plánované schůzky i jejich výsledky bude možné vytisknout do sestavy.

Formulář Analýzy bude obsahovat data z OLAP výpočtů. Bude přístupný managementu a některé části obchodníkům a technologům.



Obrázek 8. Možný vzhled formuláře TPV

Technologická část je zaměřena na výpočty cen výrobků. Jak plánované, tak ty, které byly dosaženy v jednotlivých výrobních postupech.

Administrativní část aplikace je využívána při správě uživatelských účtů a jejich nastavení. Čtení chybových zpráv a základnímu přehledu o databázi aplikace.

5.2.2 Možnosti prvků

Ovládací prvky z nadstavby Devexpress mají rozhraní v anglickém jazyce a komunikují s uživatelem tímto jazykem. Pro většinu zaměstnanců by to nebyl problém, část pracovníků však angličtině nerozumí. Proto byly všechny názvy a komunikace nadstavby Devexpress přeloženy do českého jazyka. Samotná práce na překladu je poměrně časově náročnou činností, ale slibujeme si od ní přívětivější uživatelské rozhraní.

Možnosti některých prvků byly hlavním důvodem, proč byla vybrána právě tato nadstavba. Mřížka (grid) pro zobrazování údajů z databáze je podstatně bohatší na funkce a možnosti než původní grid visual studia, upravovaný několik desítek hodin vlastními silami.

5.2.3 Uživatelské prvky

Používání ovládacích prvků visual studia anebo z nadstavby Devexpress, by vedlo k nutnosti použít znova stejné formuláře případně jejich části. Pro snížení opakování formulářů nebo jejich částí, budou základní používané části aplikace umístněny do User

Control prvků s jednoduchým rozhraním. Tyto bude možno volat odkudkoliv, kde budou k dispozici vstupní data pro tento prvek. Zda bude tento prvek také zobrazen, bude záležet jen na uživateli. Je možné obecně vložit do formuláře všechny prvky, které se vztahují k danému tématu. Není je však nutné zobrazit. Vzhledem k jednoduchosti, s jakou může uživatel měnit vzhled aplikace, anebo vložit do formuláře a vhodně uspořádat, nehrozí riziko přeplácaného vzhledu formulářů aplikace.

Výhodou takto navrženého prvku je jedna oprava chyby. Pak není nutné dohledávat chyby v jednotlivých částech. Stejně je na tom přidávání nových funkčností, ale naopak i chyby user control prvku. Chyba se projeví v každém jeho použití. Přesto si myslím, že jde spíše o klady tohoto způsobu znovupoužití části kódu.

User kontrol prvky budou základem všech formulářů. Proto jejich vývoj a odladění bude trvat nejdéle. Čas strávený vývojem těchto prvků by se měl vrátit v jednodušší údržbě aplikace a její snadnější rozšiřitelnosti, přesto že se s jejím dalším vývojem moc nepočítá. Rozšíření by bylo aktuální v případě zpoždění vývoje informačního systému.

