Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Návrh ELT modelu pro integraci datových souborů pomocí databázových funkcionalit

Adam Valášek

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Adam Valášek

Systémové inženýrství a informatika
Informatika

Název práce

Návrh ELT modelu pro integraci datových souborů pomocí databázových funkcionalit

Název anglicky

Design of the ELT model for data files integration using database utility

---

Cíle práce

Bakalářská práce je zaměřena na problematiku navržení modelu pro zpracování souborů jako zdroje dat pro stage vrstvu datového skladu. Náplní a účelem této práce je:
- specifikovat možnosti DBMS Oracle ve využívání souborů typu Flat file, csv, zip, xml pro naplnění datového skladu.
- analyzovat existující možnosti databázových metod pro import (export) dat uložených v souborech.
- navrhnout a ověřit odpovídající metadata model pro zpracování těchto souborů včetně odpovídajících externích tabulek (načtení těchto tabulek, přesuny mezi adresáři vstupu, vlastní zpracování a archiv./error.adresáře.

Metodika

Použitá metodika zadané bakalářské práce bude založena na studiu a analýze dostupných informačních zdrojů a případných existujících řešení v dané oblasti. Stěžejními metodami této práce budou metody a techniky relačně databázové technologie a SQL. Navrhované řešení bude zohledňovat identifikované požadavky a očekávání spojená s řešenou záležitostí. Na podkladě syntézy teoretických poznatků a dosažených výsledků budou formulovány závěry této bakalářské práce a následně zobecněny pro další možná použití.

Harmonogram práce:

Vymezení teoretických principů řešené problematiky, literární rešerše – do 5.9.2020: předmět 1. zápočtu z BP,

Zmapování současné situace řešené problematiky a navržení odpovídajícího řešení – do 10. 1. 2021,

Ověření navrženého řešení – do 20.2.20201: předmět 2. zápočtu z BP

Zobecnění navrhovaných záležitostí – do 10.3.2021: předmět 3. zápočtu z BP.

Doporučený rozsah práce

45-55

Klíčová slova

relačně db technologie, datové modelování, metamodel, SQL*Loader, preprocess, externí tabulka, ETL, DWH

Doporučené zdroje informací

Expert Oracle Database Architecture – Thomas Kyte Effective Oracle By Design – Thomas Kyte Oracle one-on-one – Thomas Kyte – Oficialní on-line příručky na stránkách Oracle https://docs.oracle.com/ pro Data Warehousing – příspěvky na internetu od Conora McDonalda

---

Předběžný termín obhajoby

2021/22 LS – PEF

Vedoucí práce

doc. Dr. Ing. Václav Vostrovský

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 23. 11. 2021

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 28. 11. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 29. 11. 2021

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Návrh ELT modelu pro integraci datových souborů pomocí databázových funkcionalit" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 29.11.2021 _____

Poděkování

Rád bych touto cestou poděkoval doc. Ing. Václavu Vostrovskému, Ph.D za odborné vedení, cenné rady a podněty při vypracování této bakalářské práce. Dále bych chtěl poděkovat rodině a kamarádům, kteří mě po celou dobu studia podporovali.

Návrh ELT modelu pro integraci datových souborů pomocí databázových funkcionalit

Abstrakt

První část literární rešerše této bakalářské práce se zaměřuje relační databázi a uvedení čtenáře do problematiky databází. Další část se věnuje relační databázi společnosti Oracle a jejím funkcionalitám, které souvisí s řešenou problematikou a práci s datovými soubory. V další kapitole je definován datový sklad, architektura a využití datových skladů. V závěrečné kapitole jsou obecně popsány ETL nástroje, jejich použití a rozdíl mezi pojmy ETL a ELT.

Praktická část se bude zabývat návrhem a vývojem ELT modelu, který bude využívat Oracle funkcionality a ETL nástroj, popsaný na začátku druhé části. Navržený ELT model bude načítat zdrojové soubory (zdrojová data) do databázových tabulek, které v tomto případě budou představovat vstupní vrstvu datového skladu.

Klíčová slova: relačně db technologie, datové modelování, metamodel, SQL*Loader, export/import (utility exp/imp), datapumpový export/import (utility expdp/impdp), externí tabulky (s využitím možnosti preprocess), ETL, DWH

Design of the ELT model for data files integration using database utility

Abstract

Bachelor thesis is primary focused on using Oracle methods for import (also export) data sources and creating database model for using focused methods for import data sources saved in database server or client in different forms (CSV, XML ...). In the first part of my bachelor's thesis, individuals will use the Oracle methods used and their use will be listed here. The ETL tool Informatica PowerCenter, a brief description of data warehouses, will also be mentioned.

The second part (ie the practical part) will deal with the development of a database model that will produce Oracle methods and ETL tools, described in the first part. The proposed database model will load source files (source data) into database tables, which in this case will represent the phase of the data warehouse stack.

Keywords: relational database technology, data modelling, metamodel, SQL*Loader, export/import (utilities exp/imp), datapump export/import (utilities expdp/impdp), external tabules (with preprocess option), ETL, DWH.

Obsah

Seznam obrázků

Seznam použitých zkratek

SQL	Structure Query Language
PL/SQL	Procedural Language / Structure Query Language
ETL	Extract Transform Load
ELT	Extract Load Transform
DDL	Data Definiton Language

1 Úvod

V dnešní digitální době tvoří data velmi významnou roli. Drtivá většina podniků, či korporátů už funguje digitálně a zpracovává poměrně velké množství dat, s kterými dále, co nejefektivněji nakládá. Data v podnicích jsou často zpracovávány na denní bázi, což v kombinaci s velkými objemy dat představuje komplexní řešení, jak tyto data zpracovávat. Jelikož mohou být zdrojová data dodávány v nejrůznějších podobách (csv, xml, dump apod.), jsou dnes pro integraci těchto dat využívána celá škála nástrojů a metod, které byly pro tuto problematiku vytvořeny. Místo, které je pro určeno pro začleňování a manipulaci dat, se nazývá relační databáze. Dnes na trhu existuje několik firem, které poskytují jejich databázovou platformu (Oracle, Microsoft atd.). V této práci je využita databázová platforma Oracle, což je dnes nejrozšířenější poskytovatel databázových systému.

Bakalářská práce se zabývá problematikou integrace dat ve vstupní vrstvě datového skladu, což je slučování nebo kombinování dat z různých zdrojů a jejich převádění na cenné informace. Integrace dat umožňuje kombinovat data uložená v různých formátech, která pocházejí z různých zdrojů a ukládat je na jednom místě. V konečném důsledku a možná nejdůležitějším cílem integrace dat je generování cenných a užitečných informací, které jsou pro mnohé společnosti prospěšné při řešení problémů a při rozhodování. Díky integraci dat je možné použít informace, které by jinak byly skryté. To může pomoci zlepšit komunikaci mezi jednotlivými odděleními společnosti, ale také poskytnout lepší služby zákazníkům, zlepšit provoz společnosti nebo zefektivnit obchodní rozhodování.

Na základě výběru databázové platformy jsou v této práci rozebrány Oracle funkcionality, které se dnes hojně využívají v podnicích v souvislosti se zmiňovanou problematikou. Tyto Oracle funkcionality slouží k přenosu velkého objemu dat a jsou dnes velmi podstatnou částí v informačních technologií. Práce těchto funkcionalit není pouhé kopírování dat z jednoho souboru do databáze a naopak. Mimo tento úkol musí implementovaná funkcionalita obsahovat také mechanismy umožňujícími transformaci dat do žádané podoby. V mnoha případech totiž zdrojové soubory (data) nejsou uloženy v takové formě, v jaké je potřeba je uložit do vstupní vrstvy datového skladu, nebo na jiné místo.

Rozebrány budou také datové sklady a ETL nástroje, které se dnes využívají pro tvorbu ETL, či ELT procesů. Použití ETL nástrojů pro integraci dat je velice vhodné, jelikož mají velmi úzkou vazbu s cílovými databázemi a aplikačními servery, na kterých se

nacházejí zdrojová data. Díky těmto nástrojům (resp. jejich funkcionalitám) jsou dnes v podnicích tvořeny nejrůznější komplexní řešení (modely), které souvisejí s danou problematikou této bakalářské práce.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem bakalářské práce je návrh ELT modelu, který bude prostřednictvím databázových funkcionalit datově plnit vstupní vrstvu datového skladu zdrojovými datovými soubory. Model má mimo jiné také za cíl zajistit historizaci úspěšně, či neúspěšně zpracovaných datových souborů. Dílčí cíle jsou:

- Vymezení teoretických principů souvisejících s řešenou problematikou.
- Specifikovat Oracle funkcionality pro práci s datovými soubory.
- Představit ETL nástroj pro integraci dat.

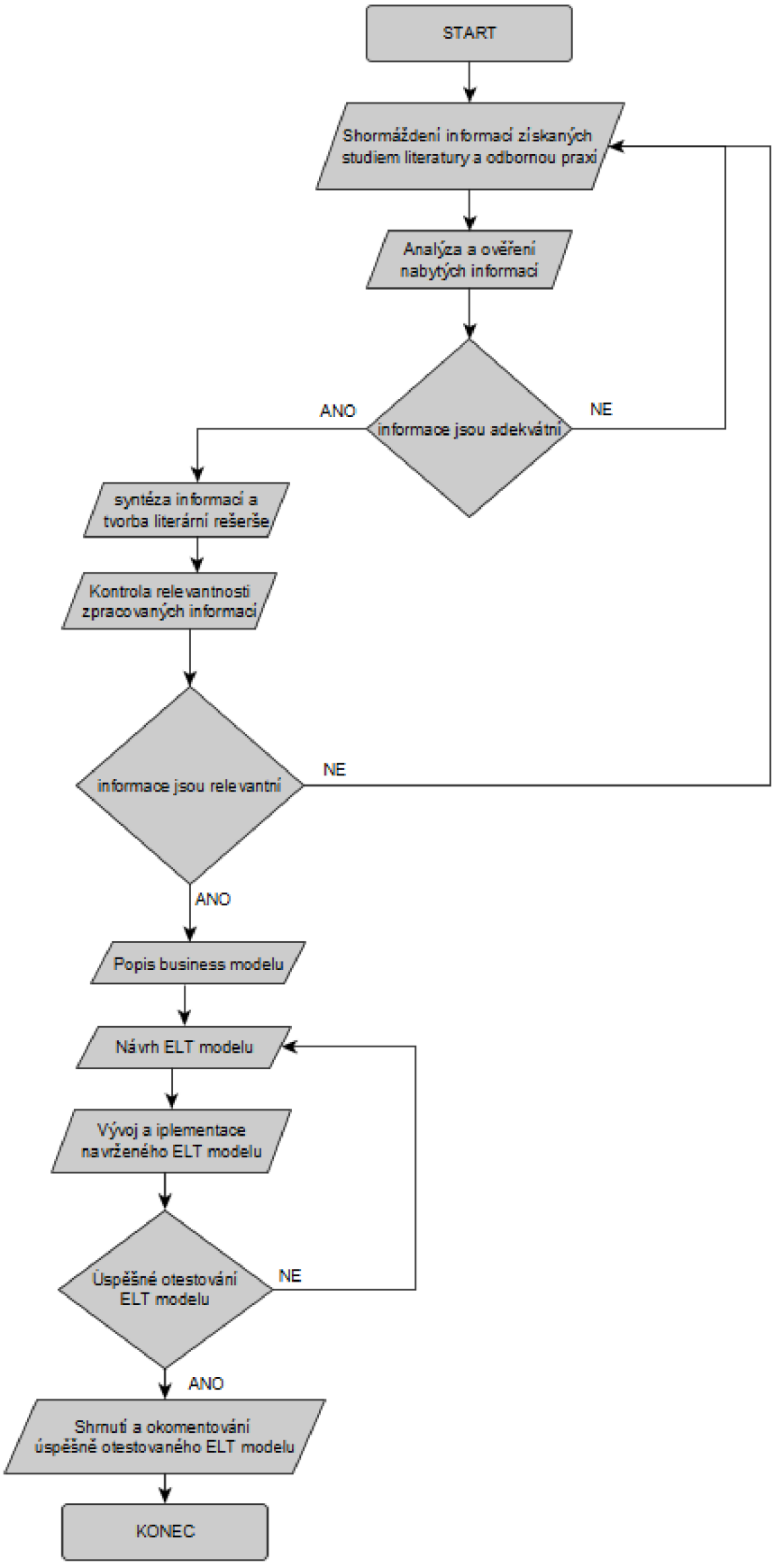
2.2 Metodika

Použitá metodika bakalářské práce je založena na studiu a analýze dostupných informačních zdrojů, odborných publikací a technických dokumentací. Z uvedených zdrojů byly zpracovány nejdůležitější termíny související s problematikou integrace dat ve vstupní vrstvě datového skladu.

Pro tvorbu ELT modelu ve vlastní práci byl vybrán ETL nástroj Informatica PowerCenter. Výběr ETL nástroje byl proveden na základě průzkumu aktuálně nejpoužívanějších nástrojů. Navržený a implementovaný business model je tvořen kombinací vybraného ETL nástroje, vytvořených metadatových tabulek a databázových Oracle funkcionalit, které jsou volány v naprogramovaných PL/SQL procedurách, či databázových skriptech ze zmíněného ETL nástroje.

Hlavní problematikou, kterou se navržený model zabývá, je integrovat zdrojové datové soubory ve vytvořené vstupní vrstvě datového skladu. Pro dosažení tohoto cíle proto bylo nezbytně nutné studium a tvorba jednoduché vstupní vrstvy datového skladu, která vychází z literatury autora Raplha Kimballa. Vstupní vrstva představuje modifikované Oracle databázové HR schéma. Posledním důležitým bodem bylo promyšlení, které Oracle funkcionality budou použity k tvorbě ELT modelu, resp. k integraci dat ve vstupní vrstvě a jakým způsobem bude ELT model zpracovávat a archivovat zdrojové soubory.

Na podkladě syntézy teoretických poznatků a dosažených výsledků budou formulovány závěry této bakalářské práce a následně zobecněny pro další možná použití.



Obrázek 1: Metodický diagram (vlastní)

3 Teoretická východiska

3.1 Relační databáze

Relační databáze je typ databáze, která ukládá a poskytuje přístup k datovým strukturám, které spolu souvisejí. Relační databáze jsou založeny na relačním modelu, intuitivním a přímočarém způsobu reprezentace dat v tabulkách. V relační databázi je každý řádek v tabulce prezentován jako záznam s jedinečným ID (primární klíč). Sloupce tabulky obsahují atributy dat a každý záznam má obvykle hodnotu pro každý atribut, což usnadňuje navázání vztahů mezi datovými body. V širším pojetí spadají do pojmu databáze i možnosti, jak s daty pracovat (ukládat, měnit a mazat). [1] [2]

3.1.1 Systém řízení báze dat

Systém řízení báze dat (dále jen SŘBD) je programová, která řeší operace nad databází. Tento název vznikl přeložením původního anglického termínu DBMS – Data Base Management System. Popis dat je tvořeno databázovým schématem. Databázové schéma popisuje objekty a vztahy mezi nimi. Na SŘBD můžeme nahlížet jako na speciální virtuální nástroj, který zapouzdřuje data. [1] [3]

3.1.1.1 SW produkty poskytující SŘBD

- Oracle Database
- Microsoft SQL Server
- MySQL
- Microsoft Access
- MariaDB
- PostgreSQL
- SQLite

3.1.2 Datová integrita

K zajištění, že data jsou vždy přesná a přístupná, jsou v relační databázi udržovaná určitá pravidla integrity. Pravidlo integrity může například určit, že v tabulce nejsou povoleny duplicitní řádky, aby se eliminoval potenciál pro chybné informace vstupující do databáze. [3]

3.1.3 Kardinalita vztahů mezi entitami

Kardinalita vztahů mezi entitami určuje vazbu mezi dvěma a více tabulek. Je tvořena přes primární a cizí klíč. [2]

- 1:1: Tento vztah je nejméně často používaný. Popisuje například situaci, kdy jedno nakladatelství vydává pouze knihy pouze jednoho autora a zároveň tento autor píše knihy pouze pro jedno nakladatelství.
- 1:N: Tento vztah vyjadřuje, že nakladatelství vydává knihy více autorů, ale jeden autor produkuje pouze pro jedno nakladatelství.
- M:N: Poslední vztah vyjadřuje, že jedno nakladatelství může vydávat více autorů a zároveň autor může vydávat ve více nakladatelstvích.

3.1.4 Úrovně pohledu na data

3.1.4.1 Konceptuální úroveň

Tento pohled popisuje objekty reálného světa, vztahy mezi nimi a funkce, pomocí kterých se tyto vztahy realizují. [2]

- **Entita**: Objekt reálného světa. Příkladem entity je stůl, kniha, autor.
- **Vztah**: Vazba mezi dvěma nebo více entitami.
- **Atribut**: Tato hodnota určuje nějakou podstatnou vlastnost entity nebo vztahu, např. jméno a příjmení autora, datum vydání knihy. Jedná se o sloupec v databázové tabulce.

3.1.4.2 Logická úroveň

Pro popis dat na logické úrovni se v relačních databázích používá datový model. Datový model zobrazuje tabulky včetně všech jejich sloupců. Hlavním účelem datového modelu je vyznačení primárního klíče v tabulkách a dále i cizí klíče, jako odkaz na primární klíč v jiné tabulce. Tato vazba je většinou vyznačena jako čára spojující primární a cizí klíč ve dvou tabulkách. Součástí datového modelu mohou být i popisy integritních omezení tabulek a sloupců. [1] [3]

3.1.4.3 Implementační úroveň

Na implementační úrovni je nejprve vybrán konkrétní databázový systém, ve kterém bude vytvořen datový model. Po jeho výběru jsou využívány i různé nestandardní (specifické) funkce zvoleného prostředí. Jejich použití by se mělo však důkladně zvážit, zejména kvůli možnému pozdějšímu přechodu na jiný databázový systém. Z hlediska jazyka SQL je nutné na této úrovni vzít v úvahu i možné dílčí odlišnosti v příkazech, zejména v příkazech pro DDL. [3]

3.2 Oracle

Databáze Oracle je soubor dat, která je považována za jednotku. Oracle je systém řízení báze dat, moderní multiplatformní databázový systém s velice pokročilými možnostmi zpracování dat, vysokým výkonem a snadnou škálovatelností. Databázový systém Oracle je vyvíjen firmou Oracle Corporation. Oracle Corporation je největší společností na světě, která dodává podnikový software firmám a organizacím všech velikostí. Tato společnost, jejíž roční obrat činí 10,2 miliardy USD, nabízí kromě podnikových aplikací a nástrojů na jejich vývoj také databázi, aplikační server a nástroje pro podnikovou spolupráci. [3] [4] [5]

Relační databáze Oracle navazuje na obecnou definici relačního modelu databázových systémů, kterou definoval Dr. Edgar F. Codd (soubor dvanácti pravidel, která musí databáze dodržovat, aby mohla být považována za skutečnou relační databázi).

