Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Fakulta ekonomická
Katedra aplikované matematiky a informatiky

Diplomová práce

Návrh databáze pro připojení systému SAP jako zdroje dat pro webovou aplikaci

Vypracoval: Bc. et Bc. Lukáš Marhoun
Vedoucí práce: doc. Ing. Ladislav Beránek, CSc.

České Budějovice 2016

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta ekonomická
Akademický rok: **2014/2015**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. et Bc. Lukáš MARHOUN**
Osobní číslo:	**E14736**
Studijní program:	**N6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Název tématu:	**Návrh databáze pro připojení systému SAP jako zdroje dat pro webovou aplikaci**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování:

Cíl práce: V teoretické části se práce zaměří na potřebné části z teorie relačních databází a jazyka SQL, možnosti propojování databázových systémů se zaměřením na propojení ERP SAP a databáze MS SQL Server včetně jazyka T-SQL. V praktické části bude popsána aplikace pro extrakci dat do webové aplikace. Dále bude provedena aplikace pro potřeby vybraného strojírenského podniku.

Metodický postup:

1. Studium odborné problematiky zaměřené na relační databáze a jazyk SQL, zejména se zaměřením na možnost propojování databázových systémů s ERP SAP a jazyk T-SQL.
2. Analýza využití databází při tvorbě webové stránky - zejména oddělení logiky od uživatelského rozhraní.
3. Analýza a realizace aplikace pro extrakci dat do webové aplikace.
4. Zhodnocení řešení, závěr.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **50 - 60 stran**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

1. **Krishnamoorthy, V., Murray, M. & Reynolds, N. (2012).** *SAP transaction codes: your quick reference to T-codes in SAP ERP.* **(1st ed.) Boston: Galileo Press.**
2. **Kühnhauser, K.-H. (2009).** *ABAP: výukový kurz.* **(Vyd. 1.) Brno: Computer Press.**
3. **Maassen, A. (2007).** *SAP R/3: kompletní průvodce.* **(Vyd. 1.). Brno: Computer Press.**
4. **Stanek, W. R. (2013).** *Microsoft SQL Server 2012: kapesní rádce administrátora.* **(1. vyd.) Brno: Computer Press.**
5. **Další materiály na webových stránkách společnost SAP.**

Vedoucí diplomové práce: **doc. Ing. Ladislav Beránek, CSc.**
Katedra aplikované matematiky a informatiky

Datum zadání diplomové práce: **9. ledna 2015**

Termín odevzdání diplomové práce: **15. dubna 2016**

doc. Ing. Ladislav Rolínek, Ph.D.
děkan

L.S.

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13 (26)
370 05 České Budějovice

prof. RNDr. Pavel Tlustý, CSc.
vedoucí katedry

V Českých Budějovicích dne 30. března 2015

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47 zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

4.9.2016

Rád bych poděkoval panu doc. Ing. Ladislavu Beránkovi, CSc. za vstřícnou podporu při vedení mé diplomové práce.

Obsah

1 Úvod

Téma této diplomové práce je určeno tím, že využitím podnikových informačních systémů a relačních databází se zabývám i pracovně. V poslední době je mé pracovní zaměření rozšířeno o přípravu databáze pro webové aplikace, které by využívaly data, která jsou běžně uložena v systému SAP.

K vývoji externích aplikací nebo řešení v podobě různých nástrojů v MS Excelu vede nedostatečná implementace podnikového informačního systému. Ta může být i záměrná, protože některé požadavky mohou být tak specifické, že by se je nevyplatilo zavádět centrálně. Někdy se přidávají i omezené možnosti reportingu či analýz prostřednictvím produktivního systému, který by byl takovými požadavky neúměrně zatěžován. Hlavní systém ale poskytuje aplikacím data, která by se jinak musela zadávat vícekrát anebo neustále kopírovat, což je zdrojem chyb, ale i ztížení práce uživatelů aplikací.

V práci budu popisovat možnosti připojení podnikového systému SAP k externímu intranetovému portálu s vlastními aplikacemi. Prostředí, ve kterém se bude úkol řešit, sestává ze systému SAP jako zdroje dat a lokálního databázového systému MS SQL Server jako databázové vrstvy pro webové aplikace tvořené v jazyce PHP. Pravidla i možnosti v práci uváděné lze ale uplatnit i při použití jiných systémů a programovacích jazyků.

Zabývat se budu téměř výhradně pozadím aplikací, minimálně potom prezentační vrstvou. Diplomová práce také není učebnicí a nebudu proto přepisovat matematický aparát, kterého je v citované literatuře dostatek, stejně jako důkazů. Zabývat se chci pouze tou částí teorie relačních databází, která souvisí s tématem práce a její praktickou částí – za více než 40 let se samozřejmě teorie rozvinula natolik, že by nebylo možné ji celou smysluplně a využitelně shrnout. Totéž bude platit pro systém SAP a MS SQL Server, kde nemá význam zabíhat do takových detailů, které bych v práci nevyužil.

1.1 Cíl práce

Cílem práce je nalézt a popsat řešení, které by doplnilo podnikový informační systém a odstranilo nevyhovující nástroje provozované v MS Excelu s často neaktuálními daty doplňovanými ručně. To by se zajistilo vhodným způsobem připojení k podnikovému systému jako jednoznačnému a spolehlivému zdroji základních informací firmy. Tato data mohou dále sloužit jako zdroj nových intranetových aplikací, ale i pro specifické reporty a analýzy. V takovém případě je následné zpřístupnění pro MS Excel vhodné, protože by s nimi mohl pracovat i běžný uživatel.

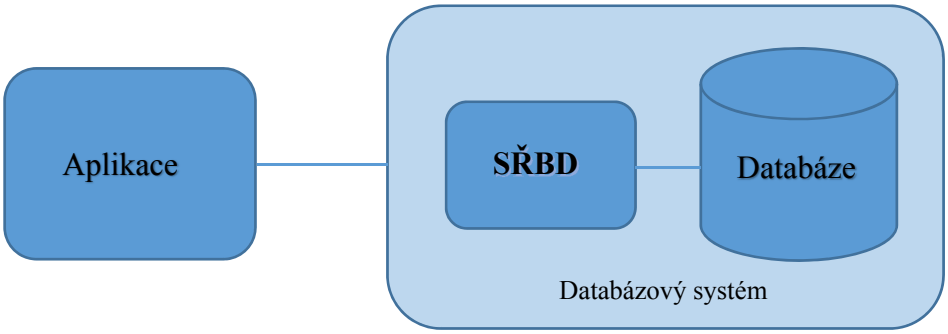
V teoretické části se chci v souladu se zadáním zaměřit na popis relačních databází a možnosti jejich propojovaní s důrazem na možné propojení produktivního podnikového systému s lokální webovou aplikací, dále na bližší popis systémů, pro které tento úkol budu řešit, zejména pak systém SAP a nástroje Business Intelligence. S ohledem na to, že relační databáze a databázové systémy jsou již mnohokrát a dostatečně popsány, bych chtěl připravit pouze nejnutnější přehled, ke kterému se vždy s popisem konkrétního postupu přidají další informace. Chtěl bych také v krátkém historickém přehledu uvést souvislosti a události, které vedly k podobě současných databázových či podnikových systémů a současně se vyhnout podrobné diskuzi o schématu, modelování databáze apod. V části věnované systému SAP chci rozebrat zejména aplikační vrstvu – to, co je pro systém SAP specifické. V části o MS SQL Serveru se naproti tomu chci zaměřit na specifika tohoto databázového systému a jazyka T-SQL.

Zároveň bych chtěl upozornit, že pravidla uvedená dále u relačních databází nemusejí platit pouze pro ně, týkají se zpravidla databázových systémů obecně. Vzhledem k tomu, že procesy a součásti databázových systémů, stejně jako jazyk, prostřednictvím kterého jsou ony nástroje používány, souvisí zejména s relačními databázemi, zařadil bych jednotlivé podkapitoly právě pod relační databáze.

2 Databázové systémy

Podniky a instituce spravují velké množství různých údajů o věcech, lidech či událostech. Takové seznamy nebo přehledy je nutné uchovávat tak, aby se v nich dalo rychle a přesně vyhledávat a aby bylo možné je spolehlivě udržovat ve stavu, který odpovídá skutečnosti. Často se k těmto požadavkům přidává i nutnost zachovávat historii změn v záznamech. Tuto úlohu plní databázové systémy, které oddělují správu dat od aplikací, které s nimi pracují. Samotné systémy sestávají z databáze - fyzického úložiště dat, a systému, který se pro práci s daty a jejich správu používá - systém řízení báze dat (SŘBD)[1].

Obrázek 1: Databázový systém



Prostřednictvím datového modelu, kterým se rozumí abstraktní přehled o datech a způsob, jakým se popisuje jejich obsah a vztahy mezi nimi (síť, graf, tabulka), se vytváří logická vrstva, se kterou pracuje uživatel, a která tvoří strukturu - schéma, což je popis konkrétního souboru dat založený na vybraném datovém modelu (CS145 Introduction to Databases, 2016). Fyzická vrstva jako úložiště strukturovaných dat potom zajišťuje úlohy podobné souborovému systému (defragmentaci, IO operace a další).

V databázích se neukládají pouze samotné údaje spravovaných objektů jako je přehled uživatelů, seznam zaměstnanců, ale i dokumenty, obrázky a také informace o samotné databázi - definice dat, reporty a procedury pro práci s daty nebo správu databáze.

Do dnešní podoby se systémy vyvíjejí již od poloviny 20. století stejně jako programovací jazyk pro práci s nimi. U toho stál na počátku stejný požadavek jako u ostatních programovacích jazyků - aby byl použitelný na různých typech počítačů a používal jazyk bližší člověku (angličtina) a ne stroji (instrukce strojového kódu). Výsledkem působení

[1] Podle anglického database management system (DBMS)

organizace CODASYL[2], která se tímto úkolem zabývala, byl mimo jiné standardy i dnes dále vyvíjený a používaný jazyk COBOL[3]. Koncem 60. let minulého století potom systémy pro ukládání a správu dat zastoupené zejména firmou IBM používaly převážně navigační modely dat, hierarchické a později síťové, zřejmě také vlivem hardwarového vybavení tehdejších počítačů[4]. Od těchto datových modelů se zcela neupustilo. Například IBM IMS[5] používá hierarchický model doposud.

[2]Conference on Data Systems Languages: http://discover.lib.umn.edu/cgi/f/findaid/findaid-idx?c=umfa;cc=umfa;rgn=main;view=text;didno=cbi00011
[3]COmmon Business Oriented Language: http://groups.engin.umd.umich.edu/CIS/course.des/cis400/cobol/cobol.html
[4] Bubnové paměti a pásky bez přímého přístupu
[5] https://www-01.ibm.com/software/data/ims/

2.1 Relační databáze

Tato práce se věnuje výhradně relačním databázím, které jsou pro podnikové systémy nejpoužívanější (CS145 Introduction to Databases, 2016) a prosazují se postupně od 80. let 20. století – zpočátku zejména prostřednictvím produktů společností Oracle (původně Relational software, Inc) a Sybase.

2.1.1 Relační model dat

V červnu 1970 publikoval[6] Edgar Frank Codd v té době odlišný pohled na data a jejich uspořádání v databázích. Popisuje stávající informační systémy založené na hierarchickém (stromovém) a síťovém uspořádání dat, uvádí jejich nedostatky spočívající v přílišné provázanosti dat a jejich skutečného způsobu ukládání a představuje nový pohled na data, datový model a jejich ukládání v databázích s hlavní myšlenkou, kterou je oddělení interní reprezentace dat v počítači od přístupu uživatele a aplikací k datům, které má vést k tomu, že změny v interním uspořádání dat nebudou mít vliv na externí reprezentaci a naopak (Codd, 1970).

Relaci zde definuje v souladu s matematickým významem následovně:

> Jsou dány množiny $S_1, S_2, \ldots, S_n$ (ne nutně různé), ***R*** je relací na těchto množinách, pokud je množinou n-tic, z nichž každá má první prvek z S_1, druhý z S_2, atd.
>
> S_j se se označuje jako j-tá doména z ***R***. V souladu s tím řekneme, že ***R*** je stupně n. (Codd, 1970)

[6] http://dl.acm.org/citation.cfm?doid=362384.362685

Pro snazší pochopení lze relaci chápat jako pole (dvourozměrné pole je tabulka), které musí mít následující vlastnosti, aby vyhovělo popisovanému relačnímu pohledu:

1. Každý řádek představuje *n-tici* z ***R***
2. Na pořadí řádků nezáleží
3. Každý řádek je odlišný
4. Pořadí sloupců má význam, neboť odpovídá pořadí domén S_1, S_2, ... , S_n, na kterých je ***R*** definována
5. Význam každého sloupce je částečně vyjádřen jeho označením názvem příslušné domény

Článek se zabývá vazbami/vztahy mezi relacemi, upozorňuje na oddělení pojmů relace a vztah[7], dále klíči (zavádí primární, cizí), normální formou a normalizací. Také shrnuje předpoklady, ze kterých by měl vycházet vhodný dotazovací jazyk pro práci s databází, založený na predikátovém počtu.

Operace s relacemi vycházejí z matematických operací s množinami – permutace, projekce, spojení, kompozice, restrikce. Výsledkem operací s relacemi jsou opět relace. Článek poskytuje matematickou podporu pro práci s daty a podporu pro normalizaci databáze - omezení redundance při jejím návrhu. Také je matematicky definována konzistence.

E. F. Codd i nadále přispíval k teorii relačních databází, mimo jiné prací na normálních formách, relační algebře nebo uvedením tzv. 12 Coddových pravidel[8], která by měl splňovat plně relační databázový systém.

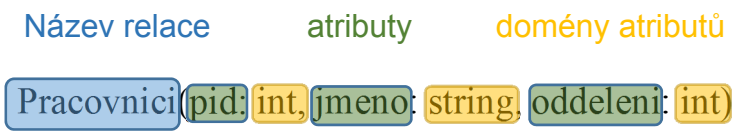
2.1.2 Vlastnosti a pravidla relačních databází

Relační databáze je tedy databáze využívající relační model dat ve smyslu Coddova článku. Implementací relace je v těchto databázích tabulka, která musí splňovat určitá,

[7] V anglicky psané literatuře se používá odlišné *relation* a *relationship*
[8] http://computing.derby.ac.uk/c/codds-twelve-rules/

dříve popsaná pravidla. Řádky tabulky jsou jednotlivé záznamy a sloupce představují atributy jednotlivých záznamů.

Obrázek 2: Relační schéma



Relační schéma je tak totální funkcí z názvů atributů do typů a schéma relační databáze je množina relačních schémat, pro každou relaci jedno (CS145 Introduction to Databases, 2016).

Klíče

Jako klíče v souladu s původním textem se označují vybrané sloupce následujícím způsobem:

Kandidátní klíč	Sloupce, pomocí kterých lze jednotlivé záznamy identifikovat
Primární klíč	Určujeme výběrem jednoho z kandidátních klíčů
Náhradní klíč	V případě, že to není vhodné (složený primární klíč s více sloupci o nevhodné datovém typu), přiřadí se relaci další sloupec (nejlépe číselný), který úlohu primárního klíče převezme
Alternativní klíč	Zbývající kandidátní klíče nevyužité jako primární
Sekundární klíč	Označení pro alternativní klíč (může být více sekundárních klíčů na rozdíl od jediného primárního)
Složený klíč	Někdy k jednoznačnému určení nestačí jediný sloupec - použije se kombinace sloupců
Cizí klíč	K vytvoření vazby mezi tabulkami se používá cizí klíč – sloupec, který obsahuje pouze hodnoty souvisejícího sloupce v odkazované tabulce

Pokud by došlo k tomu, že se záznam z odkazované tabulky vymaže a záznam v odkazující tabulce ukazuje „nikam“, záznam se označuje jako sirotek (Lacko, 2013) a hovoří se o ztrátě referenční integrity. Je to jeden z případů, kdy databáze přestává být konzistentní – záznamy jsou neslučitelné, nejsou ve vzájemném souladu. Udržení konzistence či integrity napomáhají kromě hlídání referenční integrity i další integritní omezení – entitní, které se udržuje pomocí primárního klíče a doménové, které je dáno datovým typem sloupce a někdy navíc i definicí možných hodnot, které lze uložit.

Zachovávání konzistentního stavu je zásadní téma právě u propojování různých systémů v souvislosti se synchronizací dat.