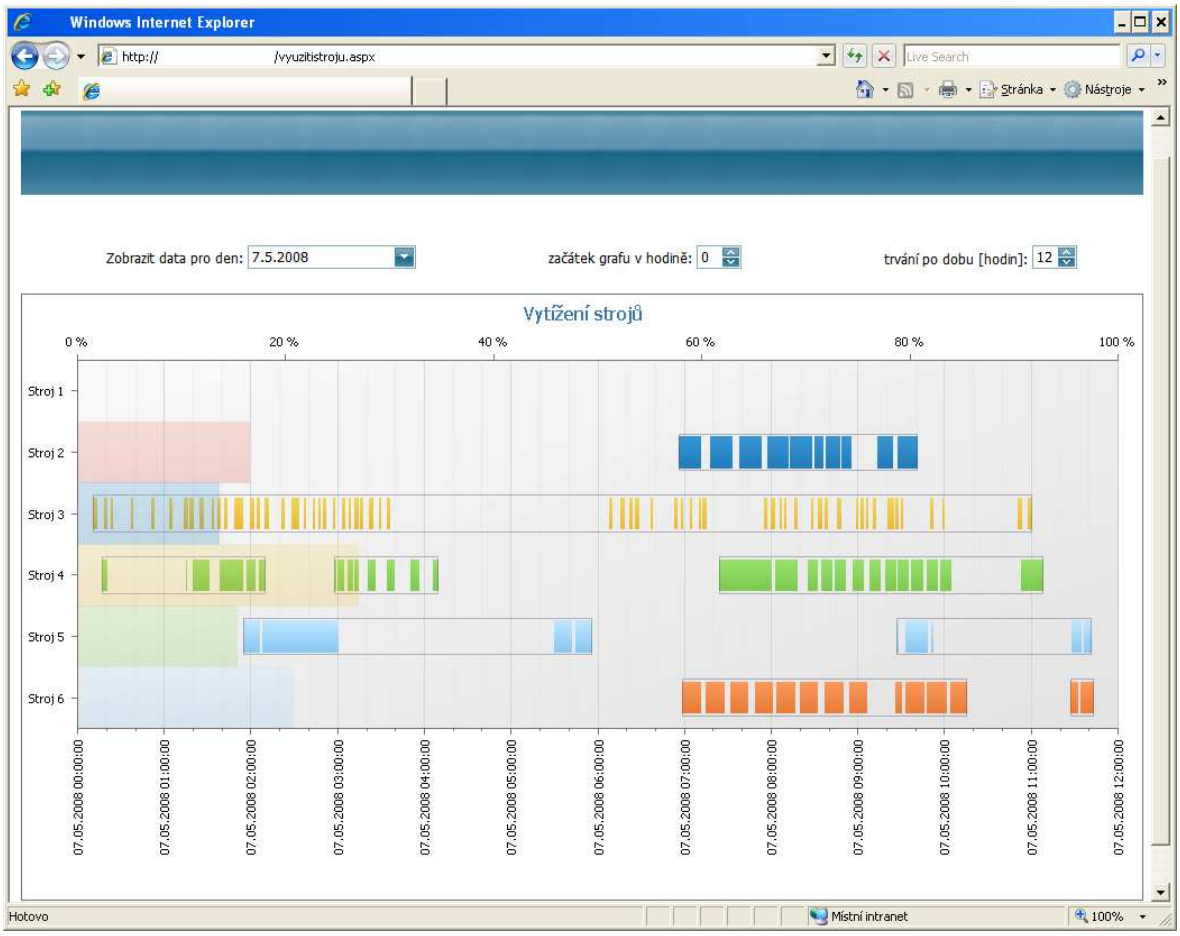
5.2.4 Možnosti exportů

Součástí ovládacích prvků je možnost exportu záznamů do sešitu MS Excel případně do tiskové sestavy. Výstup do Excelu je výhodný pro další využití nebo analýzy, podle potřeb jednotlivých uživatelů. Subjektivně však zastávám názor, aby veškeré potřebné a využitelné reporty vygeneroval již informační systém nebo v našem případě aplikace. Tvorba vlastních sestav je již suplováním činnosti informačního systému a ztrátou času zaměstnance. Ten by se měl věnovat čas vyhodnocení samotných podkladů nikoliv jejich tvorbě. Patrně nikdy nebude možné dosáhnout stavu, kdy není třeba vytvořit požadovanou sestavu ručně. Mělo by však k této situaci docházet výjimečně. Všechny sestavy, které se používají jednou za čtvrtletí, by měly být součástí informačního systému. Požadavky na nové sestavy budou v určitých mezích brány jako údržba systému.

5.2.5 Webové rozhraní

Část dat z aplikace se bude přenášet přímo do výroby. Hlavními požadavky proto jsou jednoduché rozhraní a jednoduché ovládání. Tato část se bude používat na starších typech počítačů. Tento předpoklad týkající se využití horší výpočetní techniky, nabízí zpracovat a zobrazit data přes webové rozhraní. Nevyžaduje příliš výkonný stroj a poskytuje aktuální

informace. Poškození počítače nepředstavuje problém, při obměně komponentů v administrativě je dostatek náhradních dílů nebo i celých počítačů.



Obrázek 9. Aplikace s webovým rozhraním

5.3 Reporty a výstupy

Aplikace umožňuje výstupy do několika běžných formátů. Tuto funkčnost není nutné u většiny prvků vytvářet, je poskytována jako vlastnost ovládacích prvků. A to není vše, další možností je tvořit sestavy za běhu aplikace. Otázkou je, zda bude výhodné, dát tyto nástroje do rukou koncovému uživateli. Pro část uživatelů by to byla jistě pozitivní změna, navíc bychom nemuseli modifikovat sestavy.

6 VYHODNOCENÍ APLIKACE

6.1 Stav vývoje aplikace

Pro vývoj aplikace bylo nutné projít všemi kroky vývoje aplikace. Analýza požadavků proběhla koncem loňského roku a přinesla seznam chybějících výstupů podle uživatelů. Pokud byl problém řešitelný prostředky informačního systému, pak proběhlo zapracování do informačního systému. Ostatní požadavky posloužily jako podklad pro funkčnost aplikace. V únoru bylo rozhodnuto o software, na kterém aplikace bude pracovat. Byly zahájeny práce na uživatelském rozhraní a databázové části aplikace. V březnu byly sestaveny některé části do uživatelských formulářů. Podle odezvy na rozhraní bude dále přistoupeno k jejich úpravám a změnám v návrhu. Konečná verze vzhledu a funkčnosti by měla projít testováním v průběhu června. S nasazením je pak počítáno od července.

6.2 Přínos aplikace

Důvodem pro vznik aplikace jsou požadavky z různých úrovní společností využívajících aktuální informační systém. Přínosem pro uživatele je každá informace, kterou neposkytuje informační systém nebo je nutné ji pracně zjišťovat. Aplikace neobsahuje velký počet zanoření právě z důvodu rychlého poskytování informací. Uživatelské rozhraní jde upravit za běhu aplikace a tím umožňuje přehledné zobrazování žádaných informací pro činnost uživatelů.