V době psaní této bakalářské práce byla aktuální verze Oracle 19c. [1] [5]

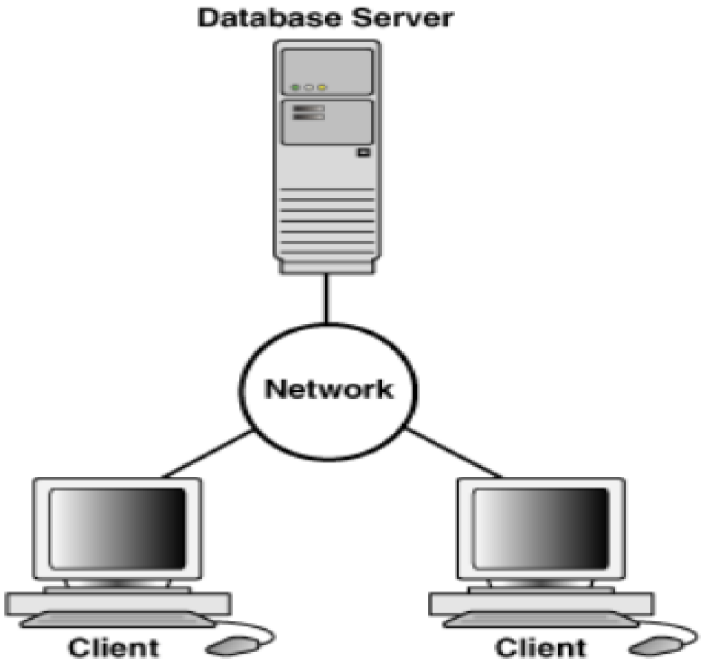
3.2.1 Databázový server Oracle

Databázový server je klíčem k řešení problémů správy informací. Server obecně spolehlivě spravuje velké množství dat ve víceuživatelském prostředí, takže k stejným datům může souběžně přistupovat mnoho uživatelů. To vše je dosaženo při vysokém výkonu. Databázový server také zabraňuje neoprávněnému přístupu a poskytuje efektivní řešení pro obnovu po selhání. [6] [7]

3.2.2 Připojení na databázový server z klienta

Na databázový server se připojíme pomocí nainstalovaného klienta. Důležité je vědět, že klient na databázový server vidí, ale databázový server na klienta nevidí. Klient

pomocí souboru tnsnames.ora a protokolu TCP vidí na databázový server. Databázový server poskytuje IP adresu, díky které se klient na databázi připojí (IP adresa databázového serveru je definovaná v souboru tnsnames.ora). Tnsnames.ora je konfigurační soubor, kde jsou nadefinované vytvořené databáze. Celý výraz TNS obsahuje parametry potřebné pro připojení k databázi. Na databázovém serveru běží služba Listener, která zprostředkuje žádost klienta o připojení na databázový server a tím i na databázi (klient posílá na databázový server IP adresu a název databáze – TNS). [4] [7]



Obrázek 2: CLIENT/SERVER schéma [7]

3.3 Oracle funkcionality sloužící pro práci s datovými soubory

Společnost Oracle poskytuje několik funkcionalit, s jejichž pomocí lze importovat, exportovat, či manipulovat s datovými soubory. Tyto funkcionalit se obecně rozdělují do dvou skupin. Do první skupiny patří ty funkcionality, které je možné spustit z databázového serveru, anebo z klienta (vzdáleně). Do druhé skupiny patří funkcionality, které je možné spustit pouze z databázového serveru.

K rozhodnutí, kterou funkcionalitu je vhodné použít, je zásadní „technické“ zadání, resp. situace (např. Má se importovat 40 tisíc záznamů, anebo 40 milionů záznamů? Mají se importovat pouze záznamy, anebo i databázová tabulka jako objekt?). [3] [4]

3.3.1 Oracle funkcionality spustitelné z databázového serveru a klienta

3.3.1.1 SQL*Loader

SQL*Loader je součástí instalace databázového klienta. Tato funkcionalita umožňuje načíst data z externího souboru do databázové tabulky a dokáže analyzovat mnoho formátů souborů s oddělovači, jako je CSV. Ovládá se přes kontrolní soubor, v kterém se specifikuje struktura dat, které jsou určeny k importu do cílové tabulky v databázi. Existují tři možnosti, jak importovat data pomocí funkcionality SQL*Loader.

První z možností je importovat data konvenčně (Conventional Path Loads), tímto způsobem jsou data importována příkazem INSERT. Konvenční import se využívá pro menší objem dat (je pomalejší). Další možností je přímý import (Direct Path Loads), který je založen na příkazu INSERT APPEND a využívá se k velkému objemu dat (je mnohem rychlejší než konvenční import). Třetí možnost je importovat data pomocí externí tabulky (External Tables). Funkcionalita External Tables bude podrobněji popsaná v kapitole 3.3.3.3. Pro SQL*Loader jsou zásadní dva soubory, které konfigurují import dat do databáze. Soubory se nazývají Control file a Input file. Při běhu importu také průběžně vznikají soubory LogFile a BadFile. [8] [9]

3.3.1.2 Oracle EXP/IMP

Tato funkcionalita je jedna z dalších alternativ, jak přesouvat data. Poskytuje jednoduchý způsob přenosu datových souborů mezi databázemi Oracle, i když jsou umístěny na platformách s různými konfiguracemi hardwaru a softwaru.

Nejen data je možné exportovat, či importovat. Tato funkcionalita umožňuje exportovat, či importovat databázové objekty (tabulky, balíky, triggery atd.), databázové schéma, databázové oprávnění, anebo rovnou celou databázi. Funkcionalita EXP/IMP primárně pracuje s DUMP souborem, který bude detailněji popsán v kapitole 3.3.3.2.1 [10]

3.3.2 Objekt Oracle Directory

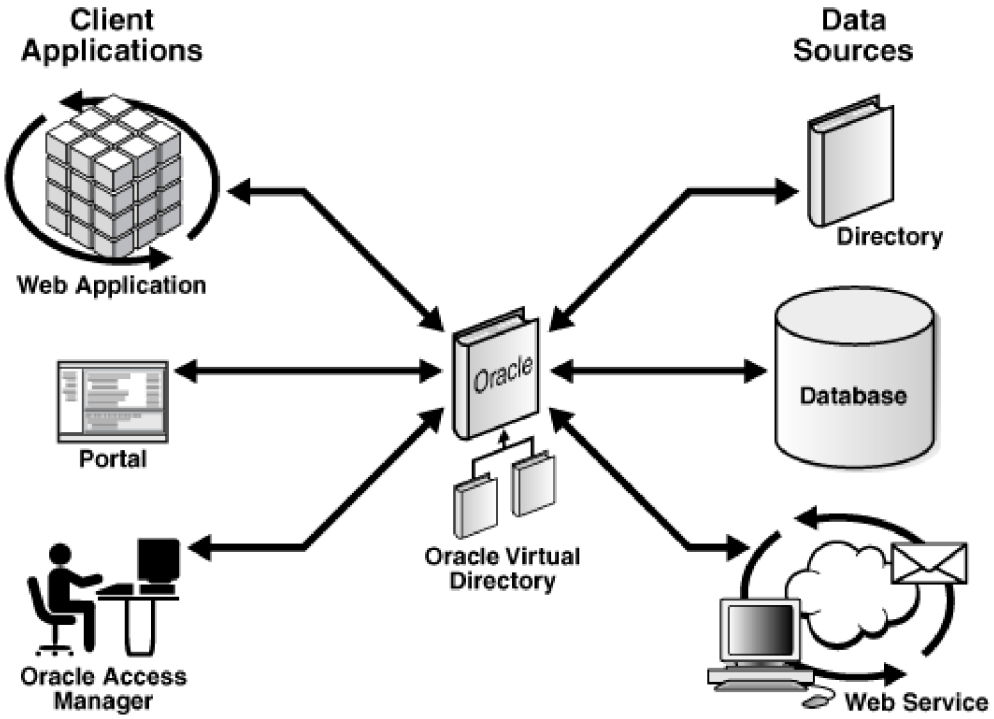
Oracle Directory je databázový objekt, který se odkazuje do fyzického adresáře operačního systému na stroji, kde je umístěn databázový server. Pomocí Oracle Directory můžeme číst, zapisovat či spouštět soubory.

Jedná se o objekt, který patří pod databázové schéma SYS (pod žádné jiné databázové schéma patřit nemůže). V praxi Oracle Directory vytváří na žádost uživatelů

databázový administrátor, který pak uživatelům uděluje systémová práva (např. přidělení práva pro přístup do Oracle Directory jiným uživatelům). Tento databázový objekt je pro používání Oracle funkcionalit, které jsou spustitelné ze serveru nezbytně nutný, protože na tento objekt, resp. adresář, se tyto funkcionality odkazují. [11] [12]

Práva pro práci s Oracle Directory: [11]

- **READ** – Umožňuje číst soubory.
- **WRITE** – Umožnuje zapisovat do uložených souborů.
- **EXECUTE** – Umožňuje spustit uložené soubory. Toto právo se často používá pro spuštění dávkových souborů.



Obrázek 3: Oracle Directory schéma [11]

3.3.3 Oracle funkcionality spustitelné pouze z databázového serveru

3.3.3.1 UTL_FILE

UTL_FILE je vestavěný balík PL/SQL, díky kterému lze na databázovém serveru vytvářet do fyzických adresářů datové soubory různých typů, anebo z těchto souborů načítat data do databázových tabulek. Fyzické adresáře na databázovém serveru se ovšem musí odkazovat na své vytvořené Oracle Directories.

Tento balík se skládá z několika vytvořených sub-programů (procedury i funkce), které slouží pro manipulaci s datovými soubory. Mimo jiné je v tomto balíku deklarovány typ FILE_TYPE, s jehož pomocí se uživatel může odkazovat na vytvořené funkce. UTL_FILE obsahuje také řadu výjimek a konstant. [13]

Objekty poskytující UTL_FILE: [13] [14]

- **FILE_TYPE**: Tento objekt se skládá ze tří atributů, s kterými UTL_FILE balík privátně pracuje. Tyto atributy slouží k řízení a identifikaci datových souborů, s kterými uživatel momentálně pracuje. FILE_TYPE je využíván jako parametr při volání sub-programů v UTL_FILE balíku.
- **Funkce FOPEN**: Tato funkce se v PL/SQL využívá k otevření adresáře, nebo vytvoření datového souboru. Zpravidla se ve výkonné oblasti PL/SQL kódu k vytvořené proměnné typu FILE_TYPE přiřazuje volání funkce FOPEN.
- **Procedura PUT_LINE**: Procedura PUT_LINE provádí zápis do souboru, narozdíl od procedury PUT, PUT_LINE zalomuje řádek.
- **Procedura GET_LINE**: Procedura GET_LINE slouží k čtení souboru, resp. textových řetězců. Tato procedura ukončuje čtení textového řetězce do konce jeho délky, anebo je možné omezit čtení řetězce na určitou délku.
- **Funkce FCLOSE**: Voláním této procedury se ukončí manipulace se zdrojovým souborem.
- **Procedura FCOPY**: Procedura FCOPY umožňuje zapisovat obsah jednoho souboru do druhého.
- **Procedura FRENAME**: Voláním této procedury je možné přesouvat datové soubory mezi adresáři (na adresáře se samozřejmě musí odkazovat patřičný Oracle Directory).
- **Procedura FREMOVE**: Tato procedura odstraňuje datové soubory.
- **Procedura FGETATTR**: Tato procedura dokáže vyhledat a vrátit atributy hledaného souboru v daném adresáři.

3.3.3.2 Oracle Data-Pump (EXPDP/IMPDP)

Oracle Data-pump je novější (od verze Oracle 10.g), rychlejší a flexibilnější funkcionalita sloužící nejen k přenosu dat, ale i přenosu databázových objektů (stejně jako

u funkcionality EXP/IMP). Tato funkcionalita kromě základních funkcí pro import a export poskytuje také API pro PL/SQL (DMBS_DATAPUMP) a podporu externích tabulek.

Oracle Data-pump, stejně jako EXP/IMP pracuje s DUMP souborem. Pro použití této funkcionality, je potřeba mít DUMP soubor uložen v Oracle Directory. Pokud by se zdrojový DUMP soubor nenacházel v Oracle Directory, tato funkcionalita by na něj jednoduše neviděla. [15]

3.3.3.2.1 DUMP soubor

DUMP soubor je binární soubor, v kterém je definovaná struktura (v případě databázové tabulky i záznamy) databázových objektů. DUMP soubory jsou v praxi používány především pro tvorbu záloh, či nasazování nových verzí databázových objektů. [15]

Rozdíly mezi EXPDP/IMPDP a EXP/IMP: [16] [17]

- EXPDP/IMPDP má možnost paralelního běhu, funkcionalita EXP/IMP nikoliv.
- EXPDP/IMPDP má přístup k zdrojovým souborům pouze na serveru, zatímco funkcionalita EXP/IMP má přístup na serveru i v klientovi.
- U EXPDP/IMPDP je možné se odpojit, nechat danou úlohu běžet a poté se kdykoliv připojit a kontrolovat průběh úlohy.
- Výkon EXPDP/IMPDP je vyšší oproti EXP/IMP.

3.3.3.3 Externí tabulky

Tato funkcionalita je tzv. doplňkem funkcionalit SQL*Loader a Oracle_Datapump. Externí tabulky je vhodné používat, pokud import provádíme opakovaně. Hlavní podstatou této funkcionality je, že se díváme na zdrojový soubor jako na databázovou tabulku (resp. Databázový objekt). Zdrojové soubory mohou být uloženy v různých formách (CSV, XML, JSON apod.). Po vytvoření externí tabulky můžeme ze zdrojového souboru mazat, či zapisovat. Za běhu externích tabulek se do Oracle_directory vytváří průběhový soubor (Log file).

Externí tabulky mají ovšem svá omezení. Nad externí tabulkou není možné provádět DML příkazy (tzn. INSERT, UPDATE, DELETE), anebo není možné vytvářet indexy sloužící k optimalizaci SQL dotazů. Je možné vytvářet nad externími tabulky tzv. constrainty, ale jejich deklarace se však liší od běžné deklarace contraintů nad klasickými

databázovými tabulkami. Externí tabulky nabízejí funkcionalitu preprocesor. Tato funkcionalita umožňuje importovat zdrojové soubory, které jsou zipované. [3] [18]

3.3.3.3.1 Typy externích tabulek: [18]

- Oracle_Loader: Externí tabulka vytvořena pod tímto typem umožňuje načítat data ze zdrojového souboru. Při tvorbě externí tabulky je tento typ nastaven jako výchozí.
- Oracle_Datapump: Tento typ se využívá pro větší objem dat. Umožňuje načítat data ze zdrojového souboru, ale i zapisovat do něj. Pokud se provádí export dat (zápis do souboru), externí tabulka vytvoří uživatelem definovaný binární soubor s kocovkou soubor.dat.

3.3.3.3.2 Externí tabulka s preprocesorem

Externí tabulka s preprocesorem se dokáže odkazovat na zazipovaný soubor. Tato funkcionalita funguje tak, že po každém jejím spuštění se nejdříve spustí skript, který rozbalí zazipovaný zdrojový soubor, a poté se provede patřičný import (Nejčastěji příkazem INSERT). [19]

3.3.3.3.3 Externí tabulka odkazující se na XML soubor

Externí tabulky umožňují odkazovat se i na jiné datové struktury, jednou z nich je struktura XML. V takovém případě se musí externí tabulka vytvořit s datovým typem CLOB, který svou délkou poskytuje dostatečnou velikost na to, aby pojal celý XML soubor. [20]

Import se v takovém případě provádí jako INSERT /* APPEND */ SELECT, kdy SQL dotaz je vytvořený tak, aby dokázal z uloženého XML souboru v typu CLOB roztřídit jednotlivé atributy. [18]

3.4 Datové sklady a ETL nástroje

3.4.1 Datové sklady

V datových skladech probíhají procesy pro sběr a správu dat z různých zdrojů, obvykle je využíván k připojení a analýze obchodních dat z heterogenních zdrojů. Účelem datových skladů je poskytnutí smysluplných obchodních náhledů pro koncové uživatele, či systémy. Datový sklad je jádrem BI (Business Inteligence) systémů. [21]

3.4.1.1 Business Inteligence (BI)

Business inteligence můžeme chápat jako ucelený a efektivní přístup k práci s firemními daty, který má vliv na správnost strategických rozhodnutí, a tím i na obchodní úspěch společnosti. V současném vysoce konkurenčním prostředí představuje informovanost jednu z hlavních konkurenčních výhod. Tato výhoda spočívá ve schopnosti efektivně využít data nashromážděná ve firmách k tvorbě informací a znalostí, na základě, kterých můžeme reagovat na rychle se měnící požadavky trhu a našich zákazníků. [26]

Pojem BI představuje vizualizaci dat, která využívá znalosti, technologie, aplikace, analýzy a další postupy za účelem sběru, integrace a správné interpretace a prezentace obchodních informací. [21]

3.4.1.2 Architektura datového skladu

Datový sklad má vždy svou specifickou architekturu. Architektura je tvořena třemi základními vrstvy: [21] [22]

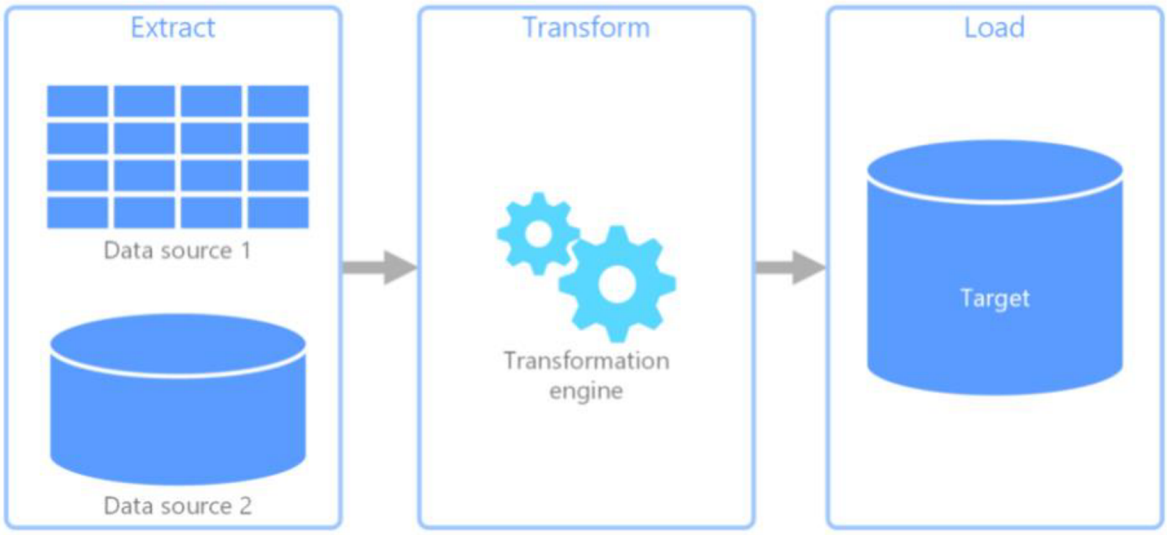
- **Vstupní vrstva** (Staging area): Tato vrstva slouží pro dočasné sjednocení zdrojových dat. V případě ETL se v této vrstvě se data nijak netransformují, jsou zde pouze vytvořeny technické atributy, které například určují datum, ke kterému se data vážou. Na základě business logiky se po nějakém určitém čase část dat maže.
- **Transformační vrstva**: Zde probíhají transformace a historizace sjednocených dat na základě implementované business logiky.
- **Výstupní vrstva** (Data Mart): Tato vrstva slouží jako koncové úložiště transformovaných dat. Tuto vrstvu dále využívají koncové systémy, či uživatelé.