Normalizace

Normalizací se označuje sled kroků nutných pro rozdělení tabulky na sadu tabulek tak, aby každá z nich obsahovala pouze jediné téma (Kroenke & Auer, 2015). Hlavním účelem je odstranění redundancí. Problémem vícenásobně ukládaných dat není velikost prostoru, který zbytečně zabírají, ale zejména následující dva:

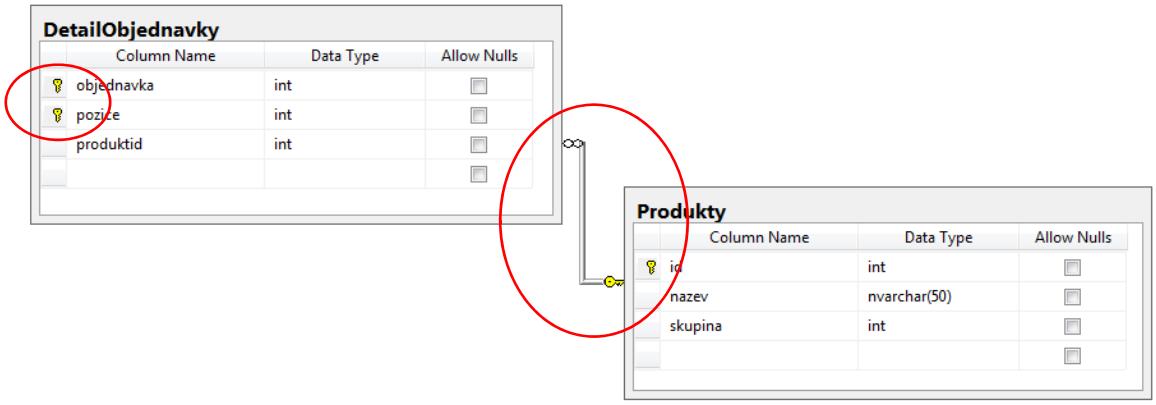
Aktualizační anomálie	Problém vícekrát uloženého údaje při editaci či výmazu Lze najít vždy všechny relevantní záznamy? Je žádoucí mazat celý záznam, i když obsahuje i data o jiných objektech?
Problémy s oprávněním	U citlivých informací uložených na různých místech může být složitější uhlídat oprávnění a mohou být odhaleny nežádoucím uživatelům – k jednomu zdroji přístup není, k jinému ano

Základní postup při normalizaci vychází z vyhledání funkční závislosti mezi daty uvnitř tabulky. Pokud údaje v jednom sloupci přímo vyplývají nebo mohou být určeny z údajů jiného sloupce, jsou funkčně závislé. Dále se vychází z určení determinantu funkční závislosti – sloupec, který obsahuje hodnoty určující hodnotu funkčně závislého sloupce. Determinant by měl být kandidátním klíčem – pokud tomu tak není, je nutné tabulku rozdělit i vícekrát tak, aby tato podmínka byla splněna (Kroenke & Auer, 2015).

K popisu toho, jak důsledně je tabulka normalizovaná se používají jednotlivé úrovně - normální formy. Tyto také představují typy anomálií, kterými se tabulky v určitých normálních formách vyznačují. Pro normální formy platí, že každá vyšší úroveň normální formy - následující normální forma vyžaduje splnění té předcházející (Kroenke & Auer, 2015).

Na následujícím obrázku je vytvořena normalizovaná struktura s použitím složeného primárního a cizího klíče.

Obrázek 3: Normalizovaná struktura a klíče



Transakce

Kromě správného návrhu databáze bez redundancí napomáhá udržet konzistentní stav i transakční zpracování. Transakcí se rozumí logická jednotka práce - sada instrukcí, která se buďto provede celá, anebo v případě nemožnosti data změnit se neprovede vůbec. Představuje tedy dávku příkazů, před jejímž provedením se prověří možnost data změnit požadovaným způsobem - data se musí zajistit před editačním přístupem jiného procesu, zálohuje se původní stav, a která v případě úspěšného ověření data následně změní. V případě neúspěchu se obnoví původní stav. Průběh transakcí se protokoluje a transakční protokol je součástí zálohování.

Transakce musejí splňovat podmínku ACID, která také jednoduše popisuje průběh transakčního zpracování (CS145 Introduction to Databases, 2016):

Atomic	Stav databáze ukazuje buďto všechny účinky transakce nebo žádný z nich
Consistent	Transakce mění databázi z jednoho integritního stavu do druhého integritního stavu
Isolated	Transakce se provádějí nezávisle na ostatních – při zpracovávání by změny v databázi neměly být pozorovatelné
Durable	Po skončení transakce musejí být změny trvalé, zapsané na disk

Hodnota NULL

Hodnota NULL vyjadřuje skutečnost, že údaj není uložen, hodnota není známa – zda jde o záměr či omyl není podstatné. NULL zde neznamená ani nulu, ani prázdný řetězec – je nutné k ní takto speciálně přistupovat a databázové systémy musí umožňovat s touto okolností pracovat. Při vytváření tabulek se také udává, jestli sloupec smí obsahovat hodnotu NULL či nikoli – například primární klíč nesmí být NULL.

Indexy

Index má v databázi stejný účel jako rejstřík na konci knihy – představuje přehled o uložení dat a usnadňuje jejich vyhledávání – v případě rozsáhlých tabulek a hlavně rekurzivních (hierarchických) dotazů je dokonce extrémně zrychluje. Jedná se o dodatečnou, z pohledu popisu dat redundantní strukturu (objekt), který je ukládán nezávisle na samotných datech. To vede k obavě, že definování/použití indexů může zpomalovat aktualizační dotazy (vkládání a změny tabulek). Při aktualizaci se zpravidla mění minimum dat v porovnání se složitějším vyhledávání a tak je přínos indexů vyšší, navíc ani jak vyplývá z praktické části, se tato obava nedá potvrdit. I použití indexů má ale svá pravidla – indexují se (jako indexy/pro indexy se vybírají) ty sloupce, podle

kterých se bude často vyhledávat, které uchovávají informace o vazbách mezi daty (tvoří hierarchickou strukturu).

2.1.3 Jazyk SQL

Původním tvůrcem relačního databázového systému byla firma IBM se svým projektem System R[9], který měl splňovat Coddovy principy. Při jeho vývoji byl dvojicí autorů Donaldem D. Chamberlinem a Raymondem R. Boycem vytvořen i jazyk SQL[10], původně nazvaný SEQUEL[11]. Představuje standardní jazyk pro práci s relačními databázemi. Jedná se o deklarativní, neprocedurální jazyk – neurčuje, jak se má s daty pracovat (jak mají být získána), ale jaká data mají být získána. Jazyk je orientovaný na práci s množinami. Splňuje nároky úplného relačního dotazovacího jazyka – podporuje všechny operace s relačním modelem. Příkazy mohou být vnořovány s ohledem na charakter operací s relacemi zmíněných dříve.

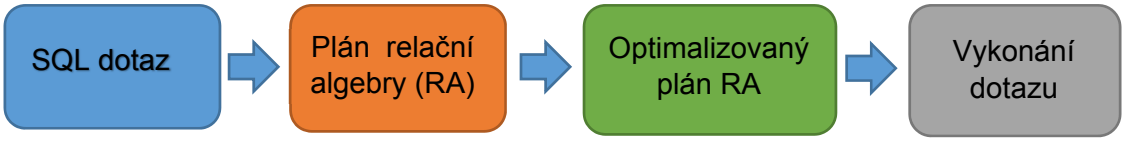
Základní příkazy jazyka SQL se rozdělují do několika skupin:

DDL – Data Definition Language	Vytváření a změny struktur databáze. Příkazy jako CREATE DATABASE, ALTER TABLE
DCL – Data Control Language	Příkazy pro řízení oprávnění databázových objektů. Například GRANT, REVOKE, DENY
DML – Data Manipulation Language	Aktualizace a vyhledávání dat. Příkazy jako INSERT, UPDATE
TCL – Transactional Control Language	Řízení transakcí v databázi - COMMIT, ROLLBACK, ...
DQL – Data Query Language	Samotný příkaz SELECT – někdy považováno za součást DML

[9] http://dl.acm.org/citation.cfm?doid=358769.358784
[10] https://docs.oracle.com/cd/B12037_01/server.101/b10759/intro001.htm
[11] http://dl.acm.org/citation.cfm?id=811515

Provádění SQL dotazů se uskutečňuje převodem do výrazů relační algebry, procedurálního jazyka pro provádění databázových operací ve smyslu algebraických výrazů, následně optimalizací, tedy vyhledáním logicky ekvivalentního, ale efektivnějšího výrazu relační algebry, a spuštěním (CS145 Introduction to Databases, 2016).

Obrázek 4: Zpracování SQL dotazu databázovým strojem



Uvádím zde pouze základní principy jazyka. Relační algebra a tím pádem i základní SQL příkazy mají své limity spočívající například v rekurzivních a hierarchických dotazech[12], ale vzhledem k tomu, že i jazyk SQL prochází vývojem a kromě ANSI standardu SQL poskytují všichni tvůrci relačních databázových systémů své vlastí varianty SQL jazyka, které sice snižují přenositelnost SQL kódu, ale zvyšují využitelnost vlastního systému, existují i řešení takových úkolů. Proto také další podrobnosti a možnosti uvedu u konkrétní implementace SQL v části věnované MS SQL Serveru.

2.1.4 OLTP a OLAP

OLTP (OnLine Transactional Processing) je způsob zpracování produktivních dat, kterému odpovídá i způsob jejich ukládání – v normalizované podobě, podporující vysoce výkonné, krátké aktualizační dotazy. OLTP systém, produktivní systém, provozní systém či normalizovaná databáze jsou pojmy s totožným významem (Lacko, 2013).

Značně odlišným přístupem k datům je OLAP (OnLine Analytical Processing). Jde o způsob zpracování dat zaměřený na analýzy a dolování dat[13]. Předpokladem úspěšného a zároveň dostatečně rychlého zpracování takových úloh je jiný, zpravidla multidimenzionální datový model, který není normalizovaný, je určený k manipulaci velkých objemů dat, která vylučuje například operace spojování tabulek a různých zdrojů

[12] Pomocí relační algebry nelze nalézt tranzitivní uzávěr
[13] Data mining – získávání informací z dat spočívající v použití analytických matematických metod

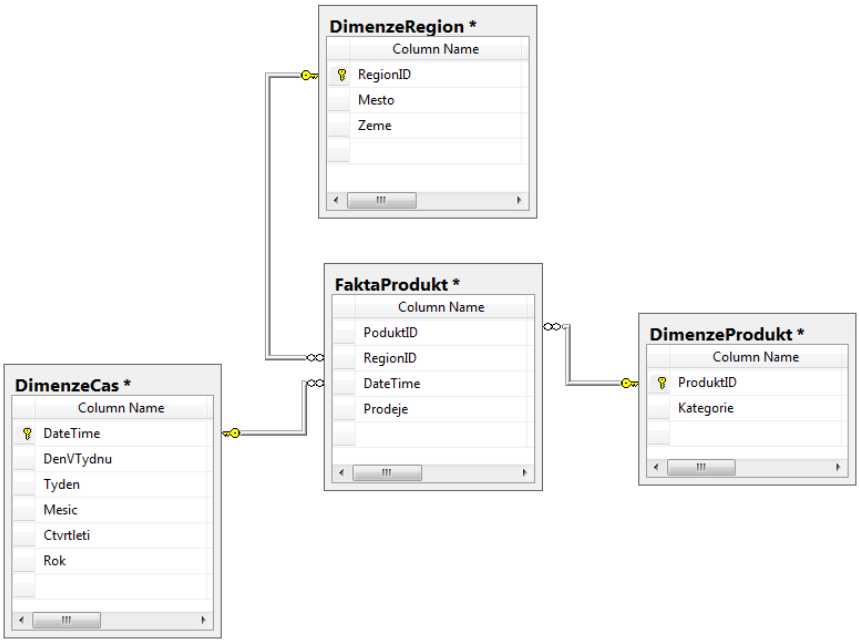
(serverů, systémů, externích úložišť) typické pro OLTP databáze. Pro OLAP je naproti tomu typické centralizované úložiště, které obsahuje historické údaje, je průběžně doplňované z produktivních systémů a je zaměřené pouze na čtecí operace. Takový typ úložiště jednoznačně zbavuje produktivní systémy zátěže v případě spouštění rozsáhlých reportů (Lacko, 2013). OLAP databáze, analytická databáze či datový sklad jsou také odpovídající pojmy a i OLAP technologie má svých 12 pravidel[14] od E. F. Codda.

OLAP strukturu tvoří dva druhy údajů (Lacko, 2013):

Fakta	Také numerické měrné jednotky obchodování, zpravidla největší tabulka v databázi. Zjištěné údaje zkombinované s jinými fakty, někdy přirozeně aditivní (možno sčítat v čase - objemy prodejů), jindy nutné vyjadřovat pomocí časové dimenze (stavy na skladě v jednotlivých měsících)
Dimenze	Obvykle hierarchická stromová struktura obsahující hodnoty ke členění dat v tabulkách faktů (region, produkt, ale i zákazník nebo čas)

Tabulky s fakty a dimenzemi potom tvoří topologické uspořádání, schéma. Nejčastěji používanými jsou schéma vločky a hvězdicové schéma.

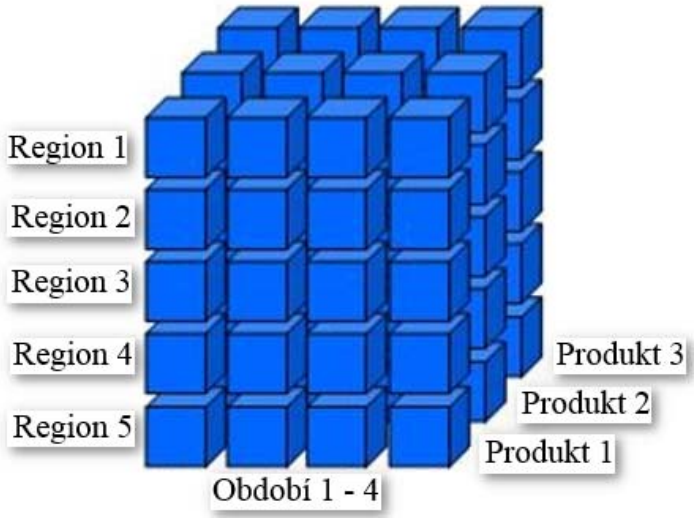
Obrázek 5: Hvězdicové schéma



[14] http://www.minet.uni-jena.de/dbis/lehre/ss2005/sem_dwh/lit/Cod93.pdf

Koncepci multidimenzionální OLAP lze vysvětlit pomocí kostky geometrické, kdy jednotlivé osy představují dimenze (tabulky dimenzí) s tím rozdílem, že jednotlivých tabulek dimenzí může být mnohem více než tři.

Obrázek 6: OLAP kostka



OLAP kostka na obrázku č. 6 odpovídá hvězdicovému schématu na obrázku č. 5, tabulka faktů potom obsahuje údaje například o prodejích konkrétních produktů v určitém regionu a období. Tyto údaje tak leží na průsečících řezů (rovin) kostkou vedených potřebným okamžikem, regionem a pro určitý produkt.

Jednotlivé typy OLAP úložišť lze rozdělit následovně (Lacko, 2013):

MOLAP	Multidimenzionální OLAP. Data získáná z datového skladu, ale i z operačních zdrojů slouží jako zdroj pro napočítání předběžných výsledků a uložení do vlastních struktur a sumářů.
ROLAP	Relační OLAP. Data jsou uložena v relačním datovém skladu. OLAP server generující SQL příkazy předkládá data jako multidimenzionální pohled
HOLAP	Hybridní OLAP. Údaje zůstávají v relačních databázích a napočítané agregace se ukládají do multidimenzionálních struktur.

2.1.5 Datový sklad a datové tržiště

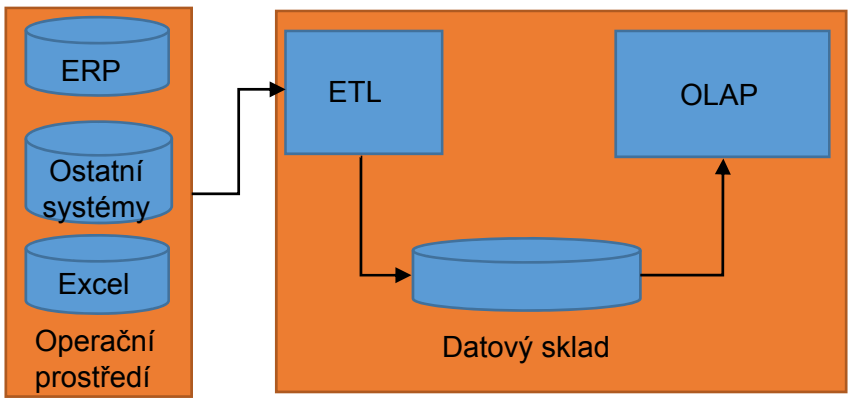
Datovým skladem se na rozdíl od klasického skladu, který slouží k dočasnému uchovávání zásob či výrobků, rozumí spíše archiv či depozitář dat z provozní činnosti podniku tříděný podle času, druhů a podobně (Lacko, 2013). Datové tržiště nebo datový trh je spíše menší datový sklad, podmnožina datového skladu na nižší hierarchické úrovni firmy. Obsahuje data jednotlivých podnikových úseků či procesů (Lacko, 2013).