6.3 Náklady na vytvoření aplikace

Náklady na vývoj aplikace nenesou společnosti využívající informační systém. Pokud se společnost rozhodne pro její využití, budou výdaje související s vývojem aplikace započítány do provozních nákladů. Vývoj aplikace se neprojevil snížením objemu času věnovanému údržbě informačního systému, protože byl vyvíjen ve volném čase po splnění požadavků na informační systém nebo jeho úpravu. Tato skutečnost však nepříznivě ovlivnila celkovou dobu vývoje. Čas strávený nad aplikací bohužel neevidujeme. Možná je to chyba, pokud by se naskytla v budoucnu stejná příležitost, nebudeme mít v ruce přesný podklad, zda je vývoj aplikace tohoto typu výhodný nebo nevýhodný. Takto zůstane jen na zvážení, jestli částka získaná za provoz aplikace je nebo není dostatečně vysoká. Dalším závěrem z takto získaných informací mohlo být směřování vlastní činnosti. Jestli se

budeme věnovat jen správě informačního systému nebo se budeme věnovat i vývoji doplňkových aplikací a zda je to finančně zajímavé.

6.4 Software pro vývoj

Další částkou nutnou pro vývoj byl software. Pro vývoj byl použit databázový systém SQL server 2005 v půl roční testovací verzi. Po vypršení zkušební doby bude vyzkoušena verze 2008 a její nové možnosti. V ostrém provozu nebude nutné kupovat novou licenci, protože společnosti na databázovém serveru od Microsoftu již provozují informační systém.

Vývojové prostředí Visual studio 2005 ve verzi Profesional bylo pořízeno za částku dvacet tisíc korun. Z licenčních důvodů vývojového prostředí, byla částka vynaložená na získání tohoto produktu nutná. S ohledem na kurz dolaru, nákup přes internet v zahraničí by byl asi o čtvrtinu levnější. Musela by se však prověřit licenční politika a platnost licence takto získaného produktu.

Na začátku se počítalo s vlastní úpravou ovládacích prvků. Tato cesta byla zrušena z důvodu vysoké časové náročnosti na úpravu prvků. Po zvážení možností nabízených třetími stranami byla cesta vlastní úpravy prvků zrušena. Cena vlastní úpravy jen několika málo ovládacích prvků podstatně převyšuje náklady vynaložené na získání celých souborů ovládacích prvků. Vybrané řešení od firmy Devexpress je zajímavé tím, že za částku 1300 amerických dolarů nabízí nejen upravené ovládací prvky, ale navíc i jejich zdrojový kód. Možnosti úprav takto získaných prvků jsou pak velmi zajímavé. Na druhou stranu není důvod modifikovat prvky, pokud se v nich nevyskytují chyby. Ovládací prvky lze využít i mimo aplikaci, například na internetových nebo i intranetových stránkách.

6.5 Provozní náklady

Aplikace byla postavena na náklady tvůrců, proto cena provozu bude zahrnovat částku vynaloženou na vývoj aplikace tak částku na údržbu aplikace. Přímé náklady na pořízení vývojového software byly 40 tisíc korun. Předpokládaná doba životnosti aplikace je 2 roky. Měsíční předpokládaná částka pro zaplacení nákladů na software je 1700,- korun. K této částce se připočtou pak náklady za správu a údržbu aplikace. Částka se možná jeví vysoká, ovšem vůči nákladům na provoz informačního systému je již poměrně nízká. Dále je nutno připočítat náklady na hardware a jeho údržbu. Protože nebude pořizován žádný nový hardware, bude tato položka řádově nižší než částka na správu a údržbu aplikace.

6.6 Hardware

Testovací databáze a aplikace jsou umístněny na běžném počítači, kterému byla rozšířena operační paměť na 8 GB, a dvou jádrový procesor byl nahrazen čtyř jádrovým. Cenový rozdíl vůči běžnému kancelářskému počítači představuje částku kolem čtyř tisíc korun. Počet připojených uživatelů je limitován použitým operačním systémem. Po odladění bude aplikace přesunuta na jeden ze serverů společnosti. Například databázový server informačního systému pracuje bez velkého zatížení při konfiguraci 8 procesorů Intel Xeon 2,8 GHz a 24GB RAM.

6.7 Údržba aplikace

Údržba aplikace bude spadat pod správu informačního systému. Úprava funkčnosti a odstranění chyb se bude řídit stejnými pravidly jako informační systém. Požadavky na úpravu bude nutné schválit vedoucími pracovníky. Předpokládaná časová náročnost na správu a údržbu je kalkulována v rozmezí 10-15 hodin měsíčně.