3.4.2 ETL nástroje

Běžný problém, který organizace čelí, je shromažďování dat z více zdrojů, ve více formátech a jejich přesun do jednoho nebo více úložišť dat. Cíl nesmí být stejného typu jako zdroj dat a často se jedná o jiný formát, nebo musí být data před jejich načtením do konečného umístění vyčištěna nebo vyčištěna. [25]

ETL (Extraction, Transformation and Loading) nástroje byly vytvořeny pro přesun velkého objemu dat. Mají za úkol tři věci, shromažďování dat z různých zdrojů,

transformaci dat dle business logiky a následné načtení do cílových úložišť (z pravidla do Datových skladů). Datové transformace v ETL jsou realizovány ve specializovaném stroji. Při transformacích se často používají tzv. dočasné tabulky (vstupní vrstva datového skladu) k dočasnému uchování a začlenění dat. [22] [24]



Obrázek 4: ETL schéma [25]

3.4.2.1 Výhody ETL nástrojů: [25]

- intuitivní vývoj – Grafické prostředí v ETL nástrojích usnadňuje vývoj (změny).
- podpora metadat – ETL nástroje rozeznávají popisné informace o zdrojových a cílových objektech.
- vysoký výkon – Využívají více vláknovou architekturu a využívají nativní přístup ke zdrojovým i cílovým systémům. Jsou schopny pracovat na úrovni miliónů záznamů.

Rozdělení ETL běhů: [22]

- Replikační (bez transformace) běhy – Data jsou načítány formou 1: 1. Obvykle jsou v tomto běhu přidány technické sloupce.
- Transformační běhy – Transformační běhy se standartně využívají pro reportovací účely. Používá se zde aplikovaná business logika, vycházející z business požadavků.

- Transformační běhy s DQ – Transformační běhy zajišťují datovou kvalitu. Často se stává, že v zdrojových datech jsou nevyhovující znaky v řetězcích, tento běh dokáže filtrovat tyto znaky a nahrazovat je požadovanými znaky. To má ovšem vliv na výkon, který se snižuje.

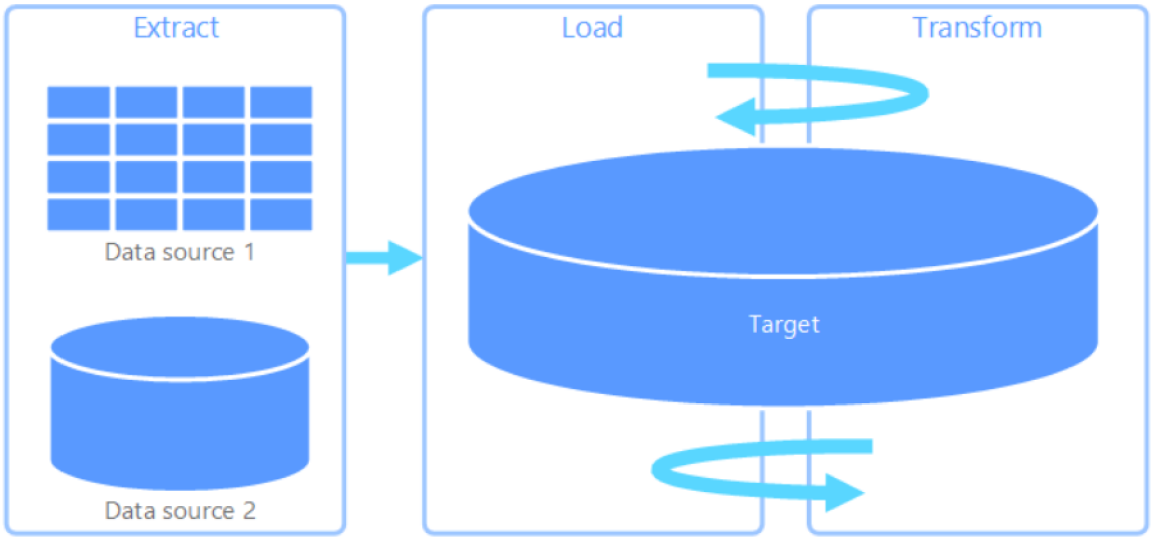
3.4.2.2 ELT

ELT je proces integrace dat pro přenos ze zdrojového serveru do datového skladu na cílovém serveru a na následnou přípravu informací pro následné použití. [22]

Prvním krokem ELT je extrahování dat. Extrahování dat je proces identifikace a čtení dat z jednoho nebo více zdrojových systémů, kterými mohou být databáze, soubory, archivy, ERP, CRM nebo jakýkoli jiný životaschopný zdroj užitečných dat.

Druhým krokem pro ELT je načtení extrahovaných dat. Načítání je proces přidávání extrahovaných dat do cílové databáze.

Třetím krokem je transformace dat. Transformace dat je proces převodu dat z jejich zdrojového formátu do formátu potřebného pro analýzu. Transformace je obvykle založena na pravidlech, která definují, jak mají být data převedena pro použití a analýzu v cílovém datovém úložišti. Ačkoli transformace dat může mít mnoho různých forem, často zahrnuje konverzi kódovaných dat na použitelná data pomocí kódu a vyhledávacích tabulek. [25]



Obrázek 5: ELT schéma [25]

3.4.2.3 Rozdíl mezi ETL a ELT

Rozdíly mezi ELT a tradičním procesem ETL jsou významnější než pouhé prohození písmen L a T. Největším určujícím faktorem je jak, kdy a kde se provádějí transformace dat. S ETL jsou netransformovaná data nedostupná v datovém skladu, protože jsou před načtením transformována. Pomocí ELT se netransformovaná data načtou do vstupní vrstvy datového skladu a na uložených datech dojde k transformacím. [24] [25]

ETL a ELT transformace zahrnují: [22]

- Nahrazení vstupních kódů hodnotami.
- Agregace číselných součtů.
- Aplikace matematických funkcí.
- Konverze datových typů.
- Úprava textových řetězců.
- Kombinování dat z různých tabulek a databází.

4 Vlastní práce

Praktická část této bakalářské práce se zabývá návrhem a tvorbou funkčního ELT modelu, který prostřednictvím ETL nástroje Informatica PowerCenter přesouvá zdrojové soubory do logicky uspořádaných adresářů na databázovém serveru, z kterých vykonává datové plnění vstupní vrstvy datového skladu pomocí vhodných Oracle funkcionalit a databázových procedur. Na základě úspěšného, či neúspěšného datového plnění ze zdrojového datového souboru je patřičný soubor dále historizován do archivačního, či chybového adresáře na databázovém serveru.

4.1 Popis ETL nástroje Informatica PowerCenter

Tento nástroj je jedním z nejpoužívanějších ve svém odvětví. Informatica PowerCenter je nástroj pro extrakci, transformaci a načítání (ETL) zdrojových dat, který se používá při budování datových skladů.
Tento nástroj se skládá ze čtyř hlavních prostředí: [23]

- **Designer** – V tomto prostředí se vytváří mapování, který může obsahovat jednu, či více pipeline. Mapování obsahuje zdrojový soubor (source), transformace a na konci zdrojovou tabulku.
- **Workflow manager** – V tomto prostředí se tvoří workflow. Workflow je kompletováno z vytvořených mapování do logických na sebe navazujících řetězců či událostí.
- **Monitor manager** – Toto prostředí poskytuje přístup k aktuálním a historickým informacím o průběhu workflows – prohlížeč logů, jak byly naplněny proměnné a s jakými parametry byly wf volány. Může sloužit i k monitorování výkonu jednotlivých workflows či dokonce transformací.
- **Repository manager** – je určen pro správu složek a administraci přístupů administraci metadat.

4.1.1 Použité komponenty ETL nástroje v praktické části

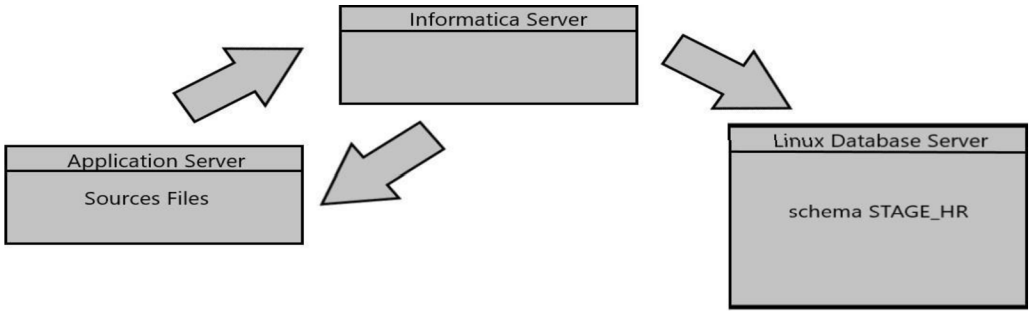
Následuje popis komponent, které byly využity pro tvorbu ELT modelu ve vybraném ETL nástroji Informatica PowerCenter. Níže popsané komponenty realizují celý běh ELT modelu. [23]

- **WorkFlow** v ETL nástrojích představují komplexní řešení, které na základě buiness logiky spouští jednotlivé Session a Tasky. ETL WorkFlow je řízeno hlavně prostřednictvím metadata tabulek, které slouží k uložení technických a business metadat (hodnot), které jsou dynamicky generovány do parametrových souborů.
- **Mapping** (Session) představuje základních stavební blok ETL Workflow, při kterém vznikají ETL tranformace. Je tvořen z jednoho, či více zdrojů dat, transformací a cílových úložišť. Mapování je sestaveno z tzv. Pipeline (mohou jich být více), která určuje tok dat (DataFlow). V mapování se také vytvářejí parametry a proměnné, které se do mapovacích transformací přiřazují. Prostřednictvím jedné mapovací transformace (Stored procedure trasformation), se mohou také volat databázové procedury, či funkce a dosazovat do nich parametry z vytvořených mapovacích proměnných.
- **Assignment task** slouží k přiřazení hodnot do WorkFlow proměnných, které jsou dále posílány do proměnných Sessions a Tasků.
- **Command task** slouží k volání PMCMD Utility, v kterém je možné volat příkazy a skripty z příkazového řádku.

4.2 Popis business modelu

Business model této bakalářské práce má za cíl datově plnit (integrovat) vstupní vrstvu, která se nachází ve vytvořeném Oracle schématu STAGE_HR (vstupní vrstva) na naistalovaném linuxovém (databázovém) serveru v prostředí Oracle VirtualBox. Sady zdrojových souborů jsou dodávány od externích zdrojů na aplikační server ve formátech CSV a XML (některé soubory jsou dodávány zazipované) a jsou rozděleny do svých patřičných zdrojových adresářů.

Připravené zdrojové sady souborů na aplikačním serveru zpracovává ETL nástroj Informatica (resp. vyrvořené WorkFlow's), který zdrojové soubory přesouvá na databázový server do logicky uspořádaných adresářů a mezi tím vykonává operace (Session's a Task's), kterými spouští ETL nástroj funkcionality Oracle používané k plnění databázového schématu STAGE_HR. Tyto ETL funkcionality budou dále popsány níže.



Obrázek 6: High-Level schéma modelu (vlastní)

4.2.1 Použité SW nástroje

- Oracle VirtualBox 6.0.16
- Oracle Database 19c Enterprise Edition Release 19.3 (64-bit)
- Informatica PowerCenter 9.6.1
- PL/SQL Developer 13

4.2.2 Struktura zdrojových souborů

Zdrojové soubory jsou vytvořeny z obsahu původních databázových tabulek ve vzorovém Oracle schématu HR, ke kterým byly vytvořeny a naplněny další tři tabulky, které byly vytvořeny za účelem širšího využití Oracle funkcionalit. Databázové schéma HR bylo vhodné použít, protože je určené pro testovací a studentské použití. Názvy zdrojových tabulek budou uvedeny níže.

4.3 Vývoj a implementace business modelu

4.3.1 Aplikační server, externí zdroje, zdrojové sady souborů

Aplikační server je v této práci reprezentován operačním systémem Windows, na který dodávají externí skupiny zdrojové soubory. Externích zdrojových skupin je celkem tři, to znamená, že tento business model bude čerpat ze tří zdrojů. Pro každou externí skupinu, je na aplikačním serveru separátně vytvořen zdrojový adresář, do kterého externí skupiny dodávají zdrojové soubory určené ke zpracování a importu.

4.3.1.1 Externí skupiny a sady souborů

- **Regional Department**: Dodává sadu čtyř zdrojových souborů ve formátu CSV. Zdrojové soubory vycházejí z obsahu původních databázových tabulek hr.countries, hr.departments, hr.locations a hr.regions. Konvence názvů

dodávaných souborů je název databázové tabulky a vkládají se do adresáře C:\Sources_files\UTLSRC. Pro tento externí zdroj byla vybrána Oracle funkcionalita UTL_FILE.

- **HR Department**: Dodává sadu čtyř zazipovaných zdrojových souborů ve formátech CSV a XML. Zdrojové soubory CSV vycházejí z původní databázové tabulky hr.employees a vytvořené databázové tabulky hr.emp_months_salary. Zdrojové soubory XML vycházejí z původních databázových tabulek hr.jobs, hr.job_history. Konvence názvů dodávaných souborů je název databázové tabulky a vkládají se do adresáře C:\Sources_files\EXTSRC. Pro tento externí zdroj byla vybrána Oracle funkcionalita Externí tabulka s preprocesorem.
- **Commercial Department**: Dodává sadu dvou zdrojových souborů ve formě DMP. Zdrojové soubory vycházejí z vytvořených databázových tabulek hr.clients a hr.emp_cli. Konvence názvů dodávaných souborů je imp_cli.dmp a imp_cli_emp.dmp a vkládají se do adresáře C:\Sources_files\IMPDPSRC. Zdrojové DMP soubory vznikly datapumpových exportem. Pro tento externí zdroj byla vybrána Oracle funkcionalita Data-pump Import.

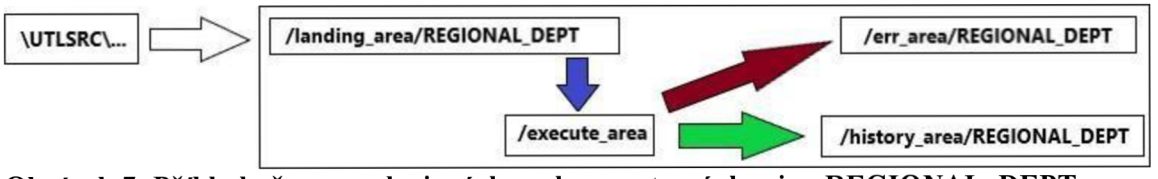
4.3.1.2 Databázový server, zpracování a umístění zdrojových sad souborů

Na databázovém serveru se nachází adresáře landing_area, execute_area, history_area, err_area. Adresáře landing_area, history_area a err_area obsahují ještě tři podadresáře regional_dept, hr_dept a commercial_dept, které jsou logicky přiřazeny zdrojovým souborovým sadám. Mezi adresáři se hierarchicky přesouvají a ukládají zdrojové sady souborů. Pod adresářem execute_area a všemi sub-adresáři byly vytvořeny Oracle Directories, aby bylo možné spouštět Oracle funkcionality.

4.3.1.3 Význam adresářů na databázovém serveru

- **Landing_area**: Do tohoto adresáře (resp. sub-adresářů) se připravují zdrojové sady souborů, které jsou přesunuty z adresářů z aplikačního serveru.
- **Execute_area**: (tzv. výkonný adresář) Zde se přesouvají jednotlivě sady souborů z adresáře Landing_area za účelem provedení importu souborů do databázových tabulek vstupní vrstvy (STAGE_HR).

- **History_area**: Historizační adresář, do kterého se přesouvají jednotlivě soubory z Execute_area v případě, že import dat do schématu STAGE_HR proběhl bez chyb. Soubory jsou zde historizovány s názvovou konvencí "YYMMDD_nazev_souboru.csv".
- **Err_area**: Chybový adresář, do kterého se přesouvají jednotlivě soubory z Execute_area v případě, že import dat do schématu STAGE_HR skončil chybně. Soubory jsou zde historizovány s názvovou konvencí "YYMMDD_nazev_souboru.csv".