2.1.6 ETL proces

Proces ETL slouží k naplňování datového skladu původně produktivními daty podniku. Musí být plánovatelný a podporovat různé zdroje. Kromě prvotního vytvoření datového skladu je nutné jej i aktualizovat. Operační data také nemusejí být pouze v jednom systému. Proces sestává ze tří etap (Lacko, 2013):

Extraction	Výběr dat z různých systémů, dokonce i mezipodnikových či internetových včetně monitorování dostupnosti zdrojů informací
Transformation	Ošetření nesprávných a chybějících údajů, sjednocení datových typů z různých systémů, ověřování referenční integrity při aktualizaci datového skladu v případě proměnlivé struktury zdroje, apod.
Loading	Přemístění a import dat do cílového systému

Obrázek 7: ETL proces



2.1.7 Business Intelligence (BI)

Termín Business Intelligence byl zaveden Howardem Dresnerem v roce 1989. Existuje více definic BI, ale pro jednoduchost a potřeby práce pod tímto názvem rozumím souhrn nástrojů, které mají konvertovat velké objemy dat na poznatky, které může využít management při rozhodování (Lacko, 2013).

BI tak zahrnuje jak OLAP úložiště a nástroje pro integraci, tak reportovací či analytické služby a data mining. Potřebné programové vybavení poskytují tvůrci databázových a podnikových systémů. Běžná je i možnost připojení kancelářských programů k OLAP úložišti a jeho využití managementem s minimální podporou IT oddělení.

2.1.8 Připojování k databázovým systémům

S ohledem na zaměření práce se nebudu zabývat distribuovanými, replikovanými systémy, které slouží ke zvýšení výkonu – škálovatelnosti databázových systémů ať již rozdělením činností mezi jednotlivé servery (členění je možné také podle zaměření jednotlivých serverů na určité úkoly/operace), které sdílejí data nebo ke zlepšení dostupnosti – rozdělení podle lokalit a jejich způsobu propojování. Cílem práce také není řešit způsob komunikace mezi servery na síťové úrovni a popis používaných protokolů a zabezpečení komunikace[15]. Ze stejného důvodu nebudu popisovat ani nativní klientské aplikace a vlastní API jednotlivých producentů databázových systémů. Zaměřím se na možné způsoby připojování k relačním databázovým systémům pro externí aplikaci a zde lze využít dvě možnosti:

Připojení k systému prostřednictvím ovladače (např. ODBC)	Běžné pro propojování databázových systémů a klientských aplikací, které umožňuje spouštět SQL příkazy na vzdáleném stroji
Export dat jednoho systému do souborů a jejich přenos s následným importem do druhého systému	Varianta pro nestandardní způsoby připojení a při chybějícím oprávnění pro přímé přípojení

[15] https://msdn.microsoft.com/en-us/library/ee210043(v=sql.105).aspx

Funkci ovladače ODBC (Open Database Connectivity) lze přirovnat k funkci ovladačů různých periferních zařízení počítače – uživatel neřeší specifické vlastnosti a požadavky systémů a využívá služeb standardizovaného ODBC rozhraní. Ovladač ODBC tak představuje další vrstvu mezi klientskou aplikací a databázovým systémem, v případě propojení systémů potom analogicky prostředníka pro vzájemnou komunikaci. To umožňuje vkládat příkazy databázového jazyka (zpravidla jako řetězce) do zdrojového textu použitého programovacího jazyka – převod zajistí ovladač ODBC, který existuje vždy pro konkrétní systém.

ODBC má také variantu pro svět Java – JDBC. Firma Microsoft, původní tvůrce konceptu ODBC, přišla v polovině 90. let s rozhraním OLE DB založené na rozhraní COM - smysl je ale obdobný jako u nízko úrovňového ODBC s rozšířením pro nerelační zdroje dat.

Druhá varianta propojování přichází v úvahu zejména při jednorázových, ne tak častých akcích anebo při problémy se získáváním oprávnění – při připojování databázového systému je nutné získat oprávnění, které nemusí odpovídat tomu, které má uživatel v aplikaci, k jejímž datům se potřebuje dostat. Řešení založené na ovladači ODBC je rychlé a spolehlivé – fungují na něm či na obdobném konceptu webové aplikace. V tomto ohledu je export dat a jejich přenos po síti s následným importem problematický. Další problém je potom struktura souboru – zpravidla textového. Jaký zvolit oddělovač sloupců, který se nebude vyskytovat ať již omylem či záměrně v nějakém textovém sloupci a nezpůsobí tak nesprávně rozdělení sloupců a chybu na konci řádky/záznamu (sloupců bude najednou na několika řádkách více, než se očekává)? V tomto případě se jako vhodnější varianta jeví použití souborů XML, které mají jasnou strukturu s daty sloupců vloženými jako elementy či atributy elementů.

2.2 ERP systém

Pojmem ERP systém (z anglického Enterprise Resource Planning) rozumíme podnikový informační systém, který slouží k řízení a podpoře podnikových procesů a ke správě dat vznikajících při těchto procesech. Systém, který nemusí být dokonce ani v současnosti nutně počítačový, by měl tedy umožňovat správu procesů a uchovávání dat personalistiky, logistiky, controllingu, výroby ale i dalších technických oddělení. Patří sem tedy účetnictví (finanční i manažerské - controlling), pořizování materiálu, řízení skladů, plánování výroby, tedy zdrojů pro výrobu, správa zaměstnanců, prodej výrobků, ale i možnost správy technické dokumentace a řízení jejích změn, správa majetku a jeho dokumentace.

Zpravidla se jedná o síťovou aplikaci typu klient – server. Uživatel, kterým je pracovník, pak podle svého oprávnění může zpracovávat právě taková data a procesy, které odpovídají jeho zařazení v rámci podniku či oddělení. Lze tak zajistit, aby citlivá data viděli jen někteří uživatelé. ERP systémy obsahují také *workflow*, kterým je možné posílat zprávy, úkoly a jehož prostřednictvím se také zajišťuje schvalovací (faktury, objednávky – schvalovací hierarchie musí být definována) nebo informativní oběh dokumentů a jiných objektů spravovaných v systému.

2.2.1 SAP jako ERP systém

Systém SAP[16] je produktem softwarové firmy SAP SE, kterou v roce 1972 založilo několik bývalých zaměstnanců firmy IBM. Jednotlivé verze systému byly označovány od první z roku 1973 jako R/1 až po poslední R/3[17] představenou v roce 1992. Verze R/3 měnila označení postupně již bez zásadních technických a uživatelských proměn přes R/3 Enterprise a mySAP ERP až po SAP ECC (ERP Central Component). Od verze R/2 představené v roce 1979 lze hovořit o SAPu jako o ERP systému, který ale ještě využíval sálové počítače. Zásadní změnou ale byla verze R/3, která se zavedením 3 - vrstvé klient-

[16] Z původního německého Systeme, Anwendungen und Produkte
[17] R znamená Real Time - Datenverarbeitung

server architektury stala kompatibilní s běžnými operačními systémy a umožnila značné rozšíření systému SAP (Maassen, Schoenen, Frick, & Gadatsch, 2007).

2.2.2 Architektura systému SAP ERP

Systém SAP je založený na třívrstvé architektuře, kdy je oddělena databáze (datová vrstva), aplikace (logika, aplikační vrstva – systém SAP) a uživatelské rozhraní (prezentační vrstva, SAP GUI či web). Systém SAP může využívat různé typy databázových systémů a jeho GUI je od verze 4.6D, která je napsaná v Javě, nezávislé na operačním systému (Maassen, Schoenen, Frick, & Gadatsch, 2007).

Jádro systému SAP je naprogramované v jazyce C, ale pro programování aplikací a manipulaci s daty v prostředí SAP se používá vlastní jazyk ABAP – Advanced Business Application Programming. Původně procesor pro formátování zpráv[18] v současné době slouží k programování SAP aplikací. Aplikační vrstvu běžící na samostatném serveru tedy představuje samotný systém SAP s ABAP aplikacemi (Kühnhauser, 2009).

Pro přístup k databázové vrstvě se využívá příkazů ABAP označovaných jako OpenSQL – přímé používání SQL příkazů nepatří mezi doporučené řešení. Je možné napsat, že příkazy OpenSQL se prostřednictvím běhového prostředí ABAPu odpovídajícím způsobem převádějí do SQL jazyka právě použitého databázového systému, kterým se ABAP programátoři nemusejí zabývat (Kühnhauser, 2009).

Aplikační programy běží v pracovních procesech, tzv. *workprocesech*, které jsou systémovými databázovými uživateli. Tyto jsou prostředníkem či kanálem mezi uživateli systému a databázovým systémem. Přidělování volných procesů jednotlivým uživatelům, kterých může být přihlášeno více, než je k dispozici *workprocesů*, má na starosti zvláštní podsystém, dispečer (Kühnhauser, 2009).

Na aplikační úrovni existuje i Java varianta s vlastním SAP NetWeaver System[19].

[18] Původní německý význam zkratky ABAP byl Allgemeiner Berichts-Aufbereitungs-Prozessor
[19] https://help.sap.com/saphelp_nw70/helpdata/en/c8/cdfacc37efa84d914699ad31eb69b8/content.htm

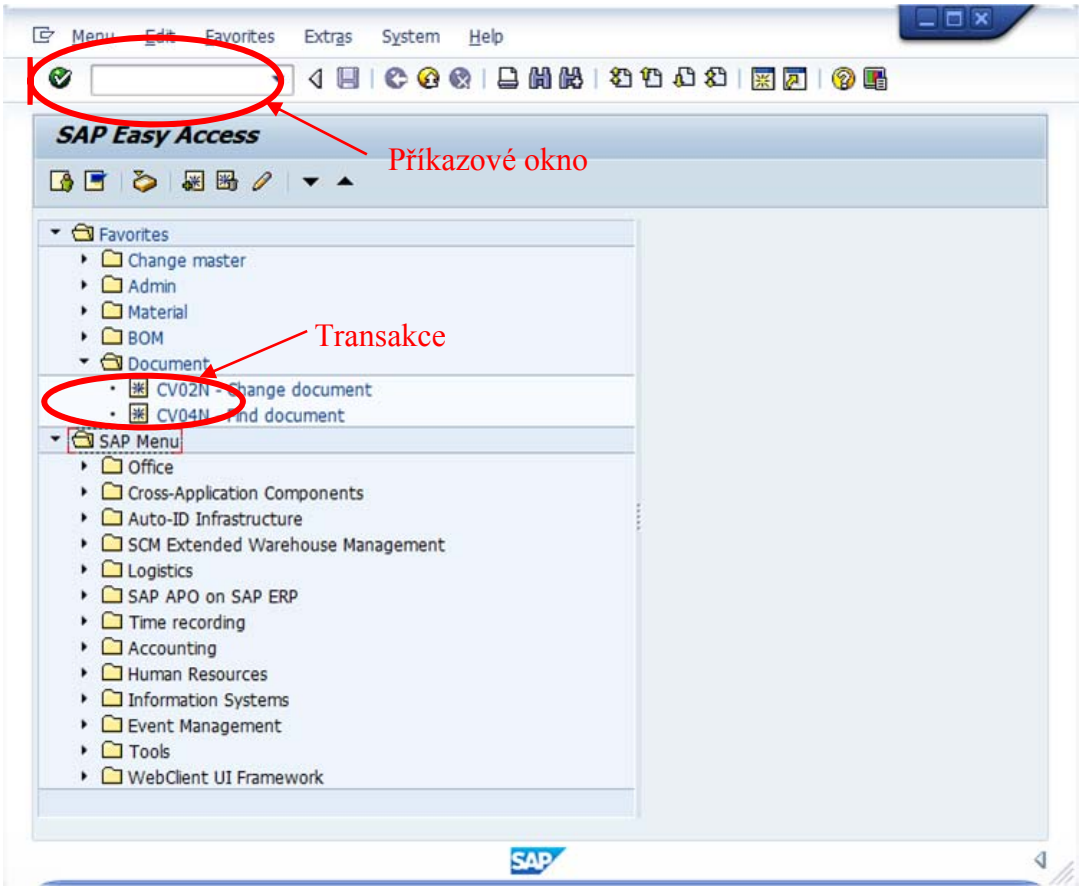
2.2.3 SAP z pohledu uživatele

Uživatel se připojuje k systému SAP prostřednictvím prezentačního rozhraní SAP GUI, ke kterému se vztahují i následující popisy - specifika webového přístupu s ohledem na využití v této práci nebudu popisovat.

Pojmem objekt rozumím podnikový objekt ve smyslu Business Framework, který je popsaný dále, tedy například materiál, zaměstnanec či faktura, a který se zakládá a aktualizuje prostřednictvím formulářů tak, jako je tomu i u jiných aplikací.

Pro správu objektů se používají transakce. Transakce v prostředí SAP označuje řetězec dialogových kroků, které spolu z hlediska podnikové ekonomiky vzájemně souvisejí (Maassen, Schoenen, Frick, & Gadatsch, 2007). Při spuštění transakce se ověří oprávnění uživatele, poté se načtou data formulářů a související programová logika, která podle typu objektu zajistí pohyb v jednotlivých částech formulářů, editaci v jednom či více krocích a uložení. Transakce odpovídá dílčímu procesu aplikace.

Obrázek 8: Prostředí SAP



Transakci lze spustit přímo zadáním kódu transakce, například MM01 pro založení materiálu, do příkazového okna nebo si lze potřebné transakce uložit opět prostřednictvím kódu mezi oblíbené a vytvořit si tak vlastní menu. Transakce je také možné vyhledávat nebo najít postupným „rozklikáváním“ standardního menu, které je ale díky obsáhlosti systému značně složité. Důležité je, že prostřednictvím formulářů je možné zjistit přímo datový zdroj - databázovou tabulku a pole, kde jsou příslušná data uložena.

Struktura je dynamická tabulka, která se sestavuje a naplňuje údaji z více databázových tabulek. Při spuštění transakce se s ohledem na vstupní údaje od uživatele a podle uložené definice struktury načtou data z daných databázových tabulek.

U některých transakcí nebo z formulářů některých objektů je možné zjišťovat přímo databázové tabulky – jde zejména o základní objekty, které se neskládají z dalších objektů nižší úrovně (kmenová data materiálu, údaje o uživateli, apod.), pokud se ale jedná o složitější objekty (kusovník, objednávky, apod.), jde zpravidla o strukturu - zjistí se pouze její název a název pole, konkrétní tabulky je nutné dohledat buďto pomocí správné transakce pro prohlížení struktur (např. SE12, SE15 či SE84) nebo v dokumentaci – pokud chybí oprávnění. Dokumentace je důležitá zejména v případě vlastních podnikových SAP aplikací, kde se nelze spolehnout na webové zdroje. Toto jsou pro účely práce zásadní skutečnosti, které budou využity v praktické části.

2.2.4 Možnosti připojování k systému SAP

Díky oddělené databázové vrstvě je sice technicky možné se připojit k datům systému SAP připojením přímo k databázi jako nalinkovanému serveru, ale není to podporované řešení. Hlavně by asi nastal problém s přidělením oprávnění přímo k databázovému serveru. Uživatelský přístup pracuje s aplikační vrstvou systému SAP a databázová vrstva je věcí *dispečerů* a *workprocesů* jak bylo uvedeno dříve.

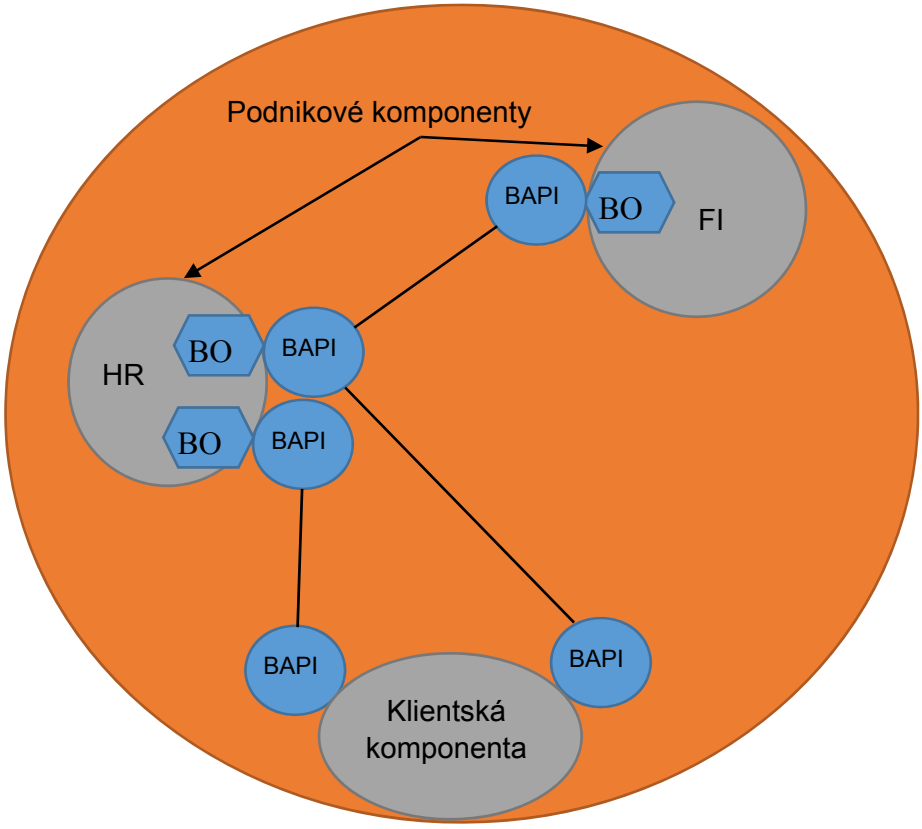
Základní způsob připojení k SAPu je prostřednictvím RFC (Remote Function Call). Toto připojení odpovídá běžnému databázovému RPC (Remote Procedure Call) a zahrnuje vlastní rozhraní a protokoly SAP. Takové připojení vyžaduje speciální uživatelský přístup, který umožňuje externím programům napsaných v různých programovacích

jazycích připojit se k systému SAP. Může ale sloužit i k vzájemnému propojení jednotlivých SAP systémů.