6.8 Nevýhody aplikace

Pokud uživatelé budou využívat výstupy z nové aplikace, můžeme být spokojeni. Pokud se podíváme na problém z druhé strany, právě výhoda spokojeného pracovníka může být do budoucna velkým problémem. Aplikace není součástí informačního systému, pouze z něj čerpá data. Je tak druhým programem, který slouží pro zpracování dat. V případě úpravy informačního systému je nutná jeho úprava. Rozdělení na 2 programy je velkou nevýhodou. Další nevýhoda je používání dvou různých programů pro zpracování a vyhodnocení dat. Pro uživatele je jistě nepohodlné používání více programů. Tuto skutečnost ovšem vykompenzuje rychlost a množství získaných údajů. Nevýhodou je taktéž těsná vazba na tvůrce této aplikace. U informačního systému je velká pravděpodobnost vývoje i v následujících obdobích. V případě aplikace je možné úplné zastavení vývoje a v horším případě její ztráta funkčnosti ze dne na den. V případě nového správce informačního systému patrně bude rozhodující čas na pochopení funkčnosti a logiky aplikace a jeho snaha pokračovat alespoň v údržbě programu.

ZÁVĚR

Cílem této práce bylo vytvořit aplikaci pro získávání údajů, které neposkytuje informační systém nebo je poskytuje nevhodnou rychlostí. Aplikace není v době psaní této práce plně funkční. K tomuto cíli se ovšem blíží. Byla zpracována analýza požadavků uživatelů informačního systému, jejich návrhy pak byly použity pro základní rozdělení problémů. Pokud byl problém jednoduše implementovatelný do informačního systému, pak tato možnost byla plně využita. Ostatní případy pak posloužili pro vytvoření návrhu aplikace. Byla vyřešena otázka vývojových nástrojů včetně jejich financování a byla dokončena jejich lokalizace do českého jazyka. Finanční náklady zůstávají pod předpokládanou částkou. Tvorba aplikace je před dokončením user control prvků. S dalšími náklady se nepočítá. Předpokládaný provoz aplikace je začátkem druhého pololetí.

Při tvorbě aplikace bylo vyřešeno několik nedostatků v informačním systému a vytvořeny požadované sestavy nebo úpravy sestav. Tím se zpřesnila řada výstupů a jejich informační hodnota. Aplikace se v prvních testech osvědčila, co se týká rychlosti i rozsahu informací. Navíc bude objem informací ještě rozšířen. Ekonomickým přínosem budou přesnější podklady a odstranění některých administrativních činností. Přesnější podklad využijeme k lepšímu zjištění efektivity výroby i obchodu. Zároveň vznikla možnost rychlé analýzy problému nebo úzkých míst ve výrobní nebo obchodní části. Odstranění administrativních činností a jejich automatizace je přínosem pro pracovníky, kteří jí museli věnovat čas, ale představovala pro ně jen rutinní činnost.

Pro vývoj aplikace se počítalo s časem kratším o dva až tři měsíce. Zpoždění však přineslo vyšší variabilitu a širší možnosti aplikace. Tyto možnosti vyplynuly ze skutečností, které jsme před půl rokem ani neplánovali.

Další vývoj bude závislý na rychlosti zavádění požadovaných funkcí do informačního systému. V případě rychlého zavedení bude aplikace postupně ztrácet na významu. Pokud se však nepodaří začlenění některých vlastností anebo optimalizací prosadit včas její význam se zvýší.

CONCLUSION

Purpose of this work was to create an application for data acquisition, which are not provided by the information system or are provided too slow. The application is not fully functional at the time, but we are bringing it to the close. The users' requirements were analyzed; theirs proposals and problems were used for basic separation of the problems. We implemented the problem into the information system, when possible. Others cases were used as the basis of concept application. The development of the software, financing and translation to Czech language were solved. The costs of the solution are lower than expected amount. We are finishing the creation of the components and we do not expect another cost. Expected setting in operation of the application is in the beginning of the second half year.

Some problems with information systems were solved and requested dead reports were created or adjusted during the development of the application. A lot of reports were revised and improved. The first benchmark was succeeding in range of information and rate of application. The range of information is going to expand. The economic gains are the adjusted information, because they eliminate some administrative practices. We expect the increase of the efficiency of the production and business. Application enables the fast detection of the problems and bottlenecks at production or business. Minor administrative activity will be automated and better used by the employees.

Expected time for development of the application was two or three month shorter. The delay of the development brought more variability and larger range of application, although these possibilities were not planed six months ago.

Next development depends on upgrade and innovation of the information system. When information system will be implemented quickly, the application lost its significance. When information system will be implemented slowly, the application must support his insufficiency.

SEZNAM POUŽITÉ LITERATURY

[1] LACKO, L. – Business Inteligence v SQL Serveru 2005. Brno: Computer Press, 2003. s. 390.

[2] LACKO, L. – Oracle : Správa, programování a použití databázového systému. Brno: Computer Press, 2003. s. 480.

[3] SACK, J. – Velká kniha T–SQL & SQL Server 2005. Brno: Zoner Press, 2007. s. 864.