Obrázek 7: Příklad přesunu zdrojových souboru externí skupiny REGIONAL_DEPT (vlastní)

4.3.2 Popis tabulek a vstupních tabulek v databázovém schématu HR a STAGE_HR

Databázové schéma HR obsahuje deset tabulek, které mají mezi sebou relační vazby. Datový model HR najdete v příloze 3.

4.3.2.1 Popis HR (zdrojových) tabulek

- **REGIONS**: Záznamy o regionech.
- **COUNTRIES**: Záznamy o státech.
- **LOCATIONS**: Záznamy o lokacích.
- **DEPARTMENTS**: Záznamy jednotlivých oddělení na různých lokací.
- **EMPLOYEES**: Záznamy všech zaměstnanců.
- **JOBS**: Záznamy pracovních pozic
- **JOB_HISTORY**: Záznamy minulých zaměstnanců a jejich pracovních pozic
- **EMP_MONTH_SALARY**: Záznamy o vyplacených výplatách a bonusů.
- **CLIENTS**: Záznamy o klientech.
- **EMP_CLIENTS**: Záznamy o klientech přiřazených určité kategorii zaměstnancům.

4.3.2.2 Popis vstupních (cílových) tabulek ve schématu STAGE_HR (vstupní vrstva)

Struktury vstupních tabulek vycházejí z HR tabulek, byly ovšem rozšířeny o 4 technické atributy: [22]

- stg_id: Slouží jako jedinečný identifikátor záznamu, nad kterým je vytvořen primární klíč. Je generován sekvenčně funkcionalitou Identity sloupce.
- int_date: Do tohoto technického atributu se ukládá datum, k jakému dni se záznam váže. Pokud by se do tohoto atributu vkládala hodnota NULL, tak se automaticky vyplní aktuální datum.
- source: Do tohoto atributu se vkládá hodnota, která se odkazuje na zdrojovou skupinu (např. HR_DEPT).
- created_date: Určuje, kdy byl záznam ve vstupní tabulce vytvořen.

Databázové tabulky byly dále rozšířeny funkcionalitou Partition s intervalem na jeden den, která sloužící k uchování záznamů za určitou periodu dní. Každým importem dat do vstupní tabulky vznikne nová Partition vázaná na sloupec int_date, která spojí importované záznamy do tzv. celku. Mimo jiné může zajistit optimalizaci při dotazování se nad tabulkou. [27]

4.3.2.3 Databázové tabulky IMPDP_CLIENTS a IMPDP_EMP_CLIENTS

Tabulky impd_clients a impd_emp_clients jsou využívány jako dočasné (temporary) tabulky. Tyto tabulky se pomocí functionality Oracle Data-pump import naplní a používají jako zdroj dat pro stage tabulky clients a emp_clients v proceduře proc_LOAD_IMPDP_DATA. Procedura proc_LOAD_IMPDP_DATA bude popsána níže.

4.3.3 Použité Oracle funkcionality a objekty pro databázový server

4.3.3.1 UTL_FILE

Funkcionalita UTL_FILE se volá v databázové proceduře proc_STAGE_imp_UTLF, která bude popsána níže.

4.3.3.2 Externí tabulky s preprocesorem

Funkcionalita se volá v databázové proceduře proc_LOAD_EXT_DATA, která bude popsána níže. Aby bylo možné tuto funkcionalitu vykonávat, je nutné mít pro každý zdrojový soubor vytvořenou Externí tabulku. Jelikož je sada zdrojových souborů dodávána zazipovaná, je nutné, aby Externí tabulka byla vytvořena s preprocesorem. Funkce preprocesoru je v tomto případě spustit batch příkaz GUNZIP.KSH, který odzipuje zdrojové soubory. GUNZIP.KSH se nachází v Oracle directory EXEC_AREA.
Z Externích tabulek pro zdrojové XML soubory bylo nutné vytvořit SQL dotazy, které byly sestaveny tak, aby se jejich výstup podobal výstupu z klasické databázové tabulky. Z těchto dotazů bylo nutné vytvořit databázové pohledy (VIEWS), které byly použity v proceduře proc_LOAD_EXT_DATA jako zdroj dat pro tzv. příkaz INSERT INTO .... SELECT FROM.

4.3.3.2.1 Přiřazení CSV souborům k Externím tabulkám

- **Employees.csv** - zip_ext_employees
- **Emp_months_sal.csv** - zip_ext_emp_monthly_salary

4.3.3.2.2 Přiřazení XML souborů k Externím tabulkám a jejich Views

- **Job_history.xml** - zip_job_history_xml_ext - view_ext_job_history
- **Jobs.xml** - zip_jobs_xml_ext - view_ext_jobs

4.3.3.3 Oracle Data-pump import

Pro tuto funkcionalitu jsou vytvořeny dva skripty využívající odlišný zdrojový DUMP soubor. Tento skript je uložen v metadata tabulce MD_WORKFLOWS v atributu script. Význam metadata tabulky MD_WORKFLOWS bude popsán níže. Datapumpové skripty jsou spouštěny v ETL nástroji za běhu WorkFlow, které bude popsáno níže. Skripty naleznete v příloze 6.

4.3.3.4 Oracle Directories ve schématu STAGE_HR

K spouštění Oracle funkcionalit určeným pro databázový server je nutné mít vytvořené objekty Oracle directories nad adresáři na databázovým serverem, protože bez nich by byly pro Oracle zdrojové soubory neviditelné. Protože má každá zdrojová sada

souborů své vytvořené sub-adresáře na databázovém serveru, je nutné, aby pod každým sub-adresářem bylo vytvořené Oracle directory. Není totiž možné, aby se jedno Oracle directory odkazovalo na více adresářů.

DIRECTORY_NAME	DIRECTORY_PATH
LAND_AREA_COMM	/home/oracle/landing_area/COMMERCIAL_DEPT
LAND_AREA_REG	/home/oracle/landing_area/REGIONAL_DEPT
LAND_AREA_HR	/home/oracle/landing_area/HR_DEPT
ERR_AREA_HR	/home/oracle/err_area/HR_DEPT
ERR_AREA_REG	/home/oracle/err_area/REGIONAL_DEPT
ERR_AREA_COMM	/home/oracle/err_area/COMMERCIAL_DEPT
HST_AREA_HR	/home/oracle/history_area/HR_DEPT
HST_AREA_REG	/home/oracle/history_area/REGIONAL_DEPT
HST_AREA_COMM	/home/oracle/history_area/COMMERCIAL_DEPT
EXEC_AREA	/home/oracle/execute_area

Obrázek 8: Seznam všech vytvořených Oracle Directories (vlastní)

4.3.4 PL/SQL procedury ve vstupní vrstvě STAGE_HR realizující datové plnění

4.3.4.1 Procedura proc_STAGE_imp_UTLF

Tato procedura je uložená v Package pkg_STAGE_UTL_LOAD_DATA. Procedura zpracovává soubory dodávané od externího zdroje REGIONAL_DEPT prostřednictvím Oracle funkcionality UTL_FILE. V této proceduře jsou vytvořeny lokální proměnné a přijímá následující vstupní parametry:

- P_DIR – Přijímá název Oracle Directory, v kterém se nachází zdrojový soubor. V tomto modelu je pro každé volání hodnota EXEC_AREA, protože se tento Oracle Directory odkazuje na výkonný adresář.
- P_FILENAME – Přijímá název zdrojového souboru, jehož obsah má být plněn do patřičné vstupní tabulky.
- P_INPUT_DATE – Přijímá datum, ke kterému jsou data ze zdrojového souboru vázány. Plní datumový atribut int_date.
- P_SOURCE – Přijímá název zdrojové skupiny (zdrojového systému).

Lokální proměnné:

- fhandle: Určuje záznamový typ souboru. Odkazuje se na privátní proměnnou pro UTL_FILE Package, která vytvoří záznam z metadata souboru, s kterým má UTL_FILE pracovat

- l_line: Proměnná typu VARCHAR2 s délkou 4000 znaků. V tomto případě se do této proměnné načítá celý řádek z CSV zdrojového souboru.

Proměnné typu ROWTYPE: Proměnné ROWTYPE se v PL/SQL odkazují na atributovou strukturu tabulky, Funguje na principu tečkové notace tabulka. atribut. Pro každý zdrojový soubor jsou vytvořeny ROWTYPE proměnné odkazující se na jejich vstupní tabulku.[28]

4.3.4.1.1 Výkonná oblast

Ve výkonné oblasti se nachází IF větvení, které na základě přiřazené hodnoty do parametru P_ FILENAME spouští Smyčku, v které se do proměnné l_line načítají pomocí procedury UTL_FILE.GET_LINE jednotlivé řádky ze zdrojového souboru, které jsou dále zpracovány SQL funkcemi SUBSTR a INSTR a přiřazovány do patřičné ROWTYPE proměnné. V dolní části smyčky se spouští DML příkaz INSERT, který vkládá záznamy do vstupní tabulky. Každá vstupní tabulka tohoto druhu musí být speciálně naprogramována v patřičné části ELSIF.

Smyčka se provádí, dokud není CSV soubor prázdný. Po skončení smyčky se provede příkaz COMMIT a vložené transakce ve vstupní tabulce potvrdí. Zdrojový kód procedury proc_STAGE_imp_UTLF najdete v příloze 4.

4.3.4.2 Procedura Proc_LOAD_EXT_DATA

Tato procedura je uložená v package pkg_STAGE_EXT_LOAD_DATA. Procedura zpracovává soubory dodávané od externího zdroje HR_DEPT prostřednictvím Oracle funkcionality Externích tabulek s preprocesorem. V této proceduře jsou vytvořeny lokální proměnné a přijímá následující vstupní parametry:

- P_FILENAME: Přijímá název zdrojového souboru, jehož obsah má být plněn do patřičné vstupní tabulky.
- P_INPUT_DATE – Přijímá datum, ke kterému jsou data ze zdrojového souboru vázány. Plní datumový atribut INT_DATE.
- P_SOURCE – Přijímá název zdrojové skupiny (zdrojového systému).

Lokální proměnné jsou pouze proměnné typu ROWTYPE, které se odkazují na vstupní tabulky. Popis typu ROWTYPE naleznete v popisu lokálních proměnných v proceduře proc_STAGE_imp_UTLF.

4.3.4.2.1 Výkonná oblast

Ve výkonné oblasti se nachází IF větvení, které na základě přiřazené hodnoty do parametru P_FILENAME spouští tzv. DML příkaz INSERT INTO SELECT FROM, kde je SQL dotaz sestaven z dané externí tabulky, či pohledu. Zdrojové kódy pro vytvoření procedury Proc_LOAD_EXT_DATA, Externích tabulek a views najdete v příloze 5.

4.3.4.3 Procedura Proc_LOAD_IMPDP_DATA

Tato procedura je uložená v balíku pkg_STAGE_IMPDP_LOAD_DATA. Procedura zpracovává záznamy od externího zdroje COMMERCIAL_DEPT, které jsou při volání procedury již importované v dočasných tabulkách impdp_clients a impdp_emp_cli.

V této proceduře jsou vytvořeny lokální proměnné a přijímá následující vstupní parametry:

- P_FILENAME: Přijímá zdrojového souboru, jehož obsah má být plněn do patřičné vstupní tabulky.
- P_INPUT_DATE – Přijímá datum, ke kterému jsou data ze zdrojového souboru vázány. Plní datumový atribut INT_DATE.
- P_SOURCE – Přijímá název zdrojové skupiny (zdrojového systému).

Lokální proměnné jsou pouze proměnné typu ROWTYPE, které se odkazují na vstupní tabulky. Popis typu ROWTYPE naleznete v popisu lokálních proměnných v proceduře proc_STAGE_imp_UTLF.

4.3.4.3.1 Výkonná oblast

Ve výkonné oblasti se nachází IF větvění, které na základě přiřazené hodnoty do parametru P_FILENAME spouští tzv. DML příkaz INSERT INTO … SELECT FROM, kde je SQL dotaz sestaven z dané dočasné tabulky. Zdrojové kódy pro vytvoření procedury Proc_LOAD_IMPDP_DATA najdete v příloze 6.

4.3.5 Procedura proc_STAGE_copy_delete_file a popis tabulky MD_STAGE_SRC_FILES

Poslední čtvrtá procedura se od předešlých procedur zásadně liší. Procedura proc_STAGE_copy_delete_file zajišťuje přesuny zdrojových souborů mezi adresáři na databázovém serveru prostřednictvím UTL_FILE procedury FRENAME. V této proceduře je vytvořena lokální proměnná l_filename, která slouží k transformaci konvence názvu zdrojového souboru. Tato procedura přijímá následující vstupní parametry:

- P_OLD_ORA_DIR – Přijímá Oracle Directory, z kterého má být přesun proveden.
- P_NEW_ORA_DIR – Přijímá Oracle Directory, do kterého má být přesun proveden.
- P_FILENAME – Přijímá zdrojového souboru, který má být přesunut. V případě přesunu všech zdrojových souborů z adresáře landing_area do execute_area je přiřazena hodnota "ALL".
- P_SOURCE – Přijímá název externího zdroje.

4.3.5.1 Výkonná oblast procedury proc_STAGE_copy_delete_file

Ve výkonné oblasti se nachází IF větvení, které na základě přiřazené hodnoty do parametrů P_OLD_ORA_DIR, P_FILENAME, P_SOURCE volá proceduru UTL_FILE.FRENAME, do které vstupují parametry přiřazené do procedury proc_STAGE_copy_delete_file. Pro případ, kdy je do parametru P_FILENAME přiřazena hodnota "ALL", byla vytvořena metadatová tabulka md_stage_src_files, v které jsou uloženy názvy zdrojových souborů (atribut file_name) a externích zdrojů, které zdrojové soubory dodávají (atribut source).

ID	FILE_PATH	FILE_NAME	REC_CREATED	LOCATION_AREA	SERVER_ID	SOURCE
1	C:\Sources_files\UTLSRC	regions.csv	11.03.2021 19:53:47	APP_SERVER	192.168.185.1	REGIONAL_DEPT
2	C:\Sources_files\UTLSRC	countries.csv	11.03.2021 19:53:47	APP_SERVER	192.168.185.1	REGIONAL_DEPT
3	C:\Sources_files\UTLSRC	locations.csv	11.03.2021 19:53:47	APP_SERVER	192.168.185.1	REGIONAL_DEPT
4	C:\Sources_files\UTLSRC	departments.csv	11.03.2021 19:53:47	APP_SERVER	192.168.185.1	REGIONAL_DEPT
5	C:\Sources_files\EXTSRC	emp_month_sal.csv.gz	11.03.2021 19:53:47	APP_SERVER	192.168.185.1	HR_DEPT
6	C:\Sources_files\EXTSRC	employees.csv.gz	11.03.2021 19:53:47	APP_SERVER	192.168.185.1	HR_DEPT
7	C:\Sources_files\EXTSRC	jobs.xml.gz	11.03.2021 19:53:47	APP_SERVER	192.168.185.1	HR_DEPT
8	C:\Sources_files\EXTSRC	job_hst.xml.gz	11.03.2021 19:53:47	APP_SERVER	192.168.185.1	HR_DEPT
9	C:\Sources_files\IMPDPSRC	imp_cli_emp.dmp	11.03.2021 19:53:47	APP_SERVER	192.168.185.1	COMMERCIAL_DEPT
10	C:\Sources_files\IMPDPSRC	imp_cli.dmp	11.03.2021 19:53:47	APP_SERVER	192.168.185.1	COMMERCIAL_DEPT

Obrázek 9: Obsah databázové metadata tabulky md_stage_src_files (vlastní)

V případě přiřazení hodnoty "*ALL*" do parametru P_FILENAME je v jedné z *ELSIF* větví spuštěna smyčka z SQL dotazu nad touto tabulkou s přiřazením parametru P_SOURCE na atribut *source*. Ve smyčce je spuštěna procedura *FRENAME*, která přesune celou sadu zdrojových souborů z adresáře landing_area do execute_area. Zdrojový kód pro vytvoření procedury proc_STAGE_copy_delete_file najdete v příloze 7.

4.3.6 Popis ETL WorkFlows, ostatních metadatových tabulek, scénáře metadata a business Workflows

4.3.6.1 Stručný popis modelových WorkFlows

V navrženém modelu byly vytvořeny dva typy WorkFlows. Dvě metadata WorkFlows a Čtyři business WorkFlows. Business WorkFlows zpracovávají svou zdrojovou sadu souborů, fungují na stejném principu a jsou spouštěny sériově. Metadata WorkFlows se od sebe liší.

Navržený model představuje jeden velký běh, který je sestaven z běhů spouštěných metadata a business WorkFlows.

4.3.6.2 Tabulka MD_WORKFLOWS

Tato metadata tabulka slouží k přípravě celého běhu modelu (resp. k běhům jednotlivých WorkFlows). V této tabulce se nachází záznamy, které obsahují část metadat, z kterých jsou generovány pro všechny business WorkFlows a metadata WorkFlow (wf_MASTERWORKFLOW) konfigurační soubory na server ETL nástroje (sever ETL nástroje se v tomto případě nachází na aplikačním serveru Windows) a následně ve WorkFlows využívány. Na obrázku č.10 jsou zachyceny výstupní hodnoty (metadata) z této tabulky, které jsou v tomto modelu generovány do konfiguračních souborů.

WORKFLOWNAME	LAYER_TYPE	FOLDER	SCRIPT	SOURCE
wf_STAGE_UTL_FILES	STAGE	LOAD_STAGE_FILES		REGIONAL_DEPT
wf_STAGE_EXT_FILES	STAGE	LOAD_STAGE_FILES		HR_DEPT
wf_STAGE_IMPDP_FILES_CLI_EMP	STAGE	LOAD_STAGE_FILES	impdp userid=stage_HR/hr@PDB2 directon	COMMERCIAL_DEPT
wf_MASTERWORKFLOW	MASTER	LOAD_STAGE_FILES		
wf_STAGE_IMPDP_FILES_CLIENTS	STAGE	LOAD_STAGE_FILES	impdp userid=stage_HR/hr@PDB2 directory	COMMERCIAL_DEPT

Obrázek 10: Obsah databázové metadata tabulky md_workflows (vlastní)

4.3.6.3 Tabulka SCHEDULED_WORKFLOW_RUNS

Do této kontrolní metadata tabulky se na začátku běhu modelu vkládají

připravené business Workflows a v průběhu běhu modelu aktualizuje průběh právě běžícího business Workflow. Na obrázku č.11 je zachycen počáteční stav business Workflow's.