Dostatečný popis připojení včetně způsobu komunikace propojených systému je k dispozici na SAP Help portálu[20]. Zde by takové podrobnosti neměly využití. Důležité je, že toto je standardní způsob pro připojení k systému SAP a ostatní řešení jej využívají jako rozhraní.

První možností, co systém SAP nabízí i u podniků, které nemají v SAPu implementované všechny své procesy, je flexibilní infrastruktura označovaná jako Business Framework - řešení pro propojování systémů, které vychází z principu COM. Architektura tohoto řešení sestává z podnikových komponent, kterými se rozumí ucelené souhrny funkcí, které se vztahují k určité podnikové oblasti (logistika, personalistika) s jejich procesy a jednotkami, které se označují jako podnikové objekty (BO – Business Objects) a lze si pod nimi představit například zakázky či faktury s jejich údaji/daty a s metodami jako je založení zakázky či faktury. Pro přístup k BO se využívá metod zvaných BAPI – Business Application Programming Interface, vlastní API v prostředí SAP.

Obrázek 9: Schéma architektury Business Framework



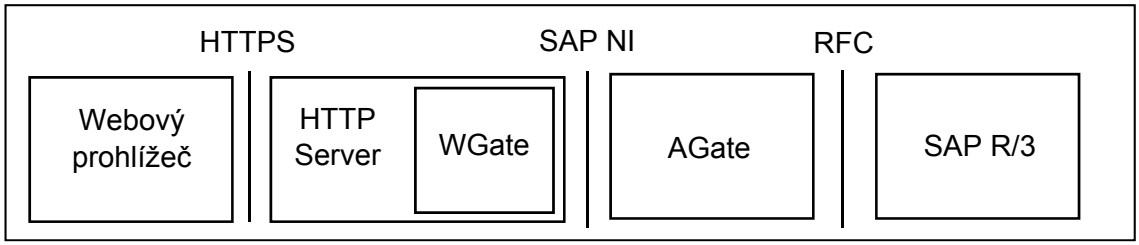
Zdroj: (Maassen, Schoenen, Frick, & Gadatsch, 2007)

[20] https://help.sap.com/saphelp_nw70/helpdata/en/6f/1bd5b6a85b11d6b28500508b5d5211/frameset.htm

Další možností je přístup s využitím ITS (Internet Transaction Server), kdy prezentační vrstvu systému SAP představuje webový prohlížeč. Řešení je vhodné pro případy, kdy data přicházející do podniku zvenčí mají spustit také nějakou aktivitu v systému. Možné je využití jak v rámci intranetu, tak i internetu – v tomto případě s dalším členěním na aplikace typu B2B a C2B (Maassen, Schoenen, Frick, & Gadatsch, 2007).

Server ITS je prostředníkem mezi webovým prohlížečem a aplikacemi IAC (Internet Application Components), které jsou uživateli přístupné podle typu oprávnění, a mezi které patří například zobrazení informací o stavu zákaznického účtu, katalog výrobků a obchod online či zadání nabídky pracovních míst nebo zadání zákaznické zakázky, dokumentaci projektu a další (Maassen, Schoenen, Frick, & Gadatsch, 2007).

Obrázek 10: Internet Transaction Server



Zdroj: (Maassen, Schoenen, Frick, & Gadatsch, 2007)

Samotný ITS se člení na softwarově oddělené části WGate a AGate, což podporuje i jejich hardwarové oddělení za účelem zvýšení bezpečnosti. WGate převádí HTTP požadavky na protokol SAP NI (Network Interface), ze kterého je AGate následně převede na RFC, odkud je následně spuštěna volaná IAC (Maassen, Schoenen, Frick, & Gadatsch, 2007).

Různé způsoby integrace systému SAP a kancelářských aplikací jsou také možné, jde ale o přístup koncového uživatele, kterým se zde nezabývám.

2.2.5 SAP BI

Systém SAP má také vlastní řešení BI, které plně odpovídá BI popsanému dříve a příliš se neliší od nástrojů BI firmy Microsoft.

Součásti SAP BI se ukrývají pod různými zkratkami. Z původního BIW (Business Information Warehouse) se název zkrátil na BW, poté na BI a zpět. Někdy do toho vstupují i řešení firmy BusinessObjects (BO, BOBJ), dnes součásti firmy SAP. Vše ale náleží do komplexní platformy SAP NetWeaver[21] jako řešení BI a lze je nalézt různě zařazené na SAP Help portálu[22]. Pro potřeby práce je podstatný datový sklad systému SAP - SAP BW a nástroj pro ETL proces - BODS (BusinessObjects Data Services).

Datový sklad nemusí nutně sloužit pouze pro procesy BI, je možné ho využít jako zdroj dat pro externí aplikace. Odpadne tak jejich neustálé kopírování z produktivního systému. V případě, kdy je datový sklad pravidelně doplňován a není požadavek na úplně aktuální data, může být takové řešení vhodné zejména tehdy, když není možné získat přístup přímo do produktivního SAP systému. Některé objekty v SAPu mohou mít platnost v budoucnu - změny na nich neplatí okamžitě, potom určitá neaktuálnost nevadí vůbec a takový zdroj dat může být i výhodný, protože jde o vyčištěná data uložená způsobem, který je pro rychlé načítání bez potřeby jejich editace vhodný[23].

Možností, jak přistupovat k datům systému SAP, je tedy více. Především ale z důvodů oprávnění a složitého procesu ověřování aplikací se práce zaměří na využití datového skladu a přenos dat s využitím souborů CSV a ETL nástroje BODS, který bude také využit pro přístup k datům produktivního systému.

[21] https://help.sap.com/netweaver
[22] https://help.sap.com/nwbw
https://help.sap.com/bobi
http://help.sap.com/bods
[23] OLTP vs. OLAP

2.3 MS SQL Server

MS SQL Server je relační databázový systém společnosti Microsoft, původně vyvíjený společností Sybase, která je v dnešní době součástí firmy SAP. Systém nezahrnuje aplikaci tak, jako je tomu u systému SAP, slouží jako samotná databázová vrstva, je také díky tomuto využití známější než systém SAP. Práce v tomto systému bude rozebrána v praktické části, kde bude také přesnější popis prostředí a funkcionality, proto zde uvedu jen zásadní informace.

MS SQL Server nabízí následující moduly, ke kterým se uživatel připojuje prostřednictvím konzole MS SQL Server Management Studio:

Database engine	Přístup k databázi, není podstatné zda OLTP či OLAP. Zde lze pracovat s databázemi, psát skripty či provádět administrátorské úkony.
Integration services (SSIS)	Provádění integračních balíčků pro propojování systémů a jiných zdrojů, ETL procesu, atd.
Reporting services (SSRS)	Zpracování reportingu (i pro webový server)
Analysis services (SSAS)	Služby MOLAP

I pro MS SQL Server je k dispozici Business Intelligence (BI) a zahrnuje právě moduly SSIS, SSRS a SSAS. Vývojovým nástrojem pro BI je SQL Server Data Tools – doplněk pro MS Visual studio. Zde se mohou i s pomocí průvodců připravovat balíčky integračních služeb, vytvářet reporty (webové stránky reportingu), OLAP struktury a připravovat struktury pro data mining. Výsledkem procesu jsou konfigurační XML soubory, které obsahují jednotlivé kroky a slouží jednotlivým službám SQL Serveru jako popis kroků při provádění.

2.3.1 Jazyk T-SQL

Jazyk Transact-SQL (T-SQL) je vlastní implementací SQL jazyka společností Microsoft a Sybase. Slouží pro psaní dotazů, pohledů, procedur a uživatelských funkcí. Dále obsahuje příkazy pro správu databáze a integrační služby. I některé události tvořené pomocí SQL Server Data Tools jsou ve skutečnosti dotazy tvořené T-SQL.

Jak již bylo zmíněno dříve, zde popisované prvky jazyka T-SQL je možné využívat i ve standardním ANSI SQL. Kódy použité v praktické části je tak s odpovídajícími obměnami možno uplatnit i v systémech Oracle či MySQL.

Následující stručný přehled jazyka T-SQL slouží k vyjádření jeho možností a konkrétní aplikace s podrobnějším popisem uvedu v praktické části.

Databázové objekty SQL Serveru	
Tabulky	Obsah veškeré databáze, v případě složitějších dotazů se využívají i dočasné tabulky
Pohledy	Chovají se jako virtuální tabulky či uložené dotazy
Indexy	Rejstříky pro zrychlené vyhledávání
Spouště	Vykonávají nastavené SQL příkazy při zásahu do tabulky
Procedury	SQL příkazy, které přijímají parametry a vracejí hodnoty, výsledky dotazů
Omezení	Zabraňují vložení nekonzistentních dat
Pravidla	Definují hodnoty, které mohou být vloženy do sloupce

Datové typy	
Přesné číselné	Bigint, bit, decimal, money, tinyint, ...
Přibližné číselné	float, real
Datum a čas	date, datetimeoffset, smalldatetime, ...
Řetězce	char, text, varchar
Unicode řetězce	nchar, nvarchar, ntext
Binární řetězce	binary, varbinary, image
Ostatní	cursor, hierarchyid, Spatial Types (geometrické a geografické), xml, table, timestamp, ...

Syntaktické prvky jazyka T-SQL	
Operátory	Obvyklé aritmetické a logické, přiřazení či porovnání a pro sčítání řetězců
Funkce	Vestavěné funkce pro práci s daty jako např. . Možnost psaní vlastních funkcí – UDF (User Defined Function)
Proměnné	DECLARE @LocalVariable - pro pokročilejší dotazy. Použití jako v ostatních programovacích jazycích
Řízení běhu	BEGIN..END, BREAK, IF..ELSE, RETURN, WHILE, ...

Moderní SQL, kam T-SQL patří, tak umožňuje nejen jednoduché načítání a změny dat, ale i jejich pokročilé zpracování před odesláním do aplikace či naopak. Také literatura doporučuje využívat možností konkrétních implementací jazyka SQL pro maximální využití daného systému (Molinaro, 2009).

2.3.2 Možnosti připojování externích systémů

SQL Server nabízí následující možnosti připojení k externím systémům:

Linked server	Uložené připojení ke vzdálenému serveru prostřednictvím ovladače cílového serveru. Je možné spouštění adHoc dotazů a procedur na vzdáleném systému nebo pravidelný import dat
OPENROWSET	T-SQL příkaz pro obdobný přístup jako v prvním případě, připojení ke vzdálenému serveru není uložené
SSIS	Konfigurace propojení pomocí průvodce a balíčků SSIS s možností využití Data Tools
Bulk insert	Import dat ze souboru napsaný v T-SQL nebo nadefinovaný v balíčku integrační služby a v případě opakování spouštěný jako naplánovaná úloha pomocí nástroje SQL Server Agent. S SQL Serverem je také možné pracovat prostřednictvím příkazového řádku[24] - v takovém případě lze použít program *bcp*[25].

Výše uvedené úlohy SQL Serveru lze plánovat prostřednictvím nástroje SQL Server Agent – bude popsáno v praktické části.

2.3.3 Připojování k databázi a extrakce dat pro webovou aplikaci

Webové aplikace psané v PHP se připojují k SQL Serveru prostřednictvím ovladače SQL Serveru pro PHP[26], který se konfiguruje na straně webového serveru. Na straně SQL Serveru se potom konfiguruje tzv. End Point[27], který slouží pro připojení klientských aplikací.

[24] https://msdn.microsoft.com/en-us/library/ms162773.aspx
[25] https://msdn.microsoft.com/en-us/library/ms162802.aspx
[26] Podrobnosti v kapitole 2.1.8
[27] https://msdn.microsoft.com/en-us/library/ms181591.aspx

Takto zajištěný způsob komunikace potom probíhá následujícím způsobem:

1. SQL Server obdrží SQL dotaz (textový řetězec jako součást PHP kódu)
2. Vykoná dotaz
3. Výsledek vrací přes TDS[28] protokol zpět aplikaci.

Dotazy, které lze spustit v SQL Server Management Studiu, lze odeslat bez úprav i tímto způsobem.

[28] https://msdn.microsoft.com/en-us/library/dd304523.aspx

3 Využití ERP systému jako zdroje dat pro webové aplikace

Účel a důvody existence externích aplikací jsou popsané v úvodu a v teoretické části, proto rovnou uvedu možná řešení v případě systému SAP a jeho datového skladu SAP BW a také možnost připojení k produktivnímu systému prostřednictvím ETL nástroje BODS.

3.1 Připojení k datovému skladu SAP BW

Podmínky, za kterých je vhodné získávat produktivní data prostřednictvím datového skladu, jsou zejména tyto dvě:

V datovém skladu jsou potřebné informace	Nejen data objektů, ale například i údaje o vazbách mezi nimi pro potřeby složitějších, hierarchických dotazů
Datový sklad je dostatečně aktualizovaný	S ohledem na proměnlivost a platnost dat a typ aplikací, které je využívají

Samotný proces probíhá v následujících krocích:

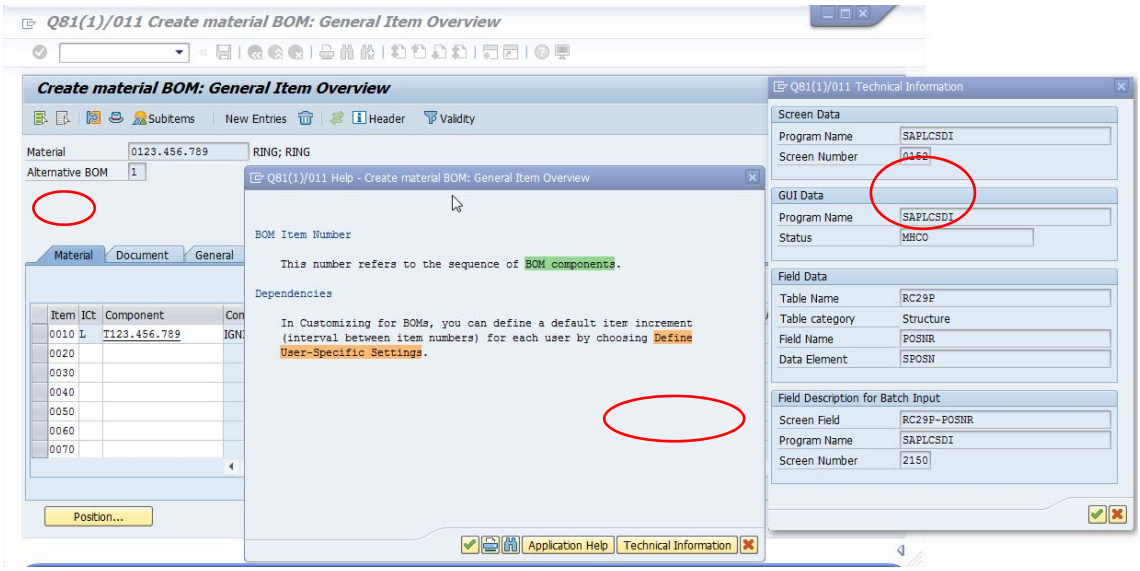
1. Určení hledaného objektu v SAPu

   Pro názornost použiji objekt typu kusovník - rozpis materiálů potřebných k výrobě nějakého dílu nebo i celého výrobku, který také obsahuje informace o množství potřebného materiálu. Jednotlivé položky (pozice) mohou tvořit hierarchickou strukturu typu rodič – potomek. Nejvyšší úroveň je výrobek, nejnižší potom základní materiál, např. ocel.

2. Potřebná políčka a zdrojové tabulky či struktury

Kusovník lze prohlížet například transakcí CS03, odkud se také snadno ověří, že transakce načítá data ze struktur RC29K a RC29P – po kliknutí do zjišťovaného pole se stiskne *F1* a poté *Technical Information*, jak je vidět na obrázku.