[4] PRICE, J. – C #– programování databází. Grada publishing, 2005. s. 624.

[5] Solid Quality Learnng – Microsoft SQL Server 2005. Brno: Computer Press, 2007. s. 320.

[6] LACKO, L. – Datové sklady analýza OLAP a dolování dat. Brno: Computer Press, 2003. s. 486.

[7] PETZOLD, C. – Windows v jazyce C# Svazek 1. Praha: SoftPress, 2003. s. 600.

[8] PETZOLD, C. – Windows v jazyce C# Svazek 2. Praha: SoftPress, 2003. s. 608.

[9] Online Documentation – http://www.devexpress.com/Support/OnlineHelp.xml

[10] MSDN Library – http://msdn2.microsoft.com/en-us/library/default.aspx

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

.NET	Platforma společnosti Microsoft
Business Intelligence	Transformace dat na informace a převod informací na poznatky
ERP	Podnikový informační systém (Enterprise Resource Planning)
Integrační služby	Služba poskytovaná MS SQL Server pro import a export dat
MS SQL Server	Databázový server od firmy Microsoft
OLAP	On Line Analytical Proscessing, databáze pro analýzy dat
SQL	Structured Query Language (strukturovaný dotazovací jazyk)
Visual Studio	Vývojové prostředí firmy Microsoft

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: ULOŽENÁ PROCEDURA PŘÍKLAD 1

```
USE [TPV]
-- ================================================
-- Author:       Kafka Pavel
-- Create date:  12.3.2008
-- Description:  Aktualizace tabulky s TPV materiály
-- ================================================
CREATE PROCEDURE [TPV].[UpdateOperaceTPVMat]
AS
BEGIN
    ------------------------------------------------------------------------------
    --vytvoření synonym pro tabulky z databáze informačního systému
    ------------------------------------------------------------------------------

    EXEC [dbo].[CreateSynonyms];
    ------------------------------------------------------------------------------
    --smazání materiálů které se už nepoužívají v technologickém postupu nebo
    --postup byl zrušen
    ------------------------------------------------------------------------------
      WITH DeleteOperaceMat (Varianta, Vyrobek, Skupina, Material, MnozstviMat,
                             TypMat) AS
          ( SELECT m.TPVVarianta, m.IDZboziVyr, m.Skupina, m.IDZboziMat,
                   m.MnozstviMat, m.TypMat
            FROM   OperaceTPVMat m
            EXCEPT
            SELECT RTRIM(LTRIM(t.Varianta)),t.CiN ,SUBSTRING(LTRIM(t.Sku),1,3),
                   t.CiP, t.Mnoz, z.PrKind
            FROM   Tpostup t, Zbozi z
             WHERE  t.CiP=z.Cis
            )
       DELETE FROM  OperaceTPVMat WHERE ID IN
           ( SELECT m.ID
             FROM   OperaceTPVMat m, DeleteOperaceMat d
             WHERE  d.Varianta=m.TPVVarianta AND d.Vyrobek=m.IDZboziVyr
                    AND d.Skupina=m.Skupina AND d.Material=m.IDZboziMat
           );
       ------------------------------------------------------------------------------
       --vložení materiálů z nových nebo upravených technologických postupů
       ------------------------------------------------------------------------------
       WITH InsertOperaceMat (Varianta, Vyrobek, Skupina, Material) AS
           (  SELECT RTRIM(LTRIM(t.Varianta)),t.CiN, SUBSTRING(LTRIM(t.Sku),1,3),
                   t.CiP
              FROM   Tpostup t, Zbozi z
               WHERE  t.CiP=z.Cis
              EXCEPT
              SELECT m.TPVVarianta, m.IDZboziVyr, m.Skupina, m.IDZboziMat
              FROM   OperaceTPVMat m
              )

       INSERT INTO OperaceTPVMat (TPVVarianta, IDZboziVyr, Skupina, IDZboziMat,
                    MnozstviMat, TypMat)
       SELECT RTRIM(LTRIM(t.Varianta)),t.CiN ,SUBSTRING(LTRIM(t.Sku),1,3),
              t.CiP, t.Mnoz, z.PrKind
       FROM   Tpostup t, Zbozi z, InsertOperaceMat i
       WHERE  RTRIM(LTRIM(t.Varianta))=i.Varianta AND t.CiN =i.Vyrobek
              AND SUBSTRING(LTRIM(t.Sku),1,3)=i.Skupina AND t.CiP=i.Material
              AND t.CiP=z.Cis;
    ------------------------------------------------------------------------------
    --smazání synonym
    ------------------------------------------------------------------------------
      EXEC dbo.DropSynonyms;
END
```