WF_RUN_ID	WORKFLOW_NAME	PRIORITY	INPUT_DATE	STATUS	ACTIVE_FLAG	RUNDATE	RUN_ID
	wf_STAGE_UTL_FILES	1	09.03.2021	READY	1		
	wf_STAGE_EXT_FILES	2	09.03.2021	READY	1		
	wf_STAGE_IMPDP_FILES_CLI_EMP	3	09.03.2021	READY	1		
	wf_STAGE_IMPDP_FILES_CLIENTS	4	09.03.2021	READY	1		

Obrázek 11: Obsah databázové metadata tabulky scheduled_workflows_runs (vlastní)

Kontrolní tabulka se průběžně aktualizuje v průběhu spuštěných Master Sessions v metadata a business Workflows (wf_MASTERWORKFLOW) na základě náběrových SQL dotazů v transformacích Source Qualifier.

4.3.6.4 Popis jednotlivých metadata WorkFlows

4.3.6.4.1 Metadata wf_MASTER_Scheduled_WorkFlows

Toto WorkFlow je spuštěno pouze na začátku celého běhu modelu. Funkcí tohoto WorkFlow je vytvořit (připravit) konfigurační soubory z tabulky md_workflows (ParameterFiles) pro ostatní WorkFlows a vytvořit kontrolní (počáteční) záznamy v tabulce scheduled_workflows_runs pro jednotlivá business WorkFlows.

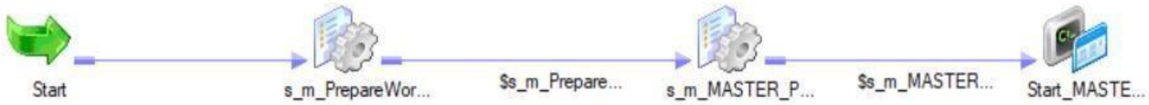
Scénář:

1. s_m_PrepareWorkFlow **(Session)** - Připraví konfigurační soubory všech ostatních WorkFlows.
2. s_m_MASTER_Prepare_runs **(Session)** - Připraví záznamy v kontrolní tabulce pro jednotlivé Business WorkFlows.
3. Start_MASTERWORKFLOW **(Command Task)** – Spustí wf_MASTERWORKFLOW pomocí PMCMD příkazu.

Pro toto WorkFlow je přiřazený statický konfigurační soubor, v kterém se nachází potřebné parametry:

- $$INPUT_DATE – Datumová hodnota, která se se používá pro plnění kontrolní tabulky v atributu input_date.

- $$WORKFLOWNAME – Název Master Workflow, které je spuštěno na konci wf_MASTER_Scheduled_WorkFlows. Název se přiřazuje do skriptu, kterým se spouští wf_MASTERWORKFLOW z Command Tasku.
- $$FOLDER – Název adresáře na ETL serveru, v kterém jsou vytvořeny všechny konfigurační soubory. Název se přiřazuje do skriptu, který spouští wf_MASTERWORKFLOW.



Obrázek 12: vytvořená metadata wf_MASTER_Scheduled_WorkFlows (vlastní)

4.3.6.4.2 Metadata wf_MASTERWORKFLOW

Toto WorkFlow je poprvé voláno na konci wf_MASTER_Scheduled_WorkFlows. Má na starost sériově spouštět jednotlivé business WorkFlows, a aktualizovat jejich stav běhu v kontrolní tabulce. Je spouštěno cyklicky po každém úspěšném, či neúspěšném běhu daného business WorkFlow. Aktualizace záznamů v tabulce scheduled_workflows_runs funguje na principu předávání hodnot wf proměnných do mapping proměnných prostřednictvím Master Sessions. Předaná hodnota v mapování je dále použita v transformaci a namapována do kontrolní tabulky. Všechny Master Sessions jsou nastavené pro aktualizace záznamu kontrolní tabulky.

Pro toto WorkFlow je přiřazený dynamický konfigurační soubor, v kterém se nachází potřebné parametry:

- **$$INPUT_DATE** – Datumová hodnota, která se předáva v průběhu celého běhu modelu a plní se s ní Atribut INT_DATE v cílových tabulkách.
- **$$FOLDER –** Název adresáře na ETL serveru, v kterém jsou vytvořeny všechny konfigurační soubory. Název se přiřazuje do skriptu, který spouští Master WorkFlow.

Obrázek 13: vytvořená metadata wf_MASTERWORKFLOW (vlastní)

Scénář:

1. Assign_WF_RUN_ID (**Assignment Task**)
2. s_m_MASTER_WF_RUN_ID (**Session**)
3. s_m_MASTER_GetWfName (**Session**)
4. Start_Secondary_WF (**Command Task**)

4.3.6.4.3 Metadata master sessions

Metadata Master Sessions (níže vypsané) spouštějí náběrové SQL dotazy, které jsou postaveny tak, aby vybraly vždy jeden požadující záznam (business WorkFlow) na základě jejich proměnných přiřazených k atributům *status*, *input_date* a *workflow_name* a sestupně setříděných záznamu atributu *priority*. Hodnoty do mapping proměnných jsou přiřazeny z workflow proměnných ve WorkFlow Sessions a nabývají různých hodnot. Mezi metada master sessions patří:

- **m_MASTER_WF_RUN_ID** – Aktualizace atributu wf_run_id z vygenerovaného $PMWorkflowRunId, což je automaticky generovanána hodnota v mappingu (Session), který se s každým spuštěním této Session mění. Tato hodnota v kontrolní tabulce indikuje, který běh (resp. ID běhu) wf_MASTERWORKFLOW spustilo business WorkFlow. Dochází k přiřazení parametru z $$WF_INPUT_DATE do $$m_INPUT_DATE, které je pak dosazeno do náběrového SQL dotazu a na vybraném záznamu business Workflow se aktualizuje atribut wf_run_id.
- **m_MASTER_GetWFName** – Aktualizace atributu status ze stavu "READY" do stavu "RUNNING" na vybraném záznamu business WorkFlow na základě přiřazení parametru z $$WF_INPUT_DATE do $$m_INPUT_DATE, které je pak dosazeno do náběrového SQL dotazu. Tato Session vrací do $$WF_NAME hodnotu z $$m_WF_NAME, která je dále dosazena jako parametr do pmcmd skriptu, z kterého se na konci běhu wf_MASTERWORKFLOW spouští business WorkFlow.
- **m_SEC_MASTER_Update_WF** – Aktualizace atributu run_id, které bylo v mapování vygenerovano z $PMWorkflowRunId. Tato hodnota v kontrolní tabulce indikuje ID běhu business WorkFlow. Výběr záznamu je proveden na základě přiřazení parametru z $$WORKFLOWNAME do $$m_WF_NAME, které je pak dosazeno do náběrového SQL dotazu. Parametr

$$WORKFLOWNAME je vygenerován v konfiguračních souborech, které jsou svým business a metadat WorkFlows na začátku běhu modelu vygenerovány.

- **m_SEC_MASTER_WF_END** – Aktualizace atributu status hodnotami "SUCCEEDED", nebo "FAILED". Výběr záznamu je proveden na základě přiřazení parametru $$WORKFLOWNAME a $$m_INPUT_DATE do $$m_WF_NAME a $$m_INPUT_DATE, které jsou pak dosazeny do náběrového SQL dotazu. Tato Master Session určuje konečný stav business Workflow v kontrolní tabulce.

4.3.6.5 Business WorkFlows

Business Workflows fungují na stejném principu. Kromě svých parametrů z konfiguračních souborů mají vytvořené své workflow proměnné, ke kterým jsou vždy na začátku WorkFlow přiřazené potřebné hodnoty. V tomto modelu představují parametry a funkce metadata, jako jsou názvy souborů a názvy patřičných Oracle Directories, které jsou posílány jako parametry do volaných databázových procedur. Snímky business Workflow's z ETL nástroje najdete v přílohách 8, 9, 10. U scénářů nahradí tyto snímky schéma, na kterém budou znázorněny běhy jednotlivých WorkFlow's.

4.3.6.5.1 Přiřazení Business WorkFlows externím skupinám:

- **wf_STAGE_UTL_FILES**: Zpracovává zdrojové sady souborů dodávané od zdrojové skupiny REGIONAL_DEPT.
- **wf_STAGE_EXT_FILES**: Zpracovává zdrojové sady souborů dodávané od zdrojové skupiny HR_DEPT.
- **wf_STAGE_IMPDP_FILES_CLI_EMP** a **wf_STAGE_IMPDP_FILES_CLIENTS** zpracovává zdrojové sady souborů dodávané od zdrojové skupiny COMMERCIAL_DEPT.

Každé business WorkFlow ma vytvořené proměnné, do kterých se vkládá celý název zdrojoveho souboru. Význam WorkFlow proměnných je stejný, slouží k předávání mapping proměnných, s kterými se volají databázové procedury.

Konfigurační soubory se liší v parametru $$SOURCE, kde pro tento parametr je přiřazena hodnota z tabulky md_workflows vždy jiná. Pro business WorkFlows, které zpracovávají zdrojové soubory DMP je jejich konfiguračním souboru přiřazen Datapumpový skript do parametru z $$SCRIPT. Ostatní WorkFlows mají tento parametr

prázdný, protože žádný skript pro ostatní Oracle funkcionality nebyl potřeba. Hodnoty jsou uvedeny v tabulce md_workflows.

4.3.6.6 Scénáře Business WorkFlows

Pro všechny business WorkFlows, platí dva možné scénáře. Což je vidět na obrázku č.14, který znázorňuje běh pro wf_STAGE_UTL_FILES a wf_STAGE_EXT_FILES. Pro business Workflows využívající Oracle Data-pump import bude znázorněno jiné schéma, avšak se stejným scénářem.

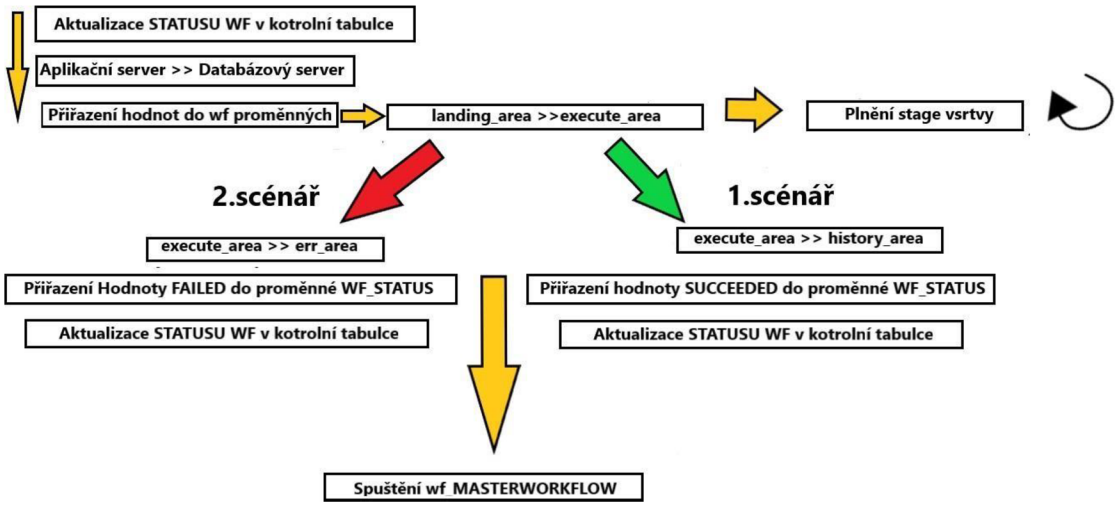
Scénář pro wf_STAGE_EXT_FILES zde uvedeny nebudou, jelikož jsou stejné jako pro wf_STAGE_UTL_FILES a liší se akorát v názvech wf proměnných, jejich hodnot a v názvech Sessions a Tasků.

Scénář pro wf_STAGE_IMPDP_FILES_CLIENTS zde uvedeny nebudou, jelikož je stejný jako pro wf_STAGE_IMPDP_FILES_CLI_EMP a též se liší pouze v názvech wf proměnných, jejich hodnot a v názvech Sessions a Tasků.

Scénář wf_STAGE_UTL_FILES – Úspěšné naplnění vstupních tabulek:

1. Asn_clear_STATUS (Assignment Task) – Přiřazení prázdné hodnoty do proměnné $$WF_STATUS.
2. s_m_SEC_MASTER_Update_WF (Session)
3. putFilesRegionalDept (Command Task) – Spustí batch skript, který přesune zdrojové sady souborů LAND_AREA_REG.
4. asn_proc_par_regions (Assignment Task) – Přiřazení hodnot všech potřebných wf proměnných.
5. s_m_STAGE_LAND_TO_EXEC (Session) – Volání procedury proc_STAGE_copy_delete_file, která přesune celou zdrojovou sadu z LAND_AREA_REG do EXEC_AREA.
6. s_m_STAGE_UTL_DATA (Session) – Volání procedury proc_STAGE_imp_UTLF, která plní vstupní tabulku zdrojovým souborem, jehož název je přiřazena ve své workflow proměnné.
7. s_m_STAGE_COPY_DELETE....HST (Session) - Volání procedury proc_STAGE_copy_delete_file, která přesune importovaný zdrojový soubor z EXEC_AREA do HST_AREA_REG.

8. 6. a 7. krok se opakuje, dokud nejsou importovány všechny. Pro každý soubor existují stejně fungující sessions, liší se akorát v názvu.
9. asn_SUCCESS (Assignment Task) – Přiřazení hodnoty "SUCCEEDED" do proměnné $$WF_STATUS.
10. s_m_SEC_MASTER_WF_END_SUCCEED (Session)
11. StartMasterfromEXT_FILES (Command Task) – Spuštění wf_MASTERWORKFLOW, které připraví a následně spustí běh Business WorkFlow wf_STAGE_EXT_FILES.



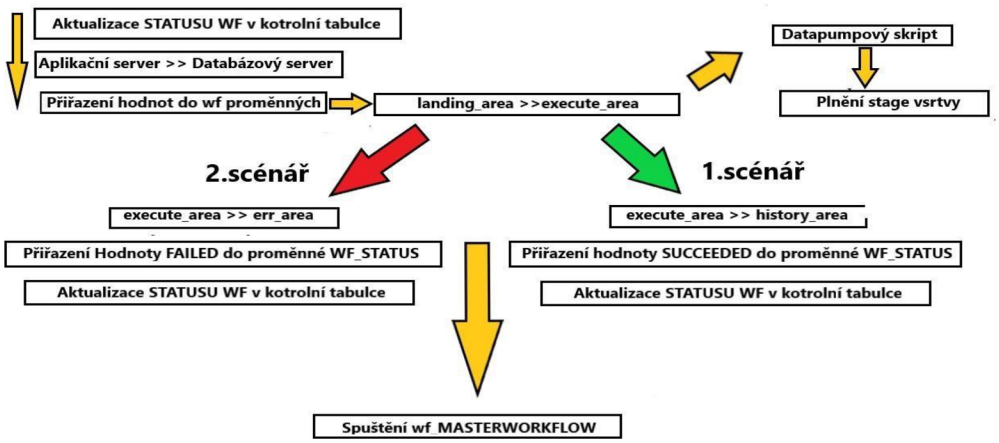
Obrázek 14: Schéma WorkFlow pro wf_STAGE_UTL_FILES a wf_STAGE_EXT_FILES (vlastní)

Na obrázku 14 znázorňují žluté šipky posloupnost jednotlivých kroků uvedené ve scénářích. Zelená šipka znázorňuje úspěšný import zdrojového souboru a červená neúspěšný. Ke druhému scénáři dochází v případě nenaplnění jedné ze vstupních tabulek. V takovém případě je končí business WorkFlow ve status FAILED.

Scénář wf_STAGE_IMPDP_FILES_CLI_EMP – Úspěšné naplnění dočasných tabulek:

1. Asn_clear_STATUS (Assignment Task) – Přiřazení přazdné hodnoty do proměnné $$WF_STATUS.
2. s_m_SEC_MASTER_Update_WF (Session) – Aktualizace atributu RUN_ID v kontrolní tabulce.

3. PutFilesCommercialDeptEmpCli (Command Task) – Spustí batch skript, který přesune zdrojový DMP soubor LAND_AREA_REG.
4. asn_proc_par_Comm (Assignment Task) – Přiřazení hodnot všech potřebných wf proměnných.
5. s_m_STAGE_LAND_TO_EXEC (Session) – Volání procedury proc_STAGE_copy_delete_file, která přesune zdrojový DMP soubor z LAND_AREA_COMM do EXEC_AREA.
6. StartIMPDPclients (Command Task) – Spuštění Datapumpového importu prostřednictvím parametru $$SCRIPT z kofiguračního souboru.
7. s_m_STAGE_IMPDP_DATA_ ... (Session) – Volání procedury proc_LOAD_IMPDP_DATA, která plní stage tabulku z dočasných tabulek, naplněných Datapumpovým importem.
8. s_m_STAGE_COPY_DELETE....HST (Session) - Volání procedury proc_STAGE_copy_delete_file, která přesune importovaný zdrojový soubor z EXEC_AREA do HST_AREA_COMM
9. asn_SUCCESS (Assignment Task) – Přiřazení přazdné hodnoty "SUCCEEDED" do proměnné $$WF_STATUS.
10. s_m_SEC_MASTER_WF_END_SUCCEED (Session) - Aktualizace atributu STATUS v kontrolní tabulce.
11. StartMasterWorkflowCLIENTS (Command Task) – Spuštění wf_MASTERWORKFLOW, které připraví a následně spustí běh Business WorkFlow wf_STAGE_IMPDP_FILES_CLIENTS.