Obrázek 11: Zjišťování datového zdroje

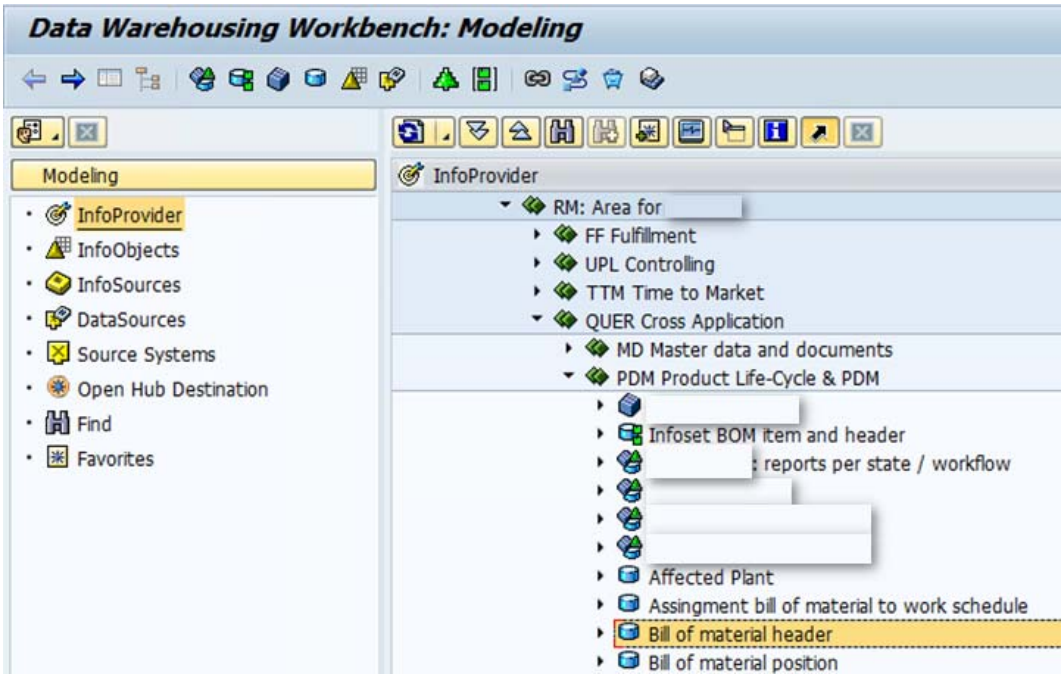


Způsobem blíže popsaným v teoretické části se dále zjistí, ze kterých tabulek struktura sestává z následujících tabulek, a ev. které z nich budou relevantní. V našem případě to byly následující tabulky a jejich význam:

MAST	Přiřazení materiálu ke kusovníku
STKO	Hlavička kusovníku
STPO	Pozice kusovníku
STAS	Informace o platnosti pozice

3. Identifikace objektů v BW se provede pomocí transakce RSA1 – BW Admin Cockpit, využije se dokumentace nebo názvů a popisů polí. Zohlednit se musí struktura datového skladu SAP, která odpovídá OLAP úložišti. Samotné tabulky jsou na nejnižší úrovni a jde o nenormalizované protějšky produktivního systému. Následují struktury podle typu a oblasti použití, které postupně obsahují agregace a propojení jednotlivých tabulek, jak je vidět na následujícím obrázku:

Obrázek 12: Struktura SAP BW



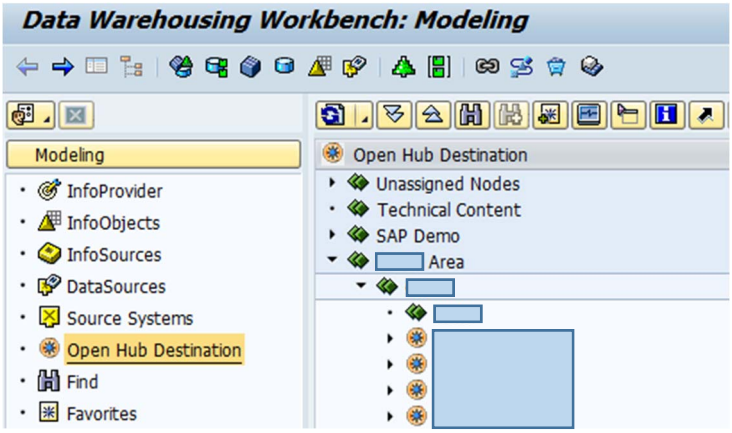
3.1.1 Získávání dat pomocí Open Hub Destination

Open Hub Destination je rozhraní, které umožňuje plánovaně vykopírovat vybraný obsah datového skladu do souborů a tyto přenášet na vzdálený server. Zde se potom těmito daty zaktualizuje lokální databáze.

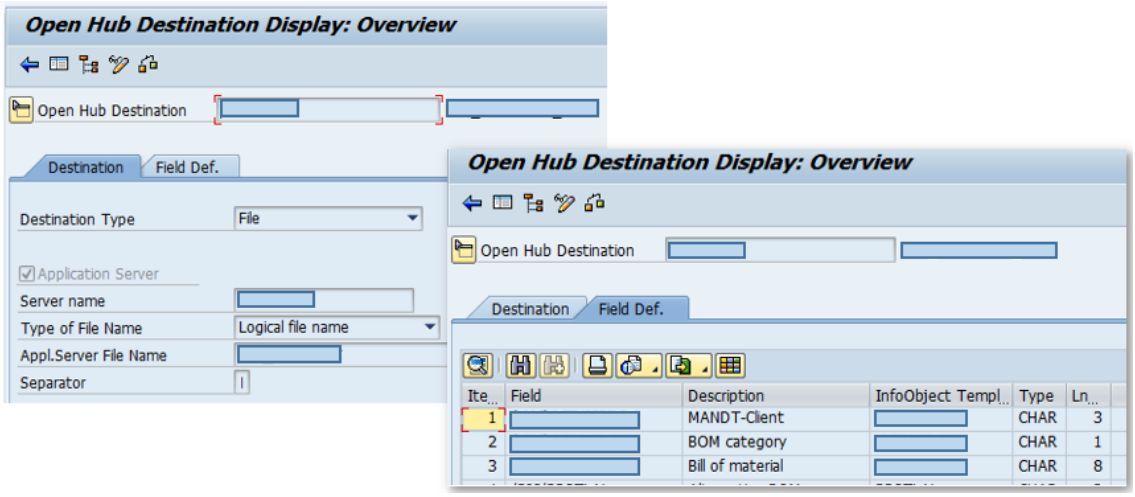
Po identifikaci datových zdrojů je možné nastavit pravidelné vykopírování:

1. Nastavení Open Hub Destination spočívá ve výběru zdrojové tabulky, selekci relevantních dat a v nastavení souboru pro vykopírování.

Obrázek 13: Výběr zdroje v SAP BW



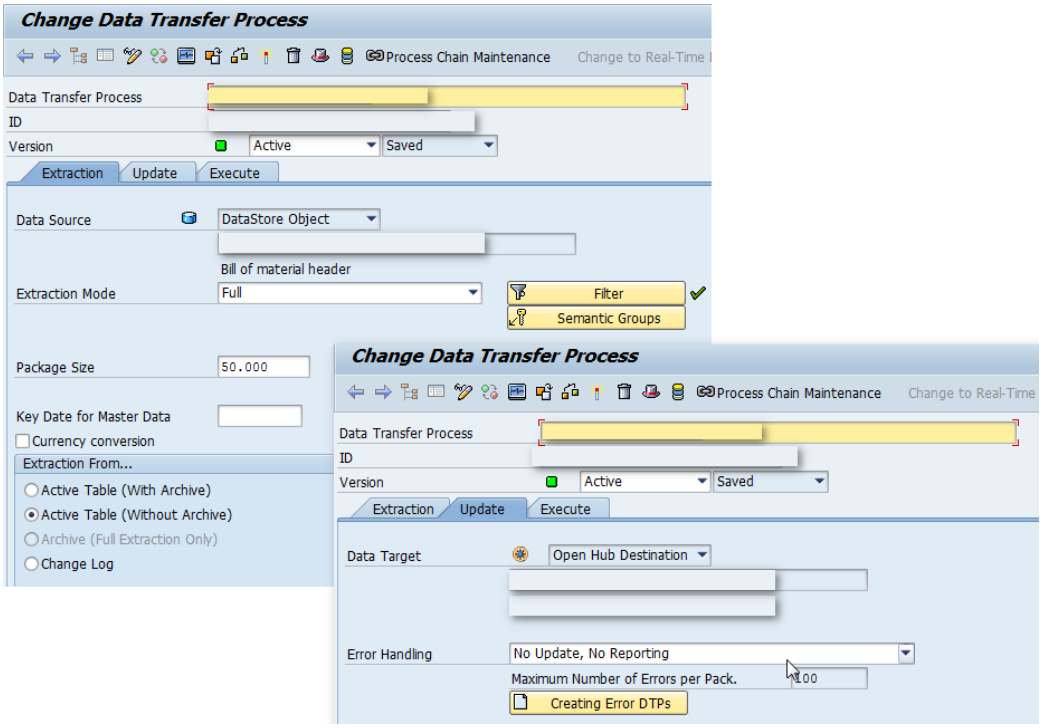
Obrázek 14: Určení cílového souboru a výběr polí



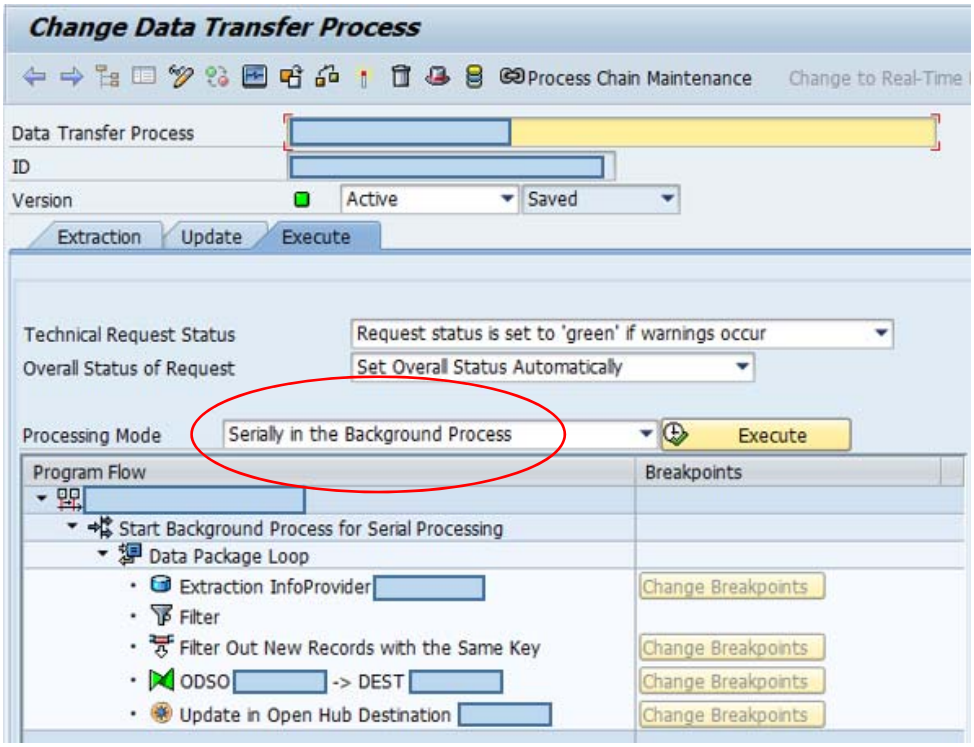
V následujícím kroku lze definovat i relevantní záznamy pro extrakci nastavením filtru.

2. Následuje nastavení přenosu souborů prostřednictvím transakce RSPC – Process Chain Maintenance pro DTP (Data Transfer Proces). Zde je možné nastavit způsob zpracování souborů – sériově, paralelně, a podobně

Obrázek 15: Nastavení extrakce záznamů



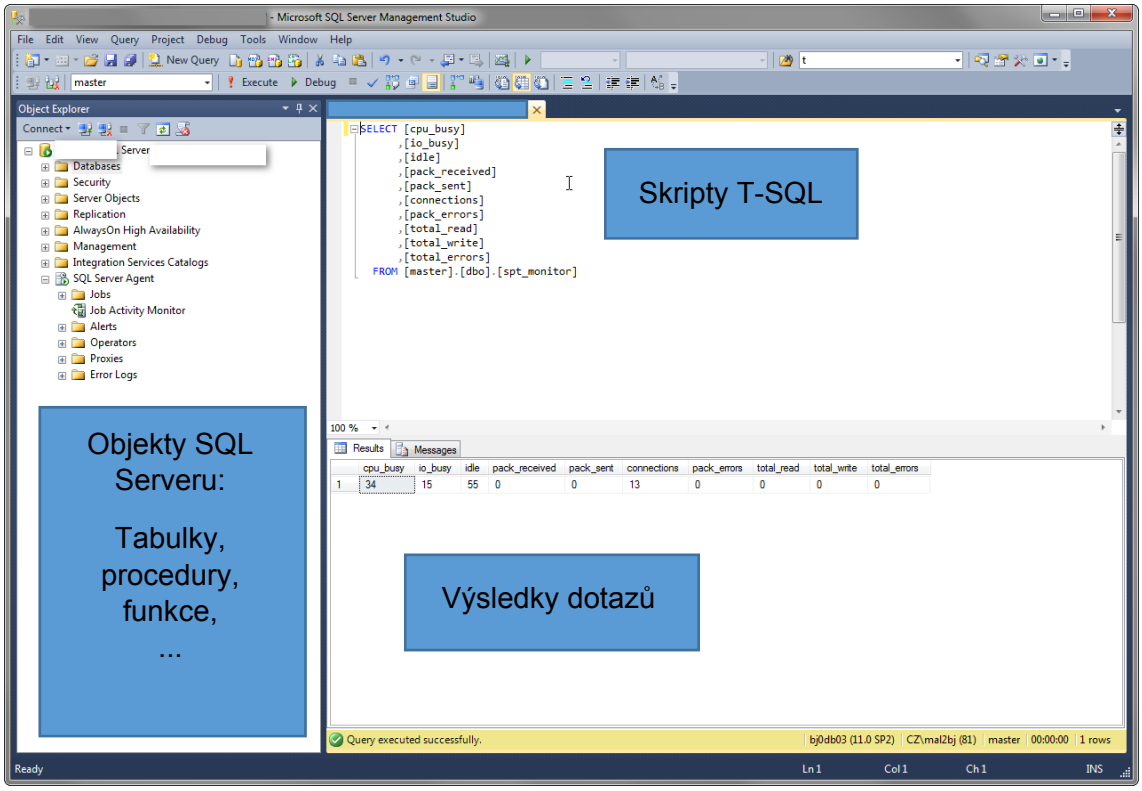
Obrázek 16: Přehled a nastavení DTP



3. Příprava odpovídajících tabulek v databázi SQL Serveru

V SQL Server Management studiu lze psát příkazy přímo, nebo je možné využít příkazů připravených (pravým tlačítkem například na tabulku a *Script Table as – CREATE To – New Query Editor Window*

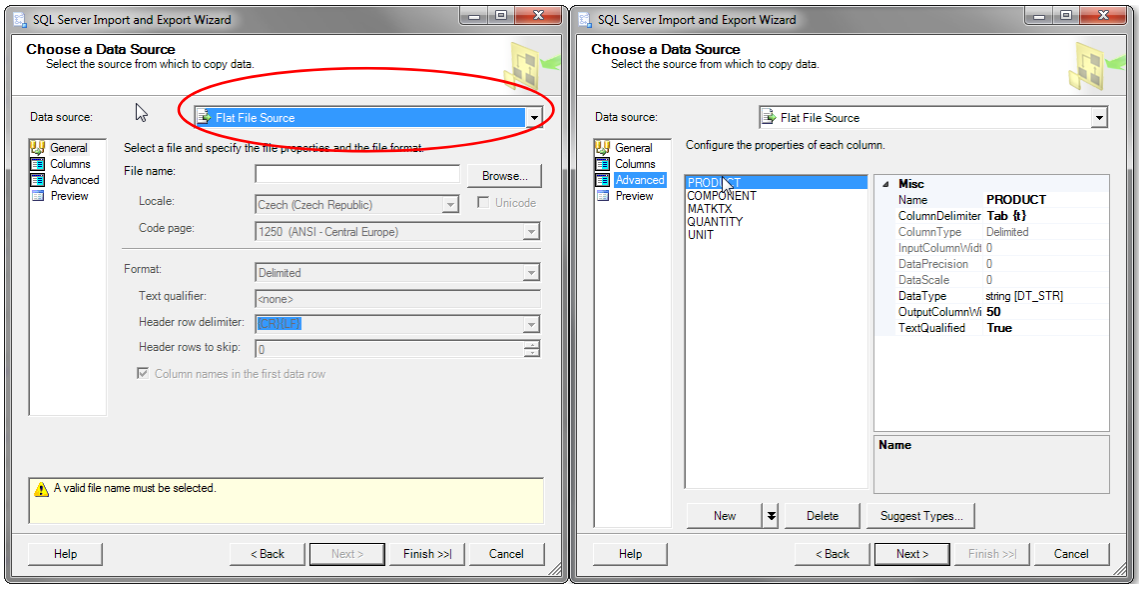
Obrázek 17: Prostředí SQL Server Management Studio