PŘÍLOHA P II: ULOŽENÉ PROCEDURY PŘÍKLAD 2

```
USE [TPV]
-- ================================================
-- Author:       Kafka Pavel
-- Create date:  28.2.2008
-- Description:  Aktualizace cen
-- ================================================
ALTER PROCEDURE [dbo].[UpdateCenyTPV]
AS
BEGIN
    ------------------------------------------------------------------------------
    --vytvoření synonym pro tabulky z databáze informačního systému
    ------------------------------------------------------------------------------

    EXEC [dbo].[CreateSynonyms];
   ------------------------------------------------------------------------------
   --zneplatnění cen materiálů které se změnily od poslední aktualizace
   ------------------------------------------------------------------------------
      WITH CenyHistorizace (IDZbo,Cena) AS
           (  SELECT z.Cis, z.SkC
              FROM   Zbozi z
              WHERE  z.Typ_Zbo=1 AND (SUBSTRING(z.Flagy,4,1) & 4)/4 =0
              EXCEPT
              SELECT IDZbozi, CenaTPV
              FROM   ZboziCenyTPV
              WHERE  PlatnostDo IS NULL
           )
      UPDATE ZboziCenyTPV
                SET   PlatnostDo=z.DPZ,
                       IDUser=z.DPZ_Kdo
                FROM  ZboziCenyTPV zc, Zbozi z, CenyHistorizace h
                WHERE zc.IDZbozi=h. IDZbo AND PlatnostDo IS NULL AND
                      h. IDZbo =z.Cis;

      ------------------------------------------------------------------------------
      --vložení nových nebo upravených cen materiálů
      ------------------------------------------------------------------------------
      WITH CenyAktualizace (IDZbo) AS
            (SELECT z.Cis
             FROM   Zbozi z
             WHERE  z.Typ_Zbo=1 AND (SUBSTRING(z.Flagy,4,1) & 4)/4 =0
             EXCEPT
             SELECT IDZbozi
             FROM   ZboziCenyTPV
             WHERE  PlatnostDo IS NULL
            )

      INSERT INTO ZboziCenyTPV (IDZbozi, CenaTPV, PlatnostOd, IDUser)
                SELECT z.Cis, z.SkC, z.DPZ, z.DPZ_Kdo
                FROM   Zbozi z, CenyAktualizace a
                WHERE  a.IDZbo=z.Cis;

      ------------------------------------------------------------------------------
      --neplatne zaznamy ve zdrojove DB, ktere neexistuji v DB aplikace
      --datum platnosti bude nastaven od predchazejici aktualizace do prave
      --provadene aktualizace
      ------------------------------------------------------------------------------

     DECLARE @datumAkt DATETIME;
     SET @ datumAkt =(SELECT p.Datum
                      FROM   Parametry p
                      WHERE  p.Zkratka='AKTUALIZACE');
```

```
      WITH CenyHistorie (IDZbo) AS
            (SELECT z.Cis
             FROM   Zbozi z
             WHERE  z.Typ_Zbo=1 AND (SUBSTRING(z.Flagy,4,1) & 4)/4 =1
             EXCEPT
             SELECT DISTINCT IDZbozi
             FROM ZboziCenyTPV
             WHERE platnostDo IS NOT NULL
            )
       INSERT INTO ZboziCenyTPV (IDZbozi, CenaTPV, PlatnostOd, PlatnostDo,
                                 IDUser)
              SELECT z.Cis, z.SkC, @datumAkt, z.DPZ, z.DPZ_Kdo
              FROM   Zbozi z, CenyHistorie h
              WHERE  h.IDZbo=z.Cis;

    ------------------------------------------------------------------------------
    --smazání synonym
    ------------------------------------------------------------------------------
    EXEC dbo.DropSynonyms;

END
```

PŘÍLOHA P III: VYTVOŘENÍ SYNONYM SQL SERVER

```
USE [TPV]
-- ================================================
-- Author:       Kafka Pavel
-- Create date:  03.2.2008
-- Description:  Vytvoreni synonym
-- ================================================
CREATE PROCEDURE [dbo].[CreateSynonyms]

AS
BEGIN
    ------------------------------------------------------------------------
    --vytvoření synonym pro tabulky z databáze informačního systému
    --po přidání další položky je nutné upravit proceduru DropSynonyms
    ------------------------------------------------------------------------
    --umožní pracovat s tabulkami informačního systému stejně jako s vlastními
    --tabulkami
    ------------------------------------------------------------------------
    CREATE SYNONYM Ciselniky
       FOR [server].[databaze].[dbo].[Ciselniky];
    CREATE SYNONYM Postup
       FOR [server].[databaze].[dbo].[Postup];
    CREATE SYNONYM Zbozi
       FOR [server].[databaze].[dbo].[Zbozi];
    CREATE SYNONYM Zakaznik
       FOR [server].[databaze].[dbo].[Zakaznik];

END
```