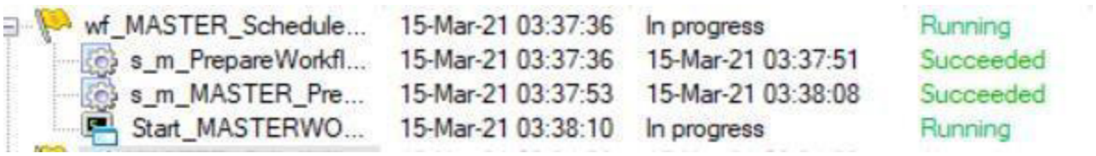


Obrázek 15: Schéma WorkFlow pro wf_STAGE_IMPDP_FILES_CLI_EMP a wf_STAGE_IMPDP_FILES_CLIENTS (vlastní)

Na obrázku č.15 znázorňují žluté šipky posloupnost jednotlivých kroků uvedené ve scénářích. Zelená šipka znázorňuje úspěšný import zdrojového souboru a červená neúspěšný. Ke druhému scénáři dochází v případě nenaplnění jedné ze vstupních tabulek. V takovém případě je končí business WorkFlow ve status FAILED. Jednotlivé batch skripty budou uvedeny v příloze 11.

4.3.7 Běh navrženého modelu

Následuje ukázka celého běhu modelu, tato ukázka bude obsahovat průběžné snímky běhu modelu. Posledním snímkem bude konečný stav business Workflows v tabulce schedule_workflow_runs. Snímky jsou tvořeny prostředí Monitor Manager ETL nástroje Informatica PowerCenter.



Obrázek 16: Dokončené wf_MASTER_Scheduled_WorkFlows (vlastní)

Na obrázku č.16 je zachyceno úspěšně doběhlé wf_MASTER_Scheduled_WorkFlows, které spustilo wf_MASTERWORKFLOW připravující běh pro wf_STAGE_UTL_FILES.

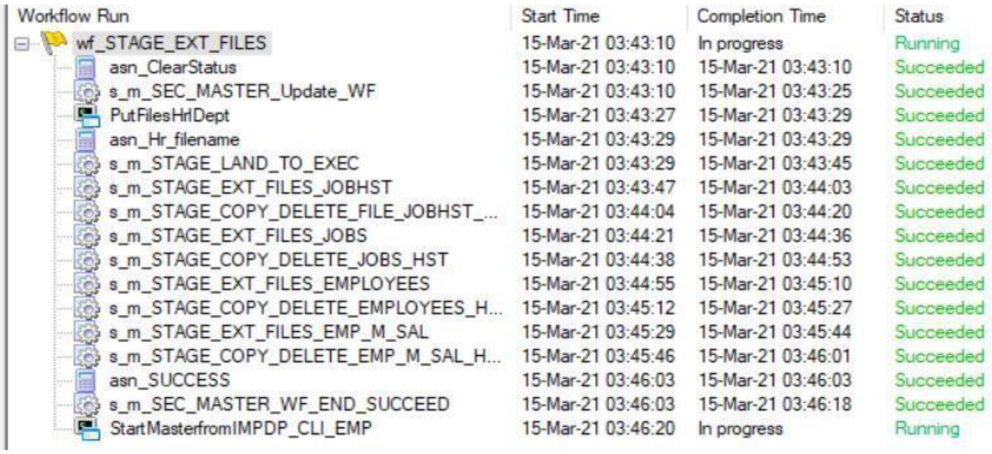
WF_RUN_ID	WORKFLOW_NAME	PRIORITY	INPUT_DATE	STATUS	ACTIVE_FLAG	RUNDATE	RUN_ID
	wf_STAGE_UTL_FILES	1	09.03.2021	READY	1		
	wf_STAGE_EXT_FILES	2	09.03.2021	READY	1		
	wf_STAGE_IMPDP_FILES_CLI_EMP	3	09.03.2021	READY	1		
	wf_STAGE_IMPDP_FILES_CLIENTS	4	09.03.2021	READY	1		

Obrázek 17: připravené business WorkFlows v tabulce schedule_workflow_runs (vlastní)

Na obrázku č.17 v tabulce wf_MASTER_Scheduled_WorkFlows zachyceny úspěšně připravené business Workflows.

Obrázek 18: Dokončené business WorkFlow wf_STAGE_UTL_FILES (vlastní)

Na obrázku č.18 je zachyceno úspěšně doběhlé wf_STAGE_UTL_FILES, které spustilo wf_MASTERWORKFLOW připravující běh pro wf_STAGE_EXT_FILES.



Workflow Run	Start Time	Completion Time	Status
wf_STAGE_EXT_FILES	15-Mar-21 03:43:10	In progress	Running
asn_ClearStatus	15-Mar-21 03:43:10	15-Mar-21 03:43:10	Succeeded
s_m_SEC_MASTER_Update_WF	15-Mar-21 03:43:10	15-Mar-21 03:43:25	Succeeded
PutFilesHrlDept	15-Mar-21 03:43:27	15-Mar-21 03:43:29	Succeeded
asn_Hr_filename	15-Mar-21 03:43:29	15-Mar-21 03:43:29	Succeeded
s_m_STAGE_LAND_TO_EXEC	15-Mar-21 03:43:29	15-Mar-21 03:43:45	Succeeded
s_m_STAGE_EXT_FILES_JOBHST	15-Mar-21 03:43:47	15-Mar-21 03:44:03	Succeeded
s_m_STAGE_COPY_DELETE_FILE_JOBHST_...	15-Mar-21 03:44:04	15-Mar-21 03:44:20	Succeeded
s_m_STAGE_EXT_FILES_JOBS	15-Mar-21 03:44:21	15-Mar-21 03:44:36	Succeeded
s_m_STAGE_COPY_DELETE_JOBS_HST	15-Mar-21 03:44:38	15-Mar-21 03:44:53	Succeeded
s_m_STAGE_EXT_FILES_EMPLOYEES	15-Mar-21 03:44:55	15-Mar-21 03:45:10	Succeeded
s_m_STAGE_COPY_DELETE_EMPLOYEES_H...	15-Mar-21 03:45:12	15-Mar-21 03:45:27	Succeeded
s_m_STAGE_EXT_FILES_EMP_M_SAL	15-Mar-21 03:45:29	15-Mar-21 03:45:44	Succeeded
s_m_STAGE_COPY_DELETE_EMP_M_SAL_H...	15-Mar-21 03:45:46	15-Mar-21 03:46:01	Succeeded
asn_SUCCESS	15-Mar-21 03:46:03	15-Mar-21 03:46:03	Succeeded
s_m_SEC_MASTER_WF_END_SUCCEED	15-Mar-21 03:46:03	15-Mar-21 03:46:18	Succeeded
StartMasterfromIMPDP_CLI_EMP	15-Mar-21 03:46:20	In progress	Running

Obrázek 19: Dokončené business WorkFlow wf_STAGE_EXT_FILES (vlastní)

Na obrázku č.19 je zachyceno úspěšně doběhlé wf_STAGE_EXT_FILES, které spustilo wf_MASTERWORKFLOW připravující běh pro wf_STAGE_IMPDP_CLI_EMP.

Workflow Run	Start Time	Completion Time	Status
wf_STAGE_IMPDP_FILES_CLI_EMP	15-Mar-21 03:47:14	15-Mar-21 03:49:07	Succeeded
asn_ClearStatus	15-Mar-21 03:47:15	15-Mar-21 03:47:15	Succeeded
s_m_SEC_MASTER_Update_WF	15-Mar-21 03:47:15	15-Mar-21 03:47:30	Succeeded
PutFilesCommercialDeptEmpCli	15-Mar-21 03:47:31	15-Mar-21 03:47:33	Succeeded
asn_proc_par	15-Mar-21 03:47:33	15-Mar-21 03:47:33	Succeeded
s_m_STAGE_LAND_TO_EXEC_CLIDMP	15-Mar-21 03:47:33	15-Mar-21 03:47:49	Succeeded
StartIMPDPemp_cli	15-Mar-21 03:47:50	15-Mar-21 03:48:04	Succeeded
s_m_STAGE_CLIDMP_HST	15-Mar-21 03:48:04	15-Mar-21 03:48:19	Succeeded
s_m_STAGE_IMPDP_DATA	15-Mar-21 03:48:21	15-Mar-21 03:48:36	Succeeded
asn_SUCCESS	15-Mar-21 03:48:37	15-Mar-21 03:48:37	Succeeded
s_m_SEC_MASTER_WF_END_SUCCESS	15-Mar-21 03:48:37	15-Mar-21 03:48:53	Succeeded
StartMasterWorkflowCLIENTS	15-Mar-21 03:48:53	15-Mar-21 03:49:07	Succeeded

Obrázek 20: Dokončená metadata wf_STAGE_IMPDP_CLI_EMP (vlastní)

Na obrázku č.20 je zachyceno úspěšně doběhlé pro wf_STAGE_IMPDP_CLI_EMP, které spustilo wf_STAGE_IMP.DP_CLIENTS.

Workflow Run	Start Time	Completion Time	Status
wf_STAGE_IMPDP_FILES_CLIENTS	15-Mar-21 03:49:39	15-Mar-21 03:51:08	Succeeded
asn_ClearStatus	15-Mar-21 03:49:44	15-Mar-21 03:49:44	Succeeded
s_m_SEC_MASTER_Update_WF	15-Mar-21 03:49:44	15-Mar-21 03:49:59	Succeeded
PutFilesCommercialDeptClients	15-Mar-21 03:50:00	15-Mar-21 03:50:02	Succeeded
asn_proc_par	15-Mar-21 03:50:02	15-Mar-21 03:50:02	Succeeded
s_m_STAGE_LAND_TO_EXEC_CLIDMP	15-Mar-21 03:50:02	15-Mar-21 03:50:17	Succeeded
StartIMPDPclients	15-Mar-21 03:50:17	15-Mar-21 03:50:22	Succeeded
s_m_STAGE_CLIDMP_HST	15-Mar-21 03:50:22	15-Mar-21 03:50:37	Succeeded
s_m_STAGE_IMPDP_DATA	15-Mar-21 03:50:38	15-Mar-21 03:50:53	Succeeded
asn_SUCCESS	15-Mar-21 03:50:53	15-Mar-21 03:50:53	Succeeded
s_m_SEC_MASTER_WF_END_SUCCESS	15-Mar-21 03:50:53	15-Mar-21 03:51:08	Succeeded

Obrázek 21: Dokončená metadata wf_STAGE_IMPDP_CLIENTS (vlastní)

Na obrázku č.21 je zachyceno úspěšně dokončené wf_STAGE_IMPDP_CLIENTS, které bylo posledním business WorkFlow.

```
115 SELECT s.*,s.ROWID FROM stage_hr.SCHEDULED_WORKFLOW_RUNS s;
116
```

	WF_RUN_ID	WORKFLOW_NAME	PRIORITY	INPUT_DATE	STATUS	ACTIVE_FLAG	RUNDATE	RUN_ID
1	2444	wf_STAGE_UTL_FILES	1	09.03.2021	SUCCEEDED	1	15.03.2021 03:38:52	2445
2	2446	wf_STAGE_EXT_FILES	2	09.03.2021	SUCCEEDED	1	15.03.2021 03:42:58	2447
3	2448	wf_STAGE_IMPDP_FILES_CLI_EMP	3	09.03.2021	SUCCEEDED	1	15.03.2021 03:47:00	2449
4	2450	wf_STAGE_IMPDP_FILES_CLIENTS	4	09.03.2021	SUCCEEDED	1	15.03.2021 03:49:33	2451

Obrázek 22: Konečný stav metadata tabulky schedule_workflow_runs business WorkFlows po běhu modelu (vlastní)

Obrázek č.22 slouží jako důkaz, že všechny business Workflows doběhly úspěšně a datový sklad byl naplněn zdrojovými daty, které se vážou k 09.03.2021. Což dokládá i obrázek 23, na kterém je proveden SQL dotaz nad jednou ze vstupních tabulek a nebo obrázek 24, kde jsou zahistorizované zdrojové soubory od externí skupiny COMMERCIAL_DEPT.

```
SELECT * FROM stage_hr.clients_employees
```

	CLIENT_ID	EMPLOYEE_ID	INT_DATE	SOURCE	CREATED_DATE	CL_EMP_ID	STG_CL_EMP_ID
3	3045809	165	09.03.2021	COMMERCIAL_DEPT	15.03.2021 03:48:36	480	178920
2	3045808	165	09.03.2021	COMMERCIAL_DEPT	15.03.2021 03:48:36	479	178919
5	3045811	166	09.03.2021	COMMERCIAL_DEPT	15.03.2021 03:48:36	482	178922
4	3045810	166	09.03.2021	COMMERCIAL_DEPT	15.03.2021 03:48:36	481	178921

Obrázek 23: Stav tabulky stage_hr.clients_employees po běhu modelu (vlastní)

/home/oracle/history_area/COMMERCIAL_DEPT/

Name	Size	Changed
..		10-Mar-21 10:54:40
210315_hst_imp_cli.dmp	284 KB	15-Mar-21 03:50:00
210315_hst_imp_cli_emp.dmp	228 KB	15-Mar-21 03:47:32

Obrázek 24: Zahistorizované zdrojové soubory s datumovým prefixem v history_area po běhu modelu (vlastní)

5 Výsledky a diskuse

5.1 Zhodnocení výsledků

Výsledkem práce je funkční ELT model, který integruje zdrojové datové soubory ve vytvořené vstupní vrstvě datového skladu. Mimo tento úkol ELT model také zajišťuje přenos zdrojových datových souborů z aplikačního serveru na databázový server a poskytuje historizační řešení zpracovaných, či nezpracovaných datových souborů do patřičných adresářů na databázovém serveru.

To vše je dosaženo na základě kombinace ETL nástroje a databázových funkcionalit, které byly použity pro vývoj celého ELT modelu. Taková kombinace v tomto případě představuje velice rychlou integraci dat, jelikož datové plnění ve vstupní vrstvě zajišťují databázové funkcionality (resp. naprogramované databázové procedury) a tím se nezatěžuje integrační služba ETL nástroje, ale databázová platforma Oracle, která je velice výkonná. ELT model byl záměrně navržen tak, aby ho tvořila výše uvedená kombinace rozdílných technologií. Tímto totiž vytvořený ELT model znázorňuje ukázku jisté flexibility mezi vybranými technologiemi a následná implementace jiného business modelu by mohla v praxi představovat optimální řešení.

Integrace dat je v této práci zajištěna databázovými funkcionality UTL_FILE, Oracle Data-pump a externími tabulky. Tyto funkcionality bylo vhodné použít, protože jsou datové soubory dodávány na databázový server, z kterého jsou výše uvedené databázové funkcionality spustitelné.

Díky vytvořeným meta-datovým tabulkám (metadat) a parametrických souborů, je zajištěn běh celého ELT modelu a také aktualizace kontrolních záznamů, díky kterým je možné kontrolovat jednotlivé běhy business WorkFlow's.

Důležitou část ELT modelů tvoří také batch (v mnoha případech i shell) skripty. Jelikož jsou v praxi zdrojové datové soubory dodávány na aplikační sever, je nutné zajistit jejich přesun na cílový server, soubory zpracovávat a na základě zpracování datové soubory historizovat. Součástí vytvořeného modelu v této práci jsou vytvořené batch skripty, které zajišťují přesun datových souborů z aplikačního server na databázový server. Vytvořené batch skripty se nachází v příloze 11.

K navrženému řešení nutno dodat, že není plně dynamické, vyskytují se v něm pouze prvky metadatového přístupu. To je vidět v přílohách zdrojových kódů databázových

PL/SQL procedur, z kterých je zřejmé, že jsou vytvořeny spíše staticky. Model by rozhodně stálo za to modifikovat do dynamické podoby a měl by být plně meta-datově řízen. Taková modifikace by rozhodně ulehčila další případný vývoj a opravy modelu by probíhaly na méně místech.

Ovšem v této bakalářské práci byl model záměrně navržen tak, aby poukazoval na kombinaci ETL nástroje a databázových funkcionalit a tím se odkazovat jako možné řešení pro hlavní problematiku této práce.

Navržené řešení v této práci nemůže být nějak výrazně ohroženo. Navržené řešení můžou ohrozit stejné faktory, které ohrožují jiná, podobná funkční řešení. Mezi ohrožující faktory patří nekvalitní zdrojové datové soubory, které by způsobily v tomto stavu modelu pád celého běhu modelu, nicméně pro tuto hrozbu je připraveno historizační řešení. Mezi další faktory patří nekvalitní uživatelé, smazání databázových objektů (databázové balíky obsahující databázové PL/SQL procedury), kompilace chybných (starých) verzí databázových objektů, či pád ETL nebo databázového serveru.

6 Závěr

Tato bakalářská práce se zabývala tvorbou ELT modelu, který integruje data ve vstupní vrstvě datového skladu pomocí databázových funkcionalit Oracle.

Teoretická část si kladla za cíl popsat databázové funkcionality pro zpracování souborů jako zdrojů dat pro databázi Oracle 19c určené především k importu dat a jejich možnostmi použití ze strany klienta a databázového serveru. Teoretická část by měla na základě důkladné analýzy databázových funkcionalit objasnit princip jejich využití. Teoretická část se dále zabývala pojmy ETL, ELT a datovými sklady, jelikož s těmito pojmy úzce souvisí hlavní problematika této práce a nechybí ani popis relačního modelu databáze, či databázového serveru Oracle.

Kapitola vlastní práce si v kladla za cíl popsat ETL nástroj, který bude využíván pro dosažení hlavního cíle této práce. Hlavním cílem této práce bylo navrhnout a vytvořit funkční ELT model na základě popisu business modelu, který bude prostřednictvím kombinace vybraných databázových funckionalit z teoretické části a ETL nástroje zpracovávat zdrojové soubory a integrovat data ve vytvořené vstupní vrstvě datového skladu. Databázové funkcionality vybrané pro integraci dat, jsou ve vytvořeném modelu sériově volány ve vytvořených business Workflow's v ETL nástroji. Cílem navrženého databázového modelu bylo mimo jiné také zajistit přesun zdrojových datových souborů na cílový server a vytvořit historizační řešení, které zdrojové soubory na základě implementované business logiky ukládá do archivačních a chybových adresářů.

Navržené řešení se z velké části opírá o techniky, popsané v teoretické části. Pro ELT model byly použity databázové funkcionality spustitelné z databázového serveru, a to UTL_FILE, Oracle Data-pump IMPDP a externí tabulky s preprocesorem. Tyto databázové funkcionality zajišťují integraci dat (datové plnění) ve vstupní vrstvě datového skladu, která byla vytvořená v kapitole vlastní práce a je její součástí.