Kromě této možnosti je možné tabulky vytvářet prostřednictvím průvodce při prvním importu souboru – stiskem pravého tlačítka myši nad konkrétní databází, kde bude nová tabulka umístěna, a následně *Tasks – Import Data...*

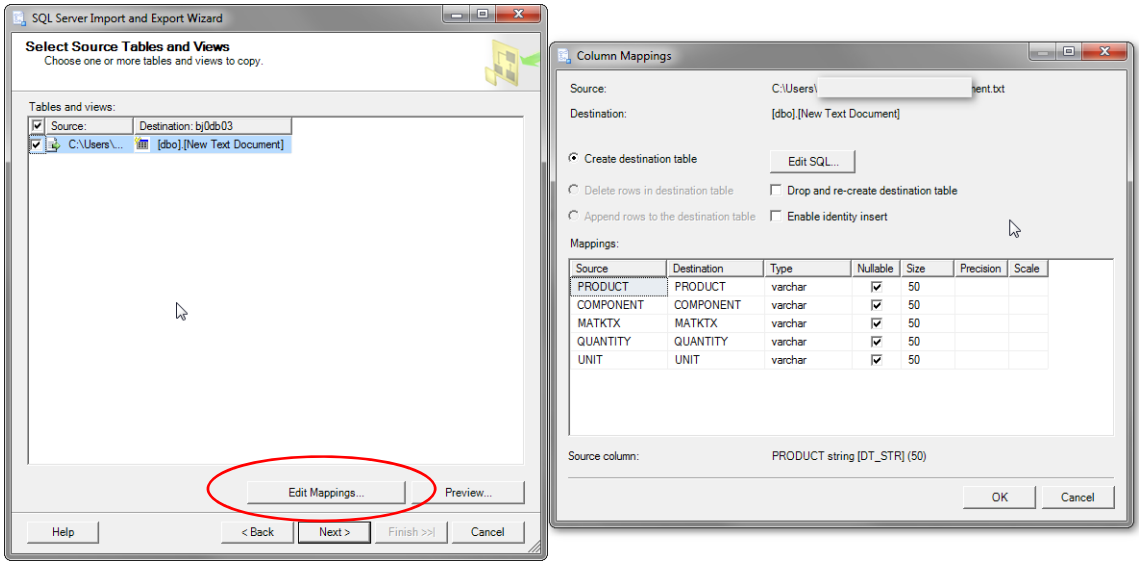
Spustí se průvodce pro import, který je stejný, jako při použití SQL Server Data Tools:

Obrázek 18: Nastavení importu ze souboru



Na záložce *Advanced* se nastavují vlastnosti sloupců tak, jak jsou ve zdrojovém souboru.

Obrázek 19: Nastavení cílové tabulky



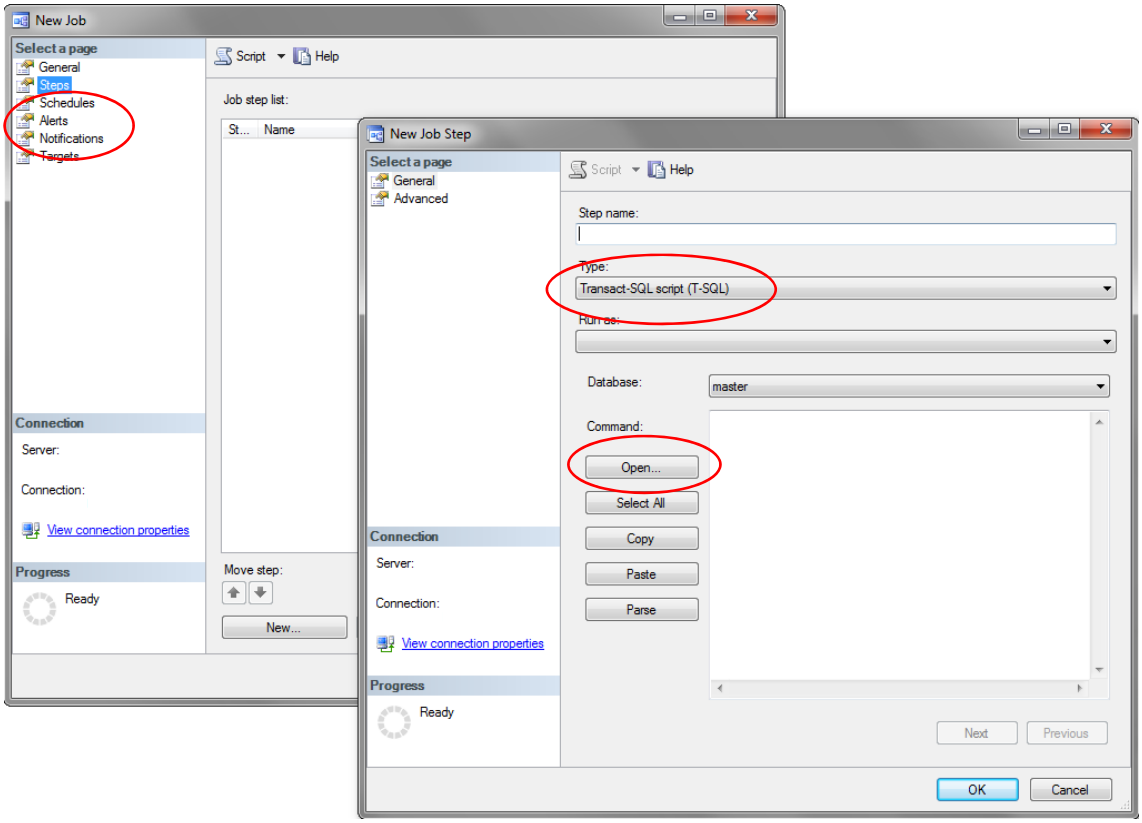
Nakonec se nastaví vlastnosti sloupců cílové tabulky.

V případě použití vyšších edic SQL Serveru lze uvedené kroky průvodce uložit jako balíček integračních služeb (SSIS).

4. Plánování importu prostřednictvím SQL Server Agenta (T-SQL či SSIS balíček)

V Management Studiu se rozklikne SQL Server Agent a pravým tlačíkem myši na Jobs se vybere *New Job...* – spustí se založení nové úlohy:

Obrázek 20: Vytvoření nové úlohy SQL Server Agenta



Je možné vepsat T-SQL skript, nebo nějaký načíst ze souboru, anebo se může vybrat uložený balíček SSIS.

Nakonec se na záložce *Schedules* naplánuje spouštění úlohy (obdobně jako plánování událostí v Outlooku). Také je možné na záložce *Notifications* nastavit zasílání informací o neúspěšném provedení úlohy například mailem. K tomu je nutné nadefinovat operátora v položce *Operators* v menu SQL Server Agent.

Samotný příkaz pro import je jednoduchý – nejprve se vymaže cílová tabulka a následně se nahraje daty ze zmíněného souboru. Je možné určit oddělovač sloupců a řádků nebo formátovací soubor (zde v řádkovém komentáři), který oddělovače obsahuje, a který je níže

```
TRUNCATE TABLE DBase.dbo.tab

BULK INSERT
   DBase.dbo.tab
      FROM 'UNC cesta k souboru'
      WITH (
            FIELDTERMINATOR = '|',
--          FORMATFILE = 'UNC cesta k XML souboru',
            ROWTERMINATOR = '0x0A'
           )
```

Obrázek 21: Příklad XML Format file

```
<?xml version="1.0"?>
<BCPFORMAT xmlns="http://schemas.microsoft.com/sqlserver/2004/bulkload/format"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <RECORD>
  <FIELD ID="1" xsi:type="CharTerm" TERMINATOR="|"/>
  <FIELD ID="2" xsi:type="CharTerm" TERMINATOR="|"/>
  <FIELD ID="3" xsi:type="CharTerm" TERMINATOR="|"/>
  <FIELD ID="4" xsi:type="CharTerm" TERMINATOR="|"/>
  <FIELD ID="5" xsi:type="CharTerm" TERMINATOR="|"/>
  <FIELD ID="6" xsi:type="CharTerm" TERMINATOR="|"/>
  <FIELD ID="7" xsi:type="CharTerm" TERMINATOR="|"/>
  <FIELD ID="8" xsi:type="CharTerm" TERMINATOR="|"/>
  <FIELD ID="9" xsi:type="CharTerm" TERMINATOR="|"/>
  <FIELD ID="10" xsi:type="CharTerm" TERMINATOR="|"/>
  <FIELD ID="11" xsi:type="CharTerm" TERMINATOR="|"/>
  <FIELD ID="12" xsi:type="CharTerm" TERMINATOR="|"/>
  <FIELD ID="13" xsi:type="CharTerm" TERMINATOR="\n"/>
 </RECORD>
 <ROW>
  <COLUMN SOURCE="1" NAME="NOTIFICATN" xsi:type="SQLVARYCHAR"/>
  <COLUMN SOURCE="2" NAME="ZAEHL" xsi:type="SQLVARYCHAR"/>
  <COLUMN SOURCE="3" NAME="CPUTIM" xsi:type="SQLVARYCHAR"/>
  <COLUMN SOURCE="4" NAME="STAR" xsi:type="SQLVARYCHAR"/>
  <COLUMN SOURCE="5" NAME="AENNR" xsi:type="SQLVARYCHAR"/>
  <COLUMN SOURCE="6" NAME="DATE" xsi:type="SQLVARYCHAR"/>
  <COLUMN SOURCE="7" NAME="LASET" xsi:type="SQLVARYCHAR"/>
  <COLUMN SOURCE="8" NAME="TCTUSERNM" xsi:type="SQLVARYCHAR"/>
  <COLUMN SOURCE="9" NAME="TCTSYSID" xsi:type="SQLVARYCHAR"/>
  <COLUMN SOURCE="10" NAME="AENST" xsi:type="SQLVARYCHAR"/>
  <COLUMN SOURCE="11" NAME="RECORDMODE" xsi:type="SQLVARYCHAR"/>
  <COLUMN SOURCE="12" NAME="TYPE2" xsi:type="SQLVARYCHAR"/>
  <COLUMN SOURCE="13" NAME="USER_AP" xsi:type="SQLVARYCHAR"/>
</BCPFORMAT>
```

Stává se, že import neprobíhá z důvodu chybějícího oprávnění ke vzdálené složce. Záleží totiž na nastavení oprávnění složky a účtu, pod kterým služba SQL Server Agent na serveru běží[29].

[29] https://msdn.microsoft.com/en-us/library/ms186264.aspx

Také by bylo možné ověřit před pokusem o import množství záznamů ve zdrojovém souboru, ale skutečnost, že je ve zdrojovém souboru méně záznamů než ve stávající tabulce, nutně neznamená, že jde o chybu. Tabulky pro import by spíše měly sloužit jako dočasné úložiště a teprve odsud by se měla data replikovat pro použití v aplikacích.

Nevýhody:

- Textové soubory mohou obsahovat různé znaky, těžko se stanoví oddělovač sloupců, který by se nemohl objevit v libovolném textovém poli. Problém by se mohl odstranit použitím souborů XML, které v tomto případě není možné
- Bulk insert ve verzích SQL Serveru do 2016 nepodporuje kódování UTF-8, datové soubory tedy nesmí používat toto kódování anebo se musí nastavit větší délka polí v cílové tabulce, aby při importu nedocházelo k oříznutí a proces neskončil chybou
- Přenos trvá dlouho, závisí i na nastavení FTP serveru

3.1.2 Získávání dat pomocí BODS

Pokud je to možné, je lepší použít nástroje ETL procesu. Odstraní se tím nevýhody přenosu přes Open Hub.

BODS (BusinessObjects Data Services) je obdobou SQL Server Data Services a způsob přípravy procesů pro přenos dat a propojování zdrojů se podobá přípravě balíčků SSIS v SQL Server Data Tools a jejich plánování prostřednictvím služby SQL Server Agent. Příprava také probíhala prostřednictvím centrálního IT oddělení a neměl jsem možnost do procesu příliš zasahovat. Proto popíšu pouze to nejnutnější pro představu o prostředí BODS. Na stránkách http://help.sap.com/businessobject/product_guides/ jsou k dispozici manuály s detailním popisem.

Zjištění zdroje dat odpovídá způsobu popsaném v předchozí části. Samotné nastavení procesu v BODS také není složité. Jde o „zaklikávací“ prostředí, na rozdíl od procesu „přes soubory“ se zde musí přesně nadefinovat propojení požadovaných sloupců zdroje se sloupci cílové databáze. Převody datových typů jsou možné. Problémy s různými datovými typy systémů SAP a SQL Server tedy nenastávají.

Na straně SQL Serveru je činnost v tomto případě výrazně jednodušší. Je nutné rozmyslet zajištění dat v případě neúspěšného importu, tabulky mohou být dočasné jako v případě propojení přes Open Hub nebo mohou být pravidelně zálohované.

K tomu je vhodné zjišťovat následujícím dotazem do meta dat stavy v intervalech odpovídajících replikacím. Následující dotaz (součást dále uvedené procedury BODSDataOverview) vrací přehledně seznam tabulek náležejících určitému schématu a počet záznamů v nich:

```
SELECT
    TableName = t.NAME,
    TableSchema = s.Name,
    RowCounts = p.rows
FROM
    sys.tables t
INNER JOIN
    sys.schemas s ON t.schema_id = s.schema_id
INNER JOIN
    sys.indexes i ON t.OBJECT_ID = i.object_id
INNER JOIN
    sys.partitions p ON i.object_id = p.OBJECT_ID AND i.index_id = p.index_id
WHERE
    t.is_ms_shipped = 0 and s.Name in ('schema1', 'schema2')
GROUP BY
    t.NAME, s.Name, p.Rows
ORDER BY
    s.Name, t.Name
```

Schématem v SQL Serveru se rozumí objekt oprávnění a organizace databáze.

Další procedura potom využívá údajů z výše uvedené procedury k záznamu denního stavu tabulek.

```
create table #Overview (TableName nvarchar(10),
                        TableSchema nvarchar(2),
                        RowCounts int
                       )
insert into #Overview
exec DBase.dbo.BODSDataOverview;

select *,
       getdate() as ReportDate
into #Log
  from #Overview

insert into DBase.dbo.LogBODS
select *
  from #Log

drop table #Overview
drop table #Log
```

Nejprve načte data z jedné procedury do dočasné tabulky, načte aktuální datum (funkce *getdate*) a zaktualizuje přehled. Nakonec dočasné tabulky zahodí.

Nevýhody při použití datového skladu jako zdroje podnikových dat vznikají v případech, kdy uživatel nemá vliv na změny v datovém skladu:

- I když budou splněny podmínky uvedené v záhlaví kapitoly, může se stát, že použití datového skladu jako zdroje pro některé aplikace nebo pokročilé reporty nebude úplně vhodné. To může nastat v případě, že data jsou zdánlivě kompletní a dostatečně aktuální, ale tvoří hierarchickou strukturu, která je ukládána pro jiné účely, než pro které je nutné ji využít. Při použití se ukáže, že tvorba dotazů nad takovými daty je obtížná.
- Nové verze nebudou obsahovat potřebná data nebo se změní způsob aktualizace na méně častý/vhodný.

3.2 Připojení k produktivnímu systému SAP prostřednictvím BODS

V některých případech může být získávání dat produktivního systému prostřednictvím datového skladu nevhodné nebo přímo nemožné – v datovém skladu nemusejí být uchovávány všechny potřebné informace/objekty nebo aktualizace datového skladu v případě některých informací/objektů neprobíhá podle potřeb a data tak nejsou dostatečně aktuální a současně je problém takovou skutečnost změnit.

Produktivní systém SAP (R/3) sice neumožňuje využít Open Hub jako v případě SAP BW, ale je možné využít BODS, který je pro extrakci dat určen. V případě připojování se k produktivnímu systému pomocí ETL nástroje se potom ale musí řešit stejné úlohy, jako v případě přípravy připojení produktivního systému k datovému skladu. Cílový systém je v podstatě v pozici datového skladu, ačkoli nedochází k úplnému ETL procesu (transformace je do značné míry omezena především z důvodu rychlosti – pokud se připojuji k produktivnímu systému, kde jsou 100% aktuální data, proč neaktualizovat cílovou databázi častěji než jednou denně – ale na druhou stranu při častější aktualizaci je počet potrefených záznamů relativně malá). Připomenu, že k extrakci dochází zvenku, to znamená, že při aktualizaci zdrojové databáze (produktivního systému) není nastaven trigger, který by mohl aktualizaci či mirroring spouštět, ale ověření aktualizovaných údajů musí být zajištěno prostřednictvím ETL nástroje.