PŘÍLOHA P IV: SMAZÁNÍ SYNONYM SQL SERVER

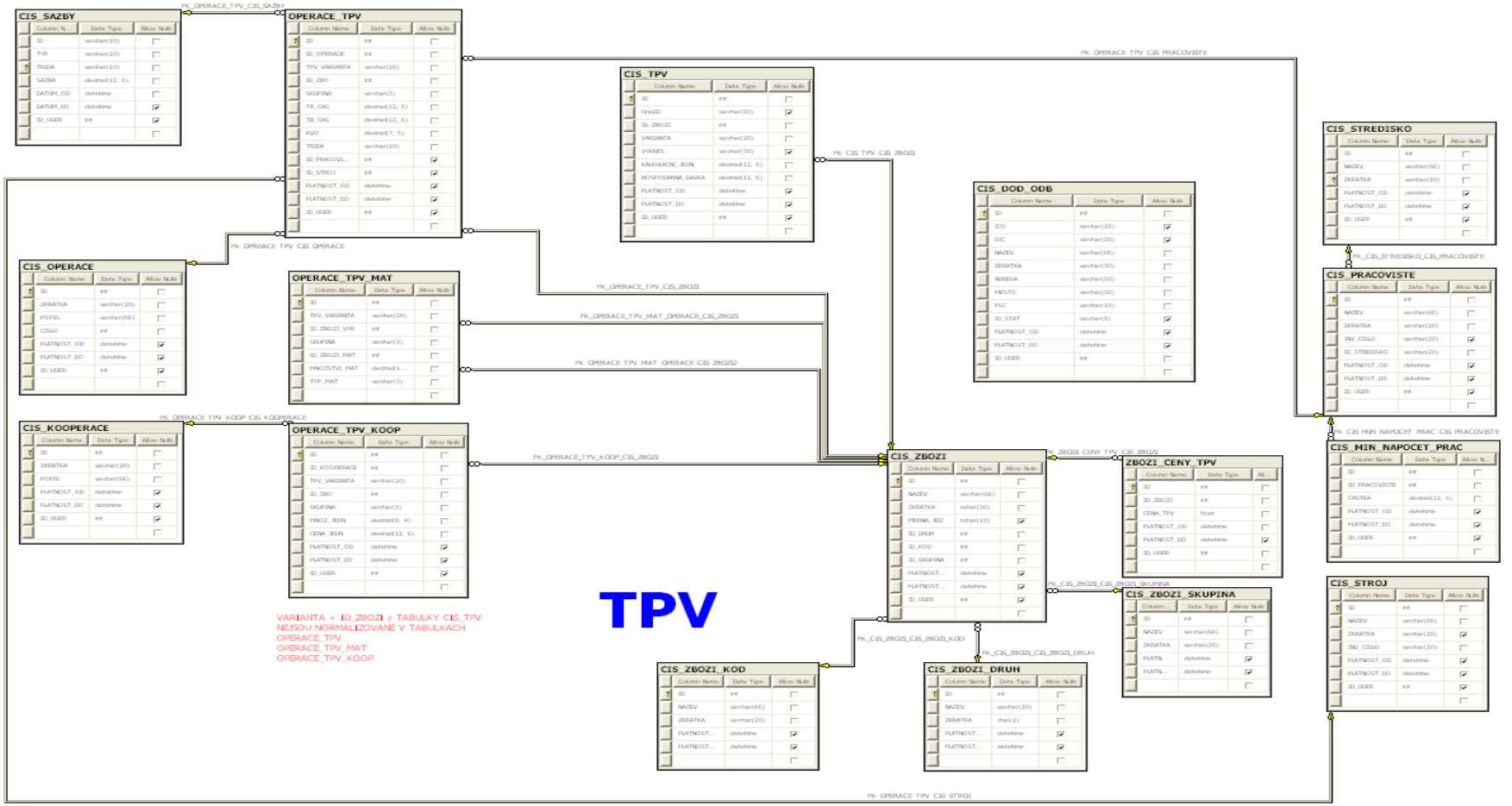
```
USE [TPV]
-- ================================================
-- Author:       Kafka Pavel
-- Create date:  03.2.2008
-- Description:  Vytvoreni synonym
-- ================================================
CREATE PROCEDURE [dbo].[DropSynonyms]

AS
BEGIN
    ------------------------------------------------------------------------
    --upravit podle CreateSynonyms, aby tu nevyselo neco zbytecne
    ------------------------------------------------------------------------
    --zrušení synonym dotazy na databázi inf. systému přestanou fungovat
    ------------------------------------------------------------------------
    DROP SYNONYM Ciselniky;
    DROP SYNONYM Postup
    DROP SYNONYM Zbozi;
    DROP SYNONYM Zakaznik;
END
```

PŘÍLOHA P V: DEFINICE POHLEDU NA AKTUÁLNÍ DATA

```
USE [TPV]
-------------------------------------------------------------------------
--lze řešit i uloženou procedurou
--pohled zvolen kvůli jednodušši práci s daty
-------------------------------------------------------------------------
CREATE VIEW [dbo].[AktualniCenyZbozi]
AS
SELECT c.IDZbozi, c.Cena
FROM   dbo.ZboziCeny c,
       (SELECT   IDZbozi, MAX(PlatnostOd) AS PosledniCena
        FROM     dbo.ZboziCeny
        GROUP BY IDZbozi) ac
WHERE c.IDZbozi = ac.IDZbozi AND c.PlatnostOd = ac. PosledniCena
```