Hlavním přínosem navrženého modelu je představení, jakým způsobem se dají kombinovat databázové funkcionality Oracle s ETL nástroji a tím poukázat na jistou flexibilitu mezi těmito technologiemi. Tato flexibilita poukazuje na další způsob, jak se také dají tvořit ELT modely. Velká výhoda tohoto modelu je, že integrace dat ve vstupní vrstvě datového skladu probíhá na databázové platformě Oracle, která je velmi výkonná a tím se nezatěžuje integrační služba ETL nástroje. To má za následek vysoký výkon modelu.

Další výhodou tohoto modelu je, že přesun zdrojových datových souborů mezi adresáři na databázovém serveru efektivně zajišťuje procedura FRENAME, která je součástí databázového funkcionality (balíku) UTL_FILE. Díky tomu nebyla potřeba tvořit shell skripty a tím se ušetřil čas vývoje ELT modelu v této práci.

K navrženému modelu nutno dodat, že je plně funkční, což i dokládá ukázka běhu v poslední podkapitole vlastní práce. Ovšem nabízejí se další možnosti jeho vývoje.

Pokud by byl model modifikován do plně dynamické podoby a byl by meta-datově řízen, mělo by to za následek snadnější údržbu a opravy modelu by neprobíhaly na více místech. Toto by byla další velká výhoda.

Další vývojem, který se pro tento ELT model nabízí, je upravit jeho sériový běh na paralelní běh. Jelikož integrace dat probíhá na databázové platformě, nebyl by problém využít zdroje databázového serveru (počet CPU a paměti) k dosažení co nejvyššího výkonu paralelním zpracováním. Takto by byla spuštěna kompletní integrace vstupní vrstvy v jednom kroku, nikoliv postupně. To by mělo za následek, že by celý běh ELT modelu trval mnohem kratší dobu.

7 Seznam použitých zdrojů

1. PROCHÁZKA, David. *ORACLE: Průvodce správou, využitím a programováním nad databázovým systémem*. GRADA, 2009. ISBN 978-80-247-2762-2
2. KYTE, Thomas. *ORACLE: Expert Oracle Database Architecture*. Berlin: Springer-Verlag Berlin and Heidelberg GmbH & Co. KG, 2010 ISBN 14-302-294-62
3. KYTE, Thomas. *ORACLE: Effective Oracle By Design*. New York: OSBORNE COMPUTING, 2003. ISBN 00-722-306-57
4. KYTE, Thomas, Klaus. *ORACLE: Expert One-On-One Oracle*. New York: APRESS, 2005. ISBN 15-905-924-33
5. Webové stránky ORACLE. *What is a Relational Database (RDBMS)?* [online]. ORACLE, 2011. [cit. 2020-09-11]. Dostupné z: https://www.oracle.com/cz/database/what-is-a-relational-database/
6. KYTE, Joe McCORMACK. *Oracle Database Database Concepts 19c: Database concepts* [online]. ORACLE, 2019 [cit. 2020-09-11]. Dostupné z: https://docs.oracle.com/en/database/oracle/oracle-database/19/cncpt/database-concepts
7. BOBROWSKI, Cindy CLOSKEY. *Oracle 7 Server Concepts Manual: Introduction to the Oracle Server* [online]. ORACLE, 2005 [cit. 2020-09-11]. Dostupné z: https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch1.htm
8. RICH Kathy. *Oracle 9i Database Utilities: Export and Import* [online]. ORACLE, 2007 [cit. 2020-09-11]. Dostupné z: https://docs.oracle.com/cd/B10501_01/server.920/a96652/part1.htm
9. LORENTZ, Diana. *Oracle Database Online Documentation 10g: Data Warehousing* [online]. ORACLE, 2008 [cit. 2020-09-11]. Dostupné z: https://docs.oracle.com/cd/B19306_01/server.102/b14215/ldr_concepts.htm
10. CHAN, Immanuel. *Oracle Database Online Documentation 11g: Data Warehousing and Business Intelligence* [online]. ORACLE, 2009 [cit. 2020-09-11]. Dostupné z: https://docs.oracle.com/cd/B28359_01/server.111/b28319/exp_imp.htm#g1070082
11. BRUMM, Allen. *Oracle Database Online Documentation 10g: Administration* [online]. ORACLE, 2008 [cit. 2020-09-18]. Dostupné z: https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_5007.htm
12. MIQUEL, Laura Hofman. *Oracle Fusion Middleware Online Documentation Library* [online]. ORACLE, 2010 [cit. 2020-09-18]. Dostupné z:

https://docs.oracle.com/cd/E12839_01/admin.1111/e10046/und_ovd.htm#OVDAG101

13. KYTE, ASHDOWN Lance. *Oracle Database Online Documentation 12c: Database Administration* [online]. ORACLE, 2012 [cit. 2020-10-05]. Dostupné z: https://docs.oracle.com/database/121/ARPLS/u_file.htm#ARPLS069
14. BURLESON, Don. *UTL_FILE package* [online]. [cit. 2020-10-05]. 2012, [cit. 2020-10-05]. Dostupné z: http://www.dba-oracle.com/t_utl_file_put.htm
15. BRIDGE, Bill. *Oracle Database Online Documentation 10g: Data Warehousing* [online]. ORACLE, 2008 [cit. 2020-10-05]. Dostupné z: https://docs.oracle.com/cd/B19306_01/server.102/b14215/dp_overview.htm
16. FOGEL, Steve. *Difference Between EXP/IMP and EXPDP/IMPDP jobs in Oracle* [online]. 2011 [cit. 2020-10-05]. Dostupné z: https://smarttechways.com/2018/08/31/difference-between-exp-imp-and-expdp-impdp-jobs-in-oracle/
17. CHAITANYA, K. *Difference between exp/imp and expdp/impdp?* [online]. 2016 [cit. 2020-10-05]. Dostupné z: https://ckoracleworld.wordpress.com/2016/01/10/difference-between-expimp-and-expdpimpdp/
18. KYTE, Thomas. *Oracle Database Online Documentation 10g: Database Utilities* [online]. ORACLE, 2012 [cit. 2020-10-26]. Dostupné z: https://docs.oracle.com/cd/B19306_01/server.102/b14215/et_concepts.htm
19. LINSLEY, Peter. *Using the Preprocessor Feature with External Tables in Oracle Database 11g* [online]. ORACLE, 2010 [cit. 2020-10-26]. Dostupné z: https://download.oracle.com/otndocs/products/database/enterprise_edition/utilities/pdf/xtables_preproc11g_1009.pdf
20. MURRAY, Chuck. *External Tables and XMLTAG to Load XML Documents in Oracle Database 12c Release 2* [online]. 2011 [cit. 2020-11-12]. Dostupné z: https://oracle-base.com/articles/12c/exteral-tables-and-xmltag-to-load-xml-documents-12cr2
21. KIMBALL, ROSS Margy. *Data Warehouse Toolkit*. New Jersey: John Wiley & Sons, 2004. ISBN:11-185-308-02
22. KIMBALL, CASERTA Joe. *Data Warehouse ETL Toolkit:Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. New Jersey: John Wiley & Sons, 2004. ISBN 07-645-675-78

23. WANG, Ted. *Informatica B2BDT: Automatic Mapping Documentation Generation* [online]. INFORMATICA, 2015 [cit. 2021-02-05]. Dostupné z: https://marketplace.informatica.com/listings/onpremise/solutions/automatic_mapping_documentation_generation.html
24. SMALLCOMBE, Mark. *ETL vs ELT: 5 Critical Differences* [online]. XPLENTY, 2021 [cit. 2021-10-31]. Dostupné z: https://www.xplenty.com/blog/etl-vs-elt/
25. ZOINER, Tejada. ETL and ELT [online]. MICROSOFT, 2021 [cit. 2021-10-31]. dostuné z: https://docs.microsoft.com/cs-cz/azure/architecture/data-guide/relational-data/etl
26. PANEC, Zdeněk. *Co je to Business Inteligence?* [online]. 2003 [cit. 2021-10-31]. Dostupné z: https://www.systemonline.cz/clanky/co-je-to-business-intelligence.htm
27. BETH, Mary. *Oracle Database Online Documentation 12c: Database VLDB and Partitioning Guide* [online]. ORACLE, 2012 [cit. 2021-02-19]. Dostupné z: https://docs.oracle.com/database/121/VLDBG/GUID-EAFD703C-EFA9-4819-85BD-79F63B761A96.htm#VLDBG1269
28. BRIDGE, Bill. *PL/SQL User's Guide and Reference 10g Release 1 (10.1)* [online]. ORACLE, 2012 [cit. 2021-02-19]. Dostupné z: https://docs.oracle.com/cd/B13789_01/appdev.101/b10807/13_elems042.htm

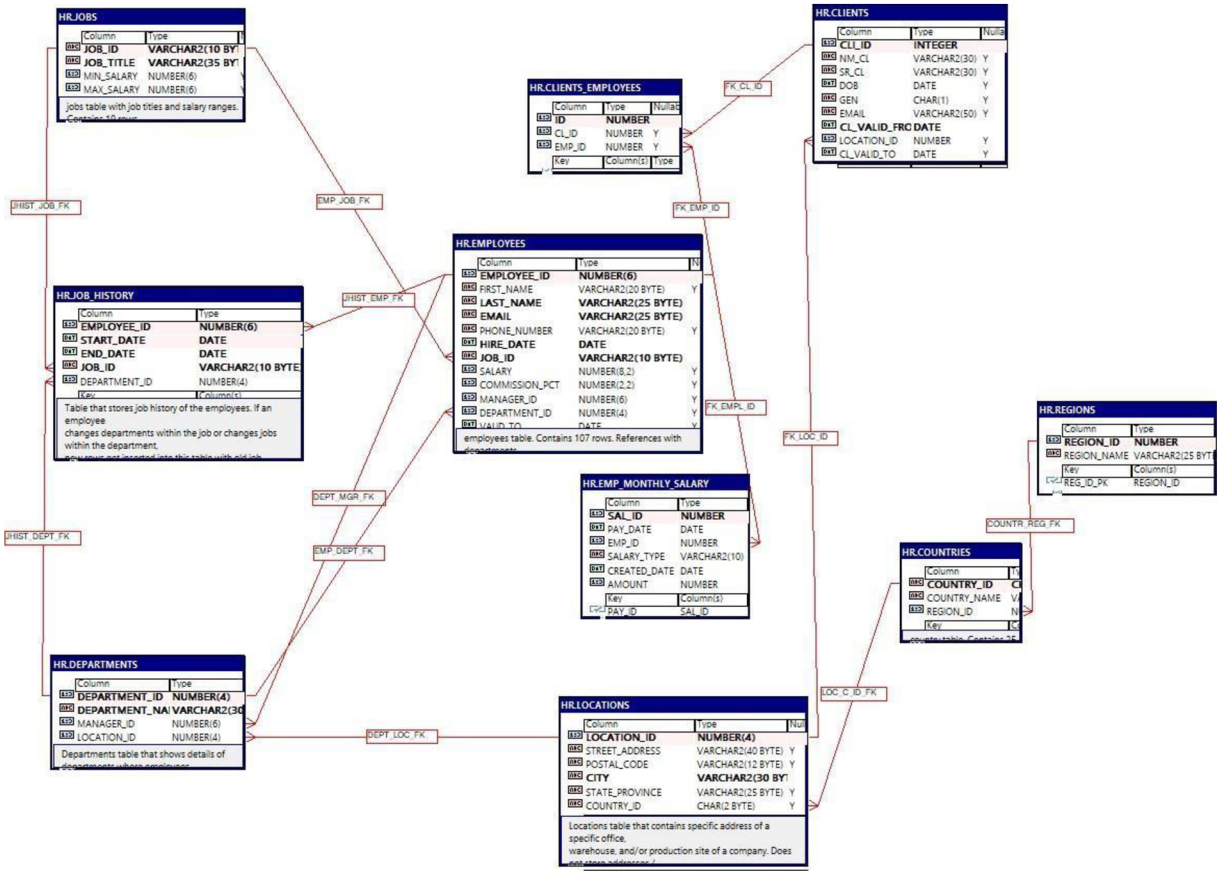
8 Přílohy


```
CREATE OR REPLACE PROCEDURE load_data( p_dir IN VARCHAR2,
                                       p_file IN VARCHAR2)
AS
  fhandle utl_file.file_type;
  l_line VARCHAR2(4000);
  l_name my_friends_tbl2.name%TYPE;
  l_surname my_friends_tbl2.surname%TYPE;
  l_born_in_year my_friends_tbl2.born_in_year%TYPE;
BEGIN
  fhandle := UTL_FILE.FOPEN(p_dir, p_file,'R');
    LOOP
      BEGIN

        UTL_FILE.GET_LINE(fhandle, l_line);
        l_name := SUBSTR(l_line, 1, INSTR(l_line, ',', 1, 1) - 1);
        l_surname :=
                  SUBSTR(
                          l_line,
                          INSTR(l_line,',', 1, 1) + 1,
                          INSTR(l_line, ',', 1, 2) - INSTR(l_line, ',', 1, 1) - 1
                         );
        l_born_in_year :=
                  SUBSTR(
                          l_line,
                          INSTR(l_line, ',', 1, 2) + 1
                         );

      INSERT INTO my_friends_tbl2(NAME, surname, born_in_year)
      VALUES(l_name, l_surname, l_born_in_year);

      EXCEPTION WHEN No_Data_Found
      THEN EXIT;

      END;
     END LOOP;
  UTL_FILE.FCLOSE(fhandle);
COMMIT;
END;
```

Příloha 1: Procedura load_data (UTL_FILE)

```
CREATE OR REPLACE PROCEDURE create_append_file (p_dir IN VARCHAR2,
                                                p_file IN VARCHAR2,
                                                p_line IN VARCHAR2)
AS
  fhandle utl_file.file_type;
BEGIN
  fhandle := utl_file.fopen(p_dir, p_file, 'A');
  utl_file.put_line(fhandle, p_line);
  utl_file.fclose(fhandle);

EXCEPTION
WHEN OTHERS THEN
dbms_output.put_line('ERROR: ' || SQLCODE || ' - ' || SQLERRM); RAISE;
END;
/
```

Příloha 2: Procedura *CREATE_APPEND_FILE* (UTL_FILE)




```
create table CLIENTS (
nm_cl VARCHAR2(30),
sr_cl VARCHAR2(30),
dob DATE,
gen CHAR(1),
email VARCHAR2(50),
valid_from  DATE, location_id NUMBER, int_date  DATE,
source  VARCHAR2(20), created_date DATE default SYSDATE, cli_id NUMBER,
stg_cli_id  NUMBER generated always as identity, valid_to DATE
)
PARTITION BY RANGE (int_date) INTERVAL ( INTERVAL '1' DAY ) (
PARTITION p0 VALUES LESS THAN ( DATE'2000-01-01' ));
```

```
alter table CLIENTS
add constraint PK_STG_CLI_ID primary key (STG_CLI_ID);

create table CLIENTS_EMPLOYEES (
client_id NUMBER,

employee_id NUMBER,
int_date  DATE default SYSDATE not null, source VARCHAR2(20),
created_date DATE default SYSDATE, cl_emp_id  NUMBER,
stg_cl_emp_id NUMBER generated always as identity
)
PARTITION BY RANGE (int_date) INTERVAL ( INTERVAL '1' DAY ) (
PARTITION p0 VALUES LESS THAN ( DATE'2000-01-01' )
);

alter table CLIENTS_EMPLOYEES
add constraint PK_STG_CL_EMP_ID primary key (STG_CL_EMP_ID);

create table COUNTRIES (
country_name  VARCHAR2(40), region_id NUMBER,
int_date  DATE default SYSDATE not null, source VARCHAR2(20),
created_date  DATE default SYSDATE, country_id  VARCHAR2(20),
stg_country_id NUMBER generated always as identity
)
PARTITION BY RANGE (int_date) INTERVAL ( INTERVAL '1' DAY ) (
PARTITION p0 VALUES LESS THAN ( DATE'2000-01-01' )
);

alter table COUNTRIES
add constraint PK_STG_COUNTRY_ID primary key (STG_COUNTRY_ID);

create table DEPARTMENTS (
department_name VARCHAR2(30), manager_id  NUMBER(6),
location_id NUMBER(4),
int_date  DATE default SYSDATE not null,
source  VARCHAR2(20), created_date  DATE default SYSDATE, department_id NUMBER,
stg_department_id NUMBER generated always as identity
)
PARTITION BY RANGE (int_date) INTERVAL ( INTERVAL '1' DAY ) (
PARTITION p0 VALUES LESS THAN ( DATE'2000-01-01' )
);

alter table DEPARTMENTS
add constraint PK_STG_DEPARTMENT_ID primary key (STG_DEPARTMENT_ID);

create table EMPLOYEES (
first_name  VARCHAR2(20),
last_name VARCHAR2(25),
email VARCHAR2(25),
phone_number  VARCHAR2(20), hire_date DATE,
job_id  VARCHAR2(10),
salary  NUMBER, commission_pct NUMBER, manager_id NUMBER(6), department_id
NUMBER(4),
int_date  DATE default SYSDATE not null, source VARCHAR2(20),
created_date  DATE default SYSDATE, employee_id NUMBER,
stg_employee_id NUMBER generated always as identity, valid_to DATE
)
PARTITION BY RANGE (int_date) INTERVAL ( INTERVAL '1' DAY ) (
PARTITION p0 VALUES LESS THAN ( DATE'2000-01-01' )
);

alter table EMPLOYEES

add constraint PK_STG_EMPLOYEE_ID primary key (STG_EMPLOYEE_ID); create table
EMP_MONTHLY_SALARY
(
pay_date  DATE,
```

```
emp_id  NUMBER, salary_type VARCHAR2(10),
created_date DATE default SYSDATE,
int_date  DATE default SYSDATE not null, source VARCHAR2(20),
amount  VARCHAR2(20),
sal_id  NUMBER,
stg_sal_id  NUMBER generated always as identity
)
PARTITION BY RANGE (int_date) INTERVAL ( INTERVAL '1' DAY ) (
PARTITION p0 VALUES LESS THAN ( DATE'2000-01-01' )
);

alter table EMP_MONTHLY_SALARY
add constraint PK_STG_SAL_ID primary key (STG_SAL_ID);

create table IMPDP_CLIENTS (
cli_id  INTEGER,
nm_cl VARCHAR2(30),
sr_cl VARCHAR2(30),
dob DATE,
gen CHAR(1),
email VARCHAR2(50),
cl_valid_from DATE, location_id NUMBER, cl_valid_to DATE
)
PARTITION BY RANGE (int_date) INTERVAL ( INTERVAL '1' DAY ) (
PARTITION p0 VALUES LESS THAN ( DATE'2000-01-01' )
);

create table IMPDP_CLI_EMP (
id  NUMBER not null, cl_id NUMBER,
emp_id NUMBER
)
PARTITION BY RANGE (int_date) INTERVAL ( INTERVAL '1' DAY ) (
PARTITION p0 VALUES LESS THAN ( DATE'2000-01-01' )
);

create table JOBS (
job_title VARCHAR2(35), min_salary  NUMBER(6), max_salary NUMBER(6),
int_date  DATE default SYSDATE not null, source VARCHAR2(10) not null, created_date
DATE default SYSDATE,
job_id  VARCHAR2(20),
stg_job_id  NUMBER generated always as identity
)
PARTITION BY RANGE (int_date) INTERVAL ( INTERVAL '1' DAY ) (
PARTITION p0 VALUES LESS THAN ( DATE'2000-01-01' )
)
PARTITION BY RANGE (int_date) INTERVAL ( INTERVAL '1' DAY ) (
PARTITION p0 VALUES LESS THAN ( DATE'2000-01-01' )
);

alter table JOBS
add constraint PK_STG_JOB_ID primary key (STG_JOB_ID);

create table JOB_HISTORY (
start_date  DATE,
end_date  DATE,
job_id  VARCHAR2(10),

department_id NUMBER(4),
int_date  DATE default SYSDATE not null,
source  VARCHAR2(10) default 'HR_DEPT' not null, created_date DATE default SYSDATE,
employee_id NUMBER,
stg_job_hst_id NUMBER generated always as identity
)
PARTITION BY RANGE (int_date) INTERVAL ( INTERVAL '1' DAY ) (
PARTITION p0 VALUES LESS THAN ( DATE'2000-01-01' )
);

alter table JOB_HISTORY
```

```
add constraint PK_STG_JOB_HST_ID primary key (STG_JOB_HST_ID);

create table LOCATIONS (
street_address VARCHAR2(40), postal_code VARCHAR2(50), city VARCHAR2(30),
state_province VARCHAR2(25), country_id CHAR(10 BYTE),
int_date  DATE default SYSDATE not null, source VARCHAR2(30) not null, created_date
DATE default SYSDATE, location_id NUMBER,
stg_loc_id  NUMBER generated always as identity
)
PARTITION BY RANGE (int_date) INTERVAL ( INTERVAL '1' DAY ) (
PARTITION p0 VALUES LESS THAN ( DATE'2000-01-01' )
);

alter table LOCATIONS
add constraint PK_STG_LOC_ID primary key (STG_LOC_ID);

create table REGIONS (
region_name VARCHAR2(25 BYTE),
int_date  DATE default SYSDATE not null, source VARCHAR2(15 BYTE) not null,
created_date  DATE default SYSDATE, region_id NUMBER,
stg_region_id NUMBER generated always as identity
) PARTITION BY RANGE (int_date)
INTERVAL ( INTERVAL '1' DAY ) (
ARTITION p0 VALUES LESS THAN ( DATE'2000-01-01' )
);

alter table REGIONS
add constraint PK_STG_REGION_ID primary key (STG_REGION_ID);
```