3.2.1 Changed data capture (CDC)

Připojení k datovému skladu znamená zorientovat se ve struktuře OLAP úložiště, naproti tomu je vyřešena aktualizace. V případě připojení k produktivnímu systému strukturu známe, je ale nutné nadefinovat způsob aktualizace cílové databáze, tedy identifikaci změněných či smazaných údajů (negativních záznamů proti datovému skladu).

V oblasti datových skladů je to poměrně rozsáhlé téma, existují metody a typy jak vybírat změněná data a jak aktualizovat datový sklad. Já jsem musel vyjít z možností, které nástroj BODS nabízel, a ty byly v podstatě dvě. Importovat vždy celý obsah znovu, anebo nahrávat pouze změněné záznamy a naopak mazat ty v původním systému smazané. Také jsem musel zohlednit způsob editace záznamů ve zdrojovém systému. Všechny potrefené tabulky v systému SAP, které jsme potřebovali, obsahují sloupce s údaji o vytvoření a

poslední změně záznamu – to by bylo ideální řešení pro aktualizaci. Problém nastává s identifikací smazaných záznamů. Někdy se totiž u mazaného záznamu ukládá informace, že je určený k výmazu, při použití některých transakcí je ale možné záznam vymazat rovnou bez této informace. Práce s těmito negativními záznamy nebyla do okamžiku dokončení práce dořešena.

3.2.2 Činnosti pro připojení k produktivnímu systému prostřednictvím BODS

V porovnání s přístupem k datovému skladu se zde po vyjasnění strategie s CDC nic zásadního neřeší. Zdrojem jsou v tomto případě přímo databázové tabulky. U SAP kusovníku to jsou zmíněné MAST, STKO, STPO a STAS. Do těch se bude BODS pravidelně dotazovat a zjišťovat nové, změněné a smazané záznamy. Vždy samozřejmě s ohledem na zachování konzistentních dat v cílové databázi. Postup pro připojení prostřednictvím BODS je popsaný v minulé kapitole. Na straně SQL Serveru se vytvoří odpovídající struktura. Zde je také navíc možné zapnout nad databází funkci *Change Tracking*[30] a sledovat změny v tabulkách. Odlišná situace by nastala v případě, že by se měla struktura zdrojové databáze při přenosu nějak měnit a cílová databáze tak vypadala jinak. Potom by se musela nastavit pravidla pro transformaci.

Já jsem pro testovací provoz zvolil pravidelné vykopírování všech vyhovujících dat se selekcí na straně zdroje a před importem vždy došlo k vymazání cílové tabulky. Přenos v případě tabulek o několika stech tisících záznamů trvá řádově desítky sekund a jsou tím vyřešeny i vymazané záznamy. Pokud by se v BODS podařilo nastavit logiku tak, aby se podařilo spolehlivě identifikovat i tyto smazané záznamy, bude samozřejmě lepší použít Delta Load – přenášejí se pouze aktualizovaná data.

Pokud se povede takový proces zprovoznit, jsou k dispozici téměř online data. Ze způsobů, kdy není možné připojit se k systému SAP přímo standardním způsobem, je to asi nejvhodnější řešení.

[30] https://msdn.microsoft.com/en-us/library/bb933875.aspx

3.3 Příprava dat pro webovou aplikaci

T-SQL dotazy se nemusejí omezovat pouze na používání agregačních funkcí, ačkoli i to zrychluje přípravu a načítání dat pro webové aplikace. Je pochopitelné, že programátora v PHP bude složitější dotaz v T-SQL mást, stejně jako naopak pro SQL specialistu není jednoduché se zorientovat v práci s PHP frameworkem bez proškolení.

Uvedl bych proto několik komentovaných příkladů kódů v T-SQL, které jsem použil při přípravě dat pro použití ve webové aplikaci. Funkce, které vykonávají, tak nemusely být řešeny v PHP.

Jednotlivé dotazy jsem tvořil s využitím literatury (Molinaro, 2009) či (Stack Overflow, 2016), nejsou tedy neobvyklé. Řadu vhodný řešení lze nalézt i mezi vzory přímo na stránkách o SQL Serveru (Microsoft Developer Network, 2016).

3.3.1 Rekurzivní dotaz

Následující dotaz vrátí tabulku se strukturou výrobku, rozpadem kusovníku:

```
WITH rozpadni
(MATNR, POSTP, POSST, COMPONENT, MATKTX, DATUVP, DATETO, UNIT, MENGE, ALPOS, EWAHR, SANFE, LEVEL, PRED)
AS (
(SELECT MATNR
       ,POSTP
       ,POSST
       ,COMPONENT
       ,MATKTX
       ,DATUVP
       ,DATETO
       ,UNIT
       ,MENGE
       ,ALPOS
       ,EWAHR
       ,SANFE
       ,1
       ,CAST(MATNR AS NVARCHAR(MAX))
FROM tab
WHERE RB_MATNR = 'cislo'
    AND POSTP = 'L'
    AND (DATUVP <= GETDATE())
    AND (DATETO > GETDATE())
)
```

Rekurzivní dotaz pomocí klauzule WITH umožnuje procházet a načítat hierarchické struktury v tabulkách. V první části vždy načte vyhovující záznamy (zadaný dílec – „cislo“ a pozice platné k určitému dni pomocí „getdate“) a v další, rekurzivní části, která

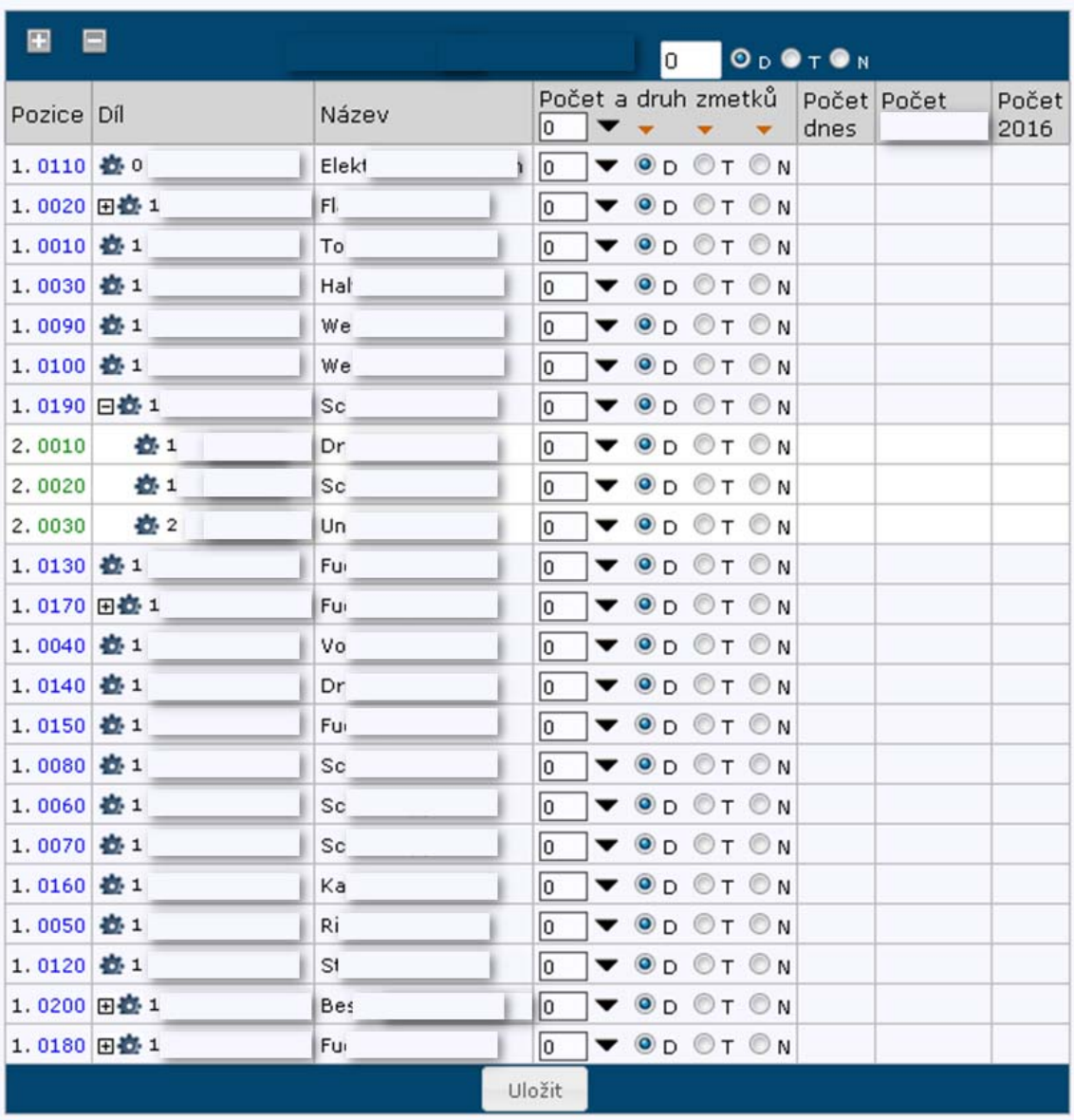
se připojí pomocí operátoru UNION ALL k dočasné struktuře[31] definované v záhlaví klauzule WITH, se dohledávají další – spojením přes INNER JOIN.

```
UNION ALL
(SELECT Z.MATNR
       ,Z.POSTP
       ,Z.POSST
       ,Z.COMPONENT
       ,Z.MATKTX
       ,Z.DATUVP
       ,Z.DATETO
       ,Z.UNIT
       ,Z.MENGE
       ,Z.ALPOS
       ,Z.EWAHR
       ,Z.SANFE
       ,A.LEVEL + 1
       ,PRED + '|' + A.COMPONENT
FROM rozpadni A INNER JOIN tab Z ON Z.MATNR = A.COMPONENT
WHERE Z.POSTP = 'L'
    AND (Z.DATUV <= GETDATE())
    AND (Z.RDATETO > GETDATE())
)
)
SELECT MATNR
      ,POSST
      ,COMPONENT
      ,MATKTX
      ,LEVEL
      ,CAST(PRED AS varchar(MAX)) PRED
      ,CAST(UNIT AS varchar(3)) UNIT
      ,RBMENGE
      ,CAST((PRED + '|' + COMPONENT) AS varchar(MAX)) RAZENI
FROM rozpadni
ORDER BY RAZENI, POSST
```

V další části je zásadní, že se může přičítat každá další úroveň průchodu strukturou (sloupec LEVEL), a že sloupec PRED, který postupně „sčítá“ čísla procházených dílců jako řetězec, může sloužit k řazení výsledků. Logika je vidět přímo v dotazu. Celá struktura se načte do webové stránky, kde se její jednotlivé úrovně mohou pomocí JavaScriptu skrývat. SQL dotaz slouží jako zdroj pro vytvoření tabulky na následující straně.

[31] CTE – Common Table Expression

Obrázek 22: Struktura výrobku podle SQL dotazu



Pozice	Díl	Název	Počet a druh zmetků	Počet dnes	Počet	Počet 2016
1. 0110	0	Elekt	0 D T N			
1. 0020	1	Fl	0 D T N			
1. 0010	1	To	0 D T N			
1. 0030	1	Hal	0 D T N			
1. 0090	1	We	0 D T N			
1. 0100	1	We	0 D T N			
1. 0190	1	Sc	0 D T N			
2. 0010	1	Dr	0 D T N			
2. 0020	1	Sc	0 D T N			
2. 0030	2	Un	0 D T N			
1. 0130	1	Fu	0 D T N			
1. 0170	1	Fu	0 D T N			
1. 0040	1	Vo	0 D T N			
1. 0140	1	Dr	0 D T N			
1. 0150	1	Fu	0 D T N			
1. 0080	1	Sc	0 D T N			
1. 0060	1	Sc	0 D T N			
1. 0070	1	Sc	0 D T N			
1. 0160	1	Ka	0 D T N			
1. 0050	1	Ri	0 D T N			
1. 0120	1	St	0 D T N			
1. 0200	1	Bes	0 D T N			
1. 0180	1	Fu	0 D T N			

Uložit

3.3.2 Příprava tabulky pro HTML rowspan

Dotazy lze využít i pro vytvoření tabulky prezence účastníků sezení a nasčítat jednotlivé výskyty pro HTML *rowspan* – začíná se procedurou přípravy seznamu účastníků (PrepareMTAttendance):

```
declare @mtgs_id int = (select mtgs_id from DBase.dbo.Meeting where ID = @mid);

with x (numorder_1, dept, id_usr, pos, usr, numorder_3, proxy, in_attend, product)
as (
select  a.numorder
      , replace(isnull(a.dept + ' (' + b.add_type + ')', a.dept), '()', '')
      , b.id
      , cast(b.numorder as float)
      , b.usr
      , 0
      , NULL
      , cast('' as nvarchar(7))
      , DBase.dbo.UniteAllArea(@mtgs_id, b.usr)
 from DBase.dbo.Positions a left join DBase.dbo.Approvers b on a.id = b.position_id
 where a.mtgs_id = @mtgs_id
 ),
z (numorder_1, dept, id_usr, pos, usr, numorder_3, proxy, in_attend, product)
as (
select  x.numorder_1
      , x.dept
      , x.id_usr
      , x.pos + (e.numorder * 0.1)
      , x.usr
      , e.numorder
      , e.usr
      , cast('' as nvarchar(7))
      , DBase.dbo.UniteAllArea(@mtgs_id, e.usr)
  from (x inner join DBase.dbo.Proxies e on x.id_usr = e.approver_id)
  left join DBase.dbo.Oblasti v on e.oblast_id = v.id
)
```

Zde se opět pomocí klauzule WITH, která nemusí sloužit pouze hierarchickým dotazům, vytváří několik dočasných tabulek, které se oddělují čárkou a zpracovávají za sebou. Mohou se dotazovat i do sebe navzájem. V tomto případě jde o přípravu seznamu schvalovatelů a jejich zástupců. Schvalovatelé patří do různých oddělení a mají v rámci sezení oprávnění vždy na určitou výrobkovou oblast, o jejich zástupcích platí to samé. V tabulkách se také ukládají informace o pořadí schvalovatelů v tabulce pro prezenci. V této první části se tedy načítají schvalovatelé, jejich zástupci na konkrétní sezení (@mtgsid, @mid) a vytváří se jejich pořadí. Také se pomocí uživatelské funkce UniteAllArea zjistí příslušný produkt – uživatel a jeho přiřazení k produktu je uloženo relačně v tabulce, tato funkce tabulku projde a vrátí seznam produktů jako řetězec.

```
select pos
     , dept
     , usr
     , proxy
     , in_attend
     , product
into #attendance
  from x
union
select pos
     , dept
     , usr
     , proxy
     , in_attend
     , product
  from z
  order by pos

update #attendance
set in_attend = 'checked'
  where pos in (select num from DBase.dbo.Attendance where mt_id = @mid and num = pos)

select *
  from #attendance

drop table #attendance
```

V další části se potom načtou a odešlou data ze struktury a doplní se informací o tom, jestli je políčko ve webovém formuláři již zaškrtnuto či nikoli (in_attend = ‘checked‘) – dotaz může být spuštěn vícekrát během sezení, účastníci mohou docházet postupně. Nakonec se tabulka může zahodit.

Procedura je dále využita procedurou pro konečnou přípravu a nasčítání jednotlivých výskytů schvalovatelů:

```
create table #attendan (pos nvarchar(10)
                      , dept nvarchar(50)
                      , usr nvarchar(10)
                      , proxy nvarchar(10) NULL
                      , in_attend nvarchar(10)
                      , product nvarchar(50)
                      )
insert into #attendan
exec PrepareMTAttendance @mid;
```

Jednoduchým způsobem se výsledek předchozí procedury načte do dočasné tabulky (#attendan).

Následuje postupné napočítání (select distinct count(*) over ...) uživatelů a jejich schvalovatelů, a to vždy vzhledem k jejich výskytu v rámci oddělení, a počet zástupců a jejich schvalovatelů. Vše bude patrné z výsledné tabulky na konci.

```
with w
as
(select distinct count(*) over(partition by usr, dept) occur_usr
            ,usr
            ,proxy
            ,dept
 from #attendan
), x
as
(select distinct count(*) over(partition by dept) occur_usr_noproxy
            ,dept
 from w
 where occur_usr = 1
), y
as
(select distinct count(*) over(partition by dept) occur_usr_wproxy
            ,dept
 from w
 where occur_usr > 1
), z
as
(select distinct count(*) over(partition by dept) occur_proxy_dept
            ,dept
 from w
 where proxy is not null
)
select distinct isnull(x.occur_usr_noproxy, 0) occur_usr_noproxy
            ,isnull(y.occur_usr_wproxy,0) occur_usr_wproxy
            ,isnull(z.occur_proxy_dept, 0) occur_proxy_dept
            ,w.dept
into #usr_struct_dept
from w
left join x
on w.dept = x.dept
left join y
on w.dept = y.dept
left join z
on w.dept = z.dept
```

Z názvů polí by mělo být patrné, o kterou položku zrovna jde. Funkce *isnull* nahrazuje hodnotu NULL nulou.