PŘÍLOHA P VI: SCHEMA DATABÁZE PRO TECHNOLOGII



PŘÍLOHA P VII: KLIKACÍ PROGRAMOVÁNÍ VS 2005

```
private void InitializeComponent()
    {
       DevExpress.XtraCharts.PaletteEntry paletteEntry1 = new
                        DevExpress.XtraCharts.PaletteEntry(System.Drawing.Color.Red, System.Drawing.Color.Red);
       DevExpress.XtraCharts.PaletteEntry paletteEntry2 = new
                        DevExpress.XtraCharts.PaletteEntry(System.Drawing.Color.Orange, System.Drawing.Color.Orange);
       DevExpress.XtraCharts.PaletteEntry paletteEntry3 = new
                        DevExpress.XtraCharts.PaletteEntry(System.Drawing.Color.Yellow, System.Drawing.Color.Yellow);
       DevExpress.XtraCharts.PaletteEntry paletteEntry4 = new
                        DevExpress.XtraCharts.PaletteEntry(System.Drawing.Color.Lime, System.Drawing.Color.Lime);
       DevExpress.XtraCharts.PaletteEntry paletteEntry5 = new
                        DevExpress.XtraCharts.PaletteEntry(System.Drawing.Color.Aqua, System.Drawing.Color.Aqua);
       DevExpress.XtraCharts.PaletteEntry paletteEntry6 = new
                        DevExpress.XtraCharts.PaletteEntry(System.Drawing.Color.Blue, System.Drawing.Color.Blue);
       DevExpress.XtraCharts.PaletteEntry paletteEntry7 = new
                        DevExpress.XtraCharts.PaletteEntry(System.Drawing.Color.Purple, System.Drawing.Color.Purple);
       DevExpress.XtraCharts.PaletteEntry paletteEntry8 = new
                        DevExpress.XtraCharts.PaletteEntry(System.Drawing.Color.Black, System.Drawing.Color.Black);
      DevExpress.XtraCharts.Palette palette1 = new DevExpress.XtraCharts.Palette("Zakladni",
                        DevExpress.XtraCharts.PaletteScaleMode.Repeat, new
                        DevExpress.XtraCharts.PaletteEntry[] {   paletteEntry1,
                                                                 paletteEntry2,
                                                                 paletteEntry3,
                                                                 paletteEntry4,
                                                                 paletteEntry5,
                                                                 paletteEntry6,
                                                                 paletteEntry7,
                                                                 paletteEntry8});
       DevExpress.XtraCharts.Series series1 = new DevExpress.XtraCharts.Series();
       DevExpress.XtraCharts.PieSeriesView pieSeriesView1 = new DevExpress.XtraCharts.PieSeriesView();
       DevExpress.XtraCharts.ChartTitle chartTitle1 = new DevExpress.XtraCharts.ChartTitle();
       this.graf = new DevExpress.XtraCharts.ChartControl();
       ((System.ComponentModel.ISupportInitialize)(this.graf)).BeginInit();
       ((System.ComponentModel.ISupportInitialize)(series1)).BeginInit();
```

```
    ((System.ComponentModel.ISupportInitialize)(pieSeriesView1)).BeginInit();
    this.SuspendLayout();
         //
    // graf
    this.graf.Dock = System.Windows.Forms.DockStyle.Fill;
    this.graf.Legend.Visible = false;
    this.graf.Location = new System.Drawing.Point(0, 0);
    this.graf.Name = "graf";
    this.graf.PaletteName = "Zakladni";
    this.graf.PaletteRepository.Add("Zakladni", palette1);
    series1.Name = "rada";
    series1.View = pieSeriesView1;
    series1.PointOptionsTypeName = "PiePointOptions";
    this.graf.SeriesSerializable = new DevExpress.XtraCharts.Series[] {series1};
    this.graf.SeriesTemplate.PointOptionsTypeName = "PiePointOptions";
    this.graf.Size = new System.Drawing.Size(640, 400);
    this.graf.TabIndex = 0;
    chartTitle1.Font = new System.Drawing.Font("Tahoma", 12F, System.Drawing.FontStyle.Bold);
    chartTitle1.Text = this.Nadpis;
    chartTitle1.TextColor = System.Drawing.Color.Beige;
    this.graf.Titles.AddRange(new DevExpress.XtraCharts.ChartTitle[] {chartTitle1});
    this.graf.Click += new System.EventHandler(this.graf_Click);
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.Controls.Add(this.graf);
    this.Name = "ucGraf";
    this.Size = new System.Drawing.Size(640, 400);
    ((System.ComponentModel.ISupportInitialize)(pieSeriesView1)).EndInit();
    ((System.ComponentModel.ISupportInitialize)(series1)).EndInit();
    ((System.ComponentModel.ISupportInitialize)(this.graf)).EndInit();
    this.ResumeLayout(false);
};
```