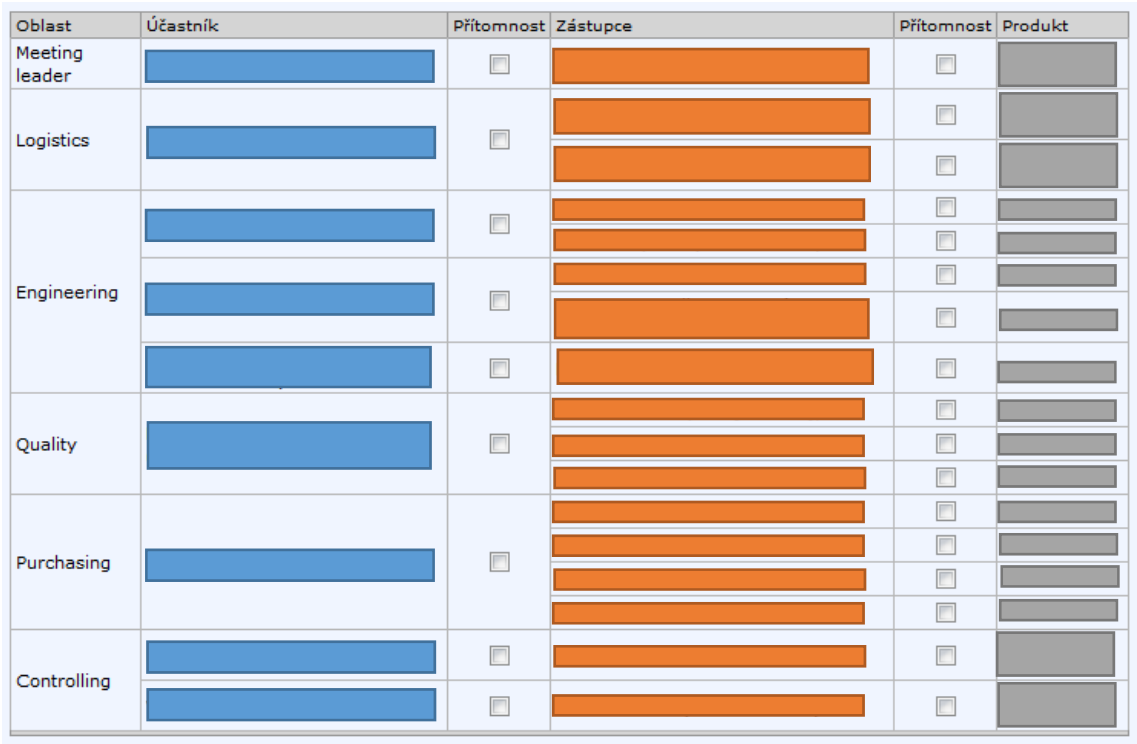
Mezistav je uchován v dočasné tabulce #usr_struct_dept a odtud se použije ke konečnému odeslání.

```
select @mid mid
      ,a.pos
      ,isnull(b.occur_proxy_dept, 0) + b.occur_usr_noproxy span_dept
      ,a.dept
      ,isnull(c.occur_proxy_usr, 0) span_usr
      ,a.usr
      ,a.proxy
      ,a.in_attend
      ,a.product
from #attendan a
left join
#usr_struct_dept b
on a.dept = b.dept
left join
(select distinct count(*) over(partition by usr, dept) occur_proxy_usr, usr, dept
 from #attendan where proxy is not null) c
on a.usr = c.usr
and a.dept = c.dept

drop table #usr_struct_dept
drop table #attendan
```

Výběrem z CTE a posledním sečtením výskytů zástupců jednotlivých uživatelů procedura a dotaz končí. Sčítání zástupců (poslední *left join*) nemohlo být provedeno v rámci klauzule WITH, protože se jejich výskyt počítá z jiné množiny, než všechny ostatní výskyty, pro které byla struktura použita.

Obrázek 23: Tabulka se sloučenými řádky podle SQL dotazu



Doufám, že záměr je z tabulky patrný. Počty schvalovatelů, zástupců i produktů se vždy nějak liší.

3.3.3 Zpracování XML na straně SQL Serveru

Smyslem následující zjednodušené procedury je zpracování XML souboru či řetězce a aktualizace tabulky těmito údaji. Využívá se standardní procedury *sp_xml_preparedocument*, která připraví odpovídající strukturu a odkáže na ni prostřednictvím *@idoc*. Potom je nutné, aby odesílaný XML řetězec měl níže uvedenou strukturu (zde pouze ID a rating_txt). Lze tak řešit odeslání celého formuláře z webové aplikace a jeho zpracování na straně SQL Serveru najednou.

```
CREATE PROCEDURE [dbo].[EditRating]
@XMLDoc nvarchar(MAX)
AS
BEGIN

DECLARE @idoc INT

EXEC sp_xml_preparedocument @idoc OUTPUT, @XMLDoc

SELECT *
INTO #RATING
FROM OPENXML (@idoc, '/ROOT/rating')
  WITH (ID int,
        rating_txt text);

MERGE DBase.[dbo].[Docs] Z
USING #RATING P
ON Z.ID = P.ID
WHEN MATCHED
  THEN UPDATE SET Z.rating_txt = P.rating_txt;

DROP TABLE #RATING

EXEC sp_xml_removedocument @idoc

END
```

Po načtení XML do dočasné tabulky #RATING dojde pomocí klauzule MERGE k porovnání zvolené databázové tabulky a její aktualizaci. Záleží na tom, co vše se pomocí WHEN nadefinuje. Mohou se mazat chybějící záznamy, doplňovat nové ale i pouze aktualizovat stávající.

3.3.4 Příklad zpracování na straně PHP

Zde je část PHP kódu pro zpracování tabulky z příkladu s HTML *rowspan*. S každou řádkou, na které začíná nové oddělení a také nový schvalovatel, přichází i informace o tom, kolik řádek se má sloučit. Potom se musí zajistit, aby se odpovídající počet následujících řádek nezobrazoval a *rowspan* fungoval správně.

```php
<?php

$sqlParticipant = mssql_query("exec DBase.dbo.ReadMTAttendance ".$mid."");

$runDept = 0;
$runProxy = 0;

echo "<table cellpadding=\"0\" id=\"tblData\" cellspacing=\"0\" border=\"0\">";
echo "<thead><tr><th>".MEDIUM."</th><th>".UCASTNIK."</th><th>".PRITOMNOST."</th><th>".PROXY."</th><th>".PRITOMNOST."</th><th>".PRODUCT."</th></tr></thead>";
echo "<tfoot><tr><th colspan=\"6\">".$msg."</th></tr></tfoot>";
echo "<tbody>";
while ($participant = mssql_fetch_array($sqlParticipant)) {
//-------------------------------------------------------------------------------------------------------------------------------//
  if ((strlen($participant[6]) == 0) and ($participant[4] > 0)) {       // schvalovatel se zastupcem
    if ($runDept == 0) {
      echo "<tr><td rowspan=".$participant[2]."><font class=\"txt\">".$participant[3]."</font></td>";
    } else {
       echo "<tr>";
    }
    echo "<td rowspan=".$participant[4]."><font class=\"txt\">".LDAPInfo($participant[5], "displayname")."</font></td>";
    echo "<td rowspan=".$participant[4]." align=\"center\"><form method=\"post\"><input type=\"checkbox\" name=\"pritomnost[]\" value=\"".$participant[1]."\" ".$participant[7].">";
  }
//-------------------------------------------------------------------------------------------------------------------------------//
  if ((strlen($participant[6]) == 0) and ($participant[4] == 0)) {      // schvalovatel bez zastupce
    if ($runDept == 0) {
      echo "<tr><td rowspan=".$participant[2]."><font class=\"txt\">".$participant[3]."</font></td>";
    } else {
       echo "<tr>";
    }
    echo "<td><font class=\"txt\">".LDAPInfo($participant[5], "displayname")."</font></td>";
    echo "<td align=\"center\"><form method=\"post\"><input type=\"checkbox\" name=\"pritomnost[]\" value=\"".$participant[1]."\" ".$participant[7].">";
    echo "<td colspan=2></td>";
    echo "<td><font class=\"txt\">".$participant[8]."</font></td>";
    echo "</tr>";
    $runDept++;
    if ($runDept == $participant[2]) {
      $runDept = 0;
    }
  }
```

Při tvorbě tabulky se rozlišuje, který řádek se zrovna převádí – nové oddělení, nový schvalovatel či pouze zástupce a sleduje se počet proběhnutých řádků v rámci oddělení a schvalovatele.

```php
//-------------------------------------------------------------------------------------------------------------------------------//
  if ((strlen($participant[6]) != 0)) {                                 // zastupci
    if ($runProxy == 0) {                                               // I. zastupce
        if ($participant[6] == '-') {                                   // existuje zastupce pro dilci produkt?
            echo "<td colspan=2><font class=\"txt\">".NO_PROXY."</font></td>";
        } else {
           echo "<td><font class=\"txt\">".LDAPInfo($participant[6], "displayname")."</font></td>";
        }
    } else {                                                            // dalsi zastupce
        echo "<tr><td><font class=\"txt\">".LDAPInfo($participant[6], "displayname")."</font></td>";
    }
    if ($participant[6] != '-') {                                       // neexistuje-li zastupce pro dany produkt, nelze vyplnit dochazku
        echo "<td align=\"center\"><form method=\"post\"><input type=\"checkbox\" name=\"pritomnost[]\" value=\"".$participant[1]."\" ".$participant[7].">";
    }
    echo "<td><font class=\"txt\">".$participant[8]."</font></td>";
    echo "</tr>";

    $runProxy++;
    if ($runProxy == $participant[4]) {
        $runProxy = 0;
    }
    $runDept++;
    if ($runDept == $participant[2]) {
      $runDept = 0;
    }
  }
//-------------------------------------------------------------------------------------------------------------------------------//
}
echo "</tbody></table>";
echo "<p><colspan=\"2\"><input type=\"submit\" name=\"odeslat\" value=\"".ULOZIT."\"></form></td></p>";

echo "</tbody></table>";
```

Zde je část PHP kódu, který je použitý při odeslání XML řetězce s daty formuláře stránky. V tomto případě tabulka, dvourozměrné pole se převede názorně do XML a odešle jako parametr proceduře ke zpracování

```php
//-----------------------------------------------------------------------------------------------------------------------------------------//
if ($_POST["send"]) {
    $rating = "<?xml version=\"1.0\"?><ROOT>";
    foreach($_POST["rating"] as $key => $value) {
        $rating = $rating . "<rating ID=\"" . $key . "\" ";
//
//      TODO: XML prostrednictvim SimpleXML
//
        $rating = $rating . "rating_txt=\"" . $value . "\"/>";
    }
    $rating = $rating . "</ROOT>";
    $sqlRating = mssql_query("exec Procedura '".$rating."'");
    $msg = '<p align="center" class="status_message" id="hlaseni_ok"><b><font color="white">Uloženo</font></b></p>';
}
//-----------------------------------------------------------------------------------------------------------------------------------------//
```

4 Závěr

SAP R/3 je specifický ERP systém, používá vlastní názvy a zkratky. Nezvyklé prostředí, menu a někdy z pohledu uživatelů i chování vede ke značně rezervovanému přístupu k tomuto systému. Pokud se k tomu přidá i nepružná organizace požadavků na vývoj systému a implementaci procesů, končí to externími aplikacemi, mnohdy kvůli podobnosti požadavků vícenásobně vyvíjeným. Systém obsahuje data kmenových podnikových procesů, ke kterým je nutné se v takových případech dostat, aby se nemusela neustále kopírovat či přepisovat.

Připojení k datovému skladu je vhodná možnost využití produktivních dat bez nutnosti napojení na produktivní systém, u kterého se mohou vyskytnout problémy s oprávněním a obavy z přetěžování produktivního systému. Ne vždy je možné takové řešení použít, potom může být jedinou možností připojení k produktivnímu systému pomocí ETL procesu.

Ukázalo se, že obě možnosti jsou ohledem na podmínky podrobně popsané v práci možné a spolehlivě řeší tyto problémy. ETL proces je provozovaný centrálním IT oddělením, je tak zaručeno dodržení bezpečnostní politiky.

V teoretické části jsem se zaměřil na souhrn takových informací, jaké požadovalo zadání práce, a na které jsem se dále v praktické části odkazoval nebo je jinak používal.

Při popisu systému SAP a jeho součástí jsem se zaměřil více na prezentaci informací, které nejsou vždy dostupné z oficiálních zdrojů nebo bývají hodně obecné. Popisoval jsem to, co se dále ať již v teoretické části a nakonec i praktické používalo, na co jsem odkazoval. Praktická část potom obsahovala zejména popis způsobu provedení.

Dále jsem měl nalézt řešení pro napojení k ERP systému SAP k lokální databázi. To se také podařilo. Možností jsem objevil, ale i ověřil několik. U každé jsem uvedl zhodnocení a její význam.

Posledním požadavkem bylo navržení možnosti oddělení logiky od prezentace pro určitý typ webové aplikace a pro potřeby konkrétního strojírenského podniku. Určitý způsob jsem navrhl a úspěšně ověřil v provozu podle podmínek firmy.

Další zajímavé potvrzení je, že jazyk T-SQL je skutečně účinným nástrojem pro manipulaci s daty a jejich přípravu pro webové aplikace. Použití XML jako prostředku pro odeslání dat z webové aplikace ke zpracování do databázového systému prostřednictvím procedur se také osvědčilo.

Je škoda, že s ohledem na některé nezveřejnitelné informace, jsem příkladů nemohl uvést více.

Práce dokládá, že i v době vysoké integrace podnikových systémů je stále nutné hledat řešení pro napojování různých systémů a tvorbu řady aplikací. Propojení základního podnikového systému a lokální aplikace je výhodné z důvodu odstranění duplicitního zadávání stále stejných údajů, které jsou právě v ERP systému zpravidla spravované. Ne vždy je ERP systém na lokální potřeby připraven, obsahuje ale data, která bez nutnosti dalšího zpracování mohou sloužit právě dalším aplikacím.

Bylo poměrně těžké popisovat tu část informatiky, která je dostatečně známa, a přitom se nedopouštět opakování všeobecně známých skutečností. Pátraní v původních zdrojích bylo ale zajímavým doplněním mého vzdělání a příjemně mě překvapilo. Snad se mi podařilo s dostatečným spádem popsat okolnosti, které k současnému stavu vedly, stejně jako poznatky, které utvářely současná pravidla.

I. Summary

The thesis deals with connecting SAP ERP system via local database system MS SQL Server using the tools SAP BI, data synchronization between systems and advanced usage of T-SQL language for preparing data for web applications and reports written in PHP. The thesis contains a brief overview of the SAP system and the possibility of connecting to the SAP system. The general principles of described solution can be used in conjunction with other systems and programming languages.

I discovered that the T-SQL is a really powerful tool for data preparation of Web applications. It can be used for data manipulation, for setting up a connection between systems and for XML processing as well.

It was relative difficult to show all the possibilities – I could not use all the screenshots and parts of code – sometimes I had to use a simplified version of a script, because the data are partially secret. It is not so easy to describe the meaning of a T-SQL script without a demonstration.

II. Keywords

Relational database, ERP system, SAP, SAP BW, MS SQL Server, T-SQL, stored procedures, user defined functions

III. Seznam použitých zdrojů

Codd, E. F. (Červen 1970). A relational model of data for large shared data banks. *Communications of the ACM*, stránky 377-387.

CS145 Introduction to Databases. (2016). Retrieved from Stanford University: http://web.stanford.edu/class/cs145/

Kroenke, D. M., & Auer, D. J. (2015). *Databáze.* Brno: Computer Press.

Kühnhauser, K.-H. (2009). *ABAP: výukový kurz.* Brno: Computer Press.

Lacko, Ľ. (2013). *Mistrovství v SQL Server 2012:[kompletní průvodce databázového experta]*. Brno: Computer Press.

Maassen, A., Schoenen, M., Frick, D., & Gadatsch, A. (2007). *SAP R/3: kompletní průvodce.* Brno: Computer Press.

Microsoft Developer Network. (2016). Načteno z Transact-SQL Reference (Database Engine): https://msdn.microsoft.com/en-us/library/bb510741.aspx

Molinaro, A. (2009). *SQL: kuchařka programátora.* Brno: Computer Press.

SAP Help Portal - The central place for SAP documentation. (2016). Načteno z SAP Help Portal - The central place for SAP documentation: http://help.sap.com/

Stack Overflow. (2016). Načteno z Stack Overflow: http://stackoverflow.com/

IV. Seznam obrázků

V. Příloha

1. DPSkripty.sql