Příloha 3: Datový model HR a tabulky ve vstupní vrstvě

```
PROCEDURE proc_STAGE_imp_UTLF (p_dir IN VARCHAR2,
                               p_file IN VARCHAR2,
                               p_input_date IN VARCHAR2,
                               p_source IN VARCHAR2)
AS
  fhandle  utl_file.file_type;
  l_line  VARCHAR2(4000);
  l_regions_trans  stage_hr.regions%ROWTYPE;
  l_countries_trans  stage_hr.countries%ROWTYPE;
  l_locations_trans  stage_hr.locations%ROWTYPE;
  l_departments_trans  stage_hr.departments%ROWTYPE;
BEGIN
  fhandle := UTL_FILE.FOPEN(p_dir, p_file, 'R');
 IF p_file = 'countries.csv'
 THEN
  LOOP
    BEGIN

      UTL_FILE.GET_LINE(fhandle, l_line);
      l_countries_trans.country_id:= SUBSTR(l_line, 1, INSTR(l_line,',', 1, 1) -1);
      l_countries_trans.country_name:=
                        SUBSTR(
                             l_line,
                             INSTR(l_line,',', 1, 1) + 1,
                             INSTR(l_line, ',', 1, 2) - INSTR(l_line, ',', 1, 1) -1
                               );

      l_countries_trans.region_id:=
           to_number(
                     SUBSTR(
                            l_line,
                            INSTR(l_line,',', 1, 2) + 1,
                            INSTR(l_line, ',', 1, 3) - INSTR(l_line, ',', 1, 2) -1
                            )
                       );
```

```
      l_countries_trans.int_date:= to_date(p_input_date, 'dd.mm.yyyy');
      l_countries_trans.source:= p_source;
      INSERT INTO stage_hr.countries (country_id, country_name, region_id,
int_date, SOURCE)
      VALUES(l_countries_trans.country_id,
             l_countries_trans.country_name,
             l_countries_trans.region_id,
             l_countries_trans.int_date,
             l_countries_trans.source);
      l_countries_trans:= NULL;
      EXCEPTION WHEN No_Data_Found
      THEN EXIT;
    END;
  END LOOP;

  ELSIF p_file = 'departments.csv'
  THEN LOOP
    BEGIN

      UTL_FILE.GET_LINE(fhandle, l_line);
      l_departments_trans.department_id:= SUBSTR(l_line, 1,
                                          INSTR(l_line,',', 1, 1) -1);

      l_departments_trans.department_name:=
                               SUBSTR(
                                      l_line,
                                      INSTR(l_line,',', 1, 1) + 1,
                                      INSTR(l_line, ',', 1, 2) -
                                      INSTR(l_line, ',', 1, 1) -1
                                     );

   l_departments_trans.manager_id:=
                 to_number(
                           SUBSTR(
                                  l_line,
                                  INSTR(l_line,',', 1, 2) + 1,
                                  INSTR(l_line, ',', 1, 3) -
                                  INSTR(l_line, ',', 1, 2) -1
                                   )
                            );

       l_departments_trans.location_id:=
                              SUBSTR(
                                     l_line,
                                     INSTR(l_line,',', 1, 3) + 1,
                                     INSTR(l_line, ',', 1, 4) -
                                     INSTR(l_line, ',', 1, 3) -1
                                    );

       l_countries_trans.int_date:= to_date(p_input_date, 'dd.mm.yyyy');
       l_departments_trans.source:= p_source;
       INSERT INTO stage_hr.departments(department_id, department_name,
manager_id, location_id, int_date, SOURCE)
       VALUES(l_departments_trans.department_id,
              l_departments_trans.department_name,
              l_departments_trans.manager_id,
              l_departments_trans.location_id,
              l_countries_trans.int_date,
              l_departments_trans.source);
       l_countries_trans:= NULL;
    EXCEPTION WHEN No_Data_Found
    THEN EXIT;
    END;
  END LOOP;

  ELSIF p_file = 'locations.csv'
  THEN
    LOOP
```

```
      BEGIN
        UTL_FILE.GET_LINE(fhandle, l_line);
        l_locations_trans.location_id:= SUBSTR(l_line, 1,
                                               INSTR(l_line,',', 1, 1) -1);
        l_locations_trans.street_address:=
                             SUBSTR(
                                     l_line,
                                     INSTR(l_line,',', 1, 1) + 1,
                                     INSTR(l_line, ',', 1, 2) -
                                     INSTR(l_line, ',', 1, 1) -1
                                   );

        l_locations_trans.postal_code:=
                            SUBSTR(
                                    l_line,
                                    INSTR(l_line,',', 1, 2) + 1,
                                    INSTR(l_line, ',', 1, 3) -
                                    INSTR(l_line, ',', 1, 2) -1
                                  );
        l_locations_trans.city:=
                          SUBSTR(
                                  l_line,
                                  INSTR(l_line,',', 1, 3) + 1,
                                  INSTR(l_line, ',', 1, 4) -
                                   INSTR(l_line, ',', 1, 3) -1
                                );

        l_locations_trans.state_province:=
                                  SUBSTR(
                                          l_line,
                                          INSTR(l_line,',', 1, 4) + 1,
                                          INSTR(l_line, ',', 1, 5) -
                                           INSTR(l_line, ',', 1, 4) -1
                                        );

        l_locations_trans.country_id:=
                            SUBSTR(
                                    l_line,
                                    INSTR(l_line,',', 1, 5) + 1,
                                    INSTR(l_line, ',', 1, 6) -
                                    INSTR(l_line, ',', 1, 5) -1
                                  );

        l_locations_trans.int_date:= to_date(p_input_date, 'dd.mm.yyyy');
        l_locations_trans.source:= p_source;
        INSERT INTO stage_hr.locations(location_id, street_address, postal_code,
city, state_province, country_id, int_date, source)
        VALUES(l_locations_trans.location_id,
               l_locations_trans.street_address,
               l_locations_trans.postal_code,
               l_locations_trans.city,
               l_locations_trans.state_province,
               l_locations_trans.country_id,
               l_locations_trans.int_date,
               l_locations_trans.source);

    EXCEPTION WHEN No_Data_Found
    THEN EXIT;
    END;
   END LOOP;

  ELSIF p_file = 'regions.csv'
    THEN
      LOOP
        BEGIN

          UTL_FILE.GET_LINE(fhandle, l_line);
          l_regions_trans.region_id:= SUBSTR(l_line, 1,INSTR(l_line,',', 1, 1) -1);
```

```
          l_regions_trans.region_name:=
                                SUBSTR(
                                        l_line,
                                        INSTR(l_line,',', 1, 1) + 1,
                                        INSTR(l_line, ',', 1, 2) -
                                         INSTR(l_line, ',', 1, 1) -1
                                      );

          l_regions_trans.int_date:= to_date(p_input_date, 'dd.mm.yyyy');
          l_regions_trans.source:= p_source;
          INSERT INTO stage_hr.regions(region_id, region_name, int_date, SOURCE)
          VALUES(l_regions_trans.region_id,
                 l_regions_trans.region_name,
                 l_regions_trans.int_date,
                 l_regions_trans.source);

        EXCEPTION WHEN No_Data_Found
        THEN EXIT;
        END;
      END LOOP;


  END IF;

UTL_FILE.FCLOSE(fhandle);
COMMIT;
END proc_STAGE_imp_UTLF;
/
```

příloha 4: Procedura *proc_STAGE_imp_UTLF* (UTL_FILE)

```
PROCEDURE proc_LOAD_EXT_DATA (p_filename  IN VARCHAR2,
                              p_input_date IN VARCHAR2,
                              p_source  IN VARCHAR2)
AS

  l_job_history_trans stage_hr.job_history%ROWTYPE;
  l_jobs_trans  stage_hr.jobs%ROWTYPE;
  l_employees_trans  stage_hr.employees%ROWTYPE;
  l_emp_monthly_salary_trans stage_hr.emp_monthly_salary%ROWTYPE;


BEGIN

IF p_filename = 'job_hst.xml.gz'
  THEN
    INSERT INTO stage_hr.job_history(employee_id,
                                     start_date,
                                     end_date,
                                     job_id,
                                     department_id,
                                     int_date,
                                     SOURCE)
    SELECT v."employee_id",
           to_date(v."start_date", 'dd.mm.yyyy'),
           to_date(v."end_date", 'dd.mm.yyyy'),
           v."job_id",
           v."department_id",
           to_date(p_input_date, 'dd.mm.yyyy'),
           p_source
    FROM view_ext_job_history v;
```

```
ELSIF p_filename = 'jobs.xml.gz'
  THEN
    INSERT INTO stage_hr.jobs(job_id,
                              job_title,
                              min_salary,
                              max_salary,
                              int_date,
                              source)
    SELECT s."job_id",
           s."job_title",
           s."min_salary",
           s."max_salary",
           to_date(p_input_date, 'dd.mm.yyyy'),
           p_source
    FROM stage_hr.view_ext_jobs s;

ELSIF p_filename = 'employees.csv.gz'
  THEN
    INSERT INTO stage_hr.employees(first_name,
                                   last_name,
                                   email,
                                   phone_number,
                                   hire_date,
                                   job_id,
                                   salary,
                                   commission_pct,
                                   manager_id,
                                   department_id,
                                   int_date,
                                   SOURCE,
                                   employee_id,
                                   valid_to)
    SELECT e.first_name,
           e.last_name,
           e.email,
           e.phone_number,
           to_date(e.hire_date, 'dd.mm.yyyy'),
           e.job_id,
           e.salary,
           e.commission_pct,
           e.manager_id,
           e.department_id,
           to_date(p_input_date, 'dd.mm.yyyy'),
           p_source,
           e.employee_id,
           to_date(e.valid_to, 'dd.mm.yyyy')
    FROM stage_hr.zip_ext_employees e;

ELSIF p_filename = 'emp_month_sal.csv.gz'
  THEN
  INSERT INTO stage_hr.emp_monthly_salary(sal_id,
                                          pay_date,
                                          emp_id,
                                          salary_type,
                                          amount,
                                          int_date,
                                          source)
SELECT ms.sal_id,
       to_date(ms.pay_date, 'dd.mm.yyyy'),
       ms.emp_id,
```

```
       ms.salary_type,
       LTRIM(ms.amount),
       to_date(p_input_date, 'dd.mm.yyyy'),
       p_source
FROM stage_hr.zip_ext_emp_monthly_salary ms;

END IF;

END;
/


CREATE TABLE zip_job_history_xml_ext (
doc1 CLOB
)
ORGANIZATION EXTERNAL (
TYPE oracle_loader
DEFAULT DIRECTORY EXEC_AREA ACCESS PARAMETERS (
RECORDS XMLTAG ("history") PREPROCESSOR EXEC_AREA:'gunzip.ksh' FIELDS
NOTRIM
MISSING FIELD VALUES ARE NULL
)) REJECT LIMIT UNLIMITED;
CREATE TABLE zip_jobs_xml_ext (
doc1 CLOB
)
ORGANIZATION EXTERNAL (
TYPE oracle_loader
DEFAULT DIRECTORY EXEC_AREA ACCESS PARAMETERS (
RECORDS XMLTAG ("job")
PREPROCESSOR EXEC_AREA:'gunzip.ksh' FIELDS NOTRIM
MISSING FIELD VALUES ARE NULL
)
)

REJECT LIMIT UNLIMITED;

Externí tabulka zip_ext_employees

CREATE TABLE zip_ext_employees (
first_name varchar2(20 byte),
last_name varchar2(25 byte),
email varchar2(25 byte),
Phone_number varchar2(20 byte),
hire_date  varchar2(20 byte),
job_id  varchar2(20 byte),
salary  varchar2(25 byte),
commission_pct varchar2(25 byte),
manager_id    varchar2(25 byte),
department_id varchar2(25 byte),
int_date varchar2(30 byte),
SOURCE  varchar2(20 byte),
employee_id varchar2(25 byte),
valid_to    varchar2(25 byte)
)
ORGANIZATION EXTERNAL (
TYPE ORACLE_LOADER
DEFAULT DIRECTORY EXEC_AREA ACCESS PARAMETERS
(
RECORDS DELIMITED BY NEWLINE PREPROCESSOR EXEC_AREA:'gunzip.ksh'
FIELDS TERMINATED BY ',' (employee_id,
                          first_name,
```

```
                            last_name,
                            email,
                            Phone_number,
                            hire_date,
                            job_id,
                            salary,
                            commission_pct,
                            manager_id,
                            department_id,
                            valid_to,
                            int_date,
                            SOURCE
)

)
LOCATION('employees.csv.gz')
)
REJECT LIMIT UNLIMITED;


Externí tabulka zip_ext_emp_monthly_salary

CREATE TABLE zip_ext_emp_monthly_salary (
    SAL_ID varchar2(25 byte),
    PAY_DATE varchar2(25 byte),
    EMP_ID varchar2(25 byte),
    SALARY_TYPE varchar2(25 byte),
    AMOUNT NUMBER,
    INT_DATE varchar2(25 byte),
    SOURCE varchar2(25 byte)
    )
ORGANIZATION EXTERNAL (
TYPE ORACLE_LOADER
DEFAULT DIRECTORY EXEC_AREA ACCESS PARAMETERS
(
RECORDS DELIMITED BY NEWLINE PREPROCESSOR EXEC_AREA:'gunzip.ksh' FIELDS
TERMINATED BY ',' (SAL_ID,
PAY_DATE, EMP_ID, SALARY_TYPE, AMOUNT, INT_DATE, SOURCE)
)
LOCATION('emp_month_sal.csv.gz')
)
REJECT LIMIT UNLIMITED;


CREATE VIEW view_ext_jobs AS(
SELECT xt.*
FROM zip_jobs_xml_ext EXTERNAL MODIFY(LOCATION('job.xml.gz')) x,
     XMLTABLE('/job' PASSING XMLTYPE(x.doc1) COLUMN
     "job_id"  VARCHAR2(50 BYTE) PATH  'job_id',
    "job_title"  VARCHAR2(50 BYTE) PATH 'job_title',
    "min_salary" VARCHAR2(55 BYTE) PATH  'min_salary',
    "max_salary" VARCHAR2(55 BYTE) PATH 'max_salary',
    "int_date"  VARCHAR2(50 BYTE) PATH  'int_date',
    "source"  VARCHAR2(50 BYTE) PATH  'source') xt

);


CREATE VIEW view_ext_job_history AS ( SELECT xt.*
FROM zip_job_history_xml_ext EXTERNAL MODIFY(LOCATION('job_hst.xml.gz')) x,
XMLTABLE('/history' PASSING XMLTYPE(x.doc1)
```

```
COLUMNS
"employee_id"   NUMBER  PATH  'Employee_id',
"start_date"   VARCHAR2(50 BYTE) PATH 'start_date',
"end_date"  VARCHAR2(55 BYTE) PATH 'end_date',
"job_id"      VARCHAR2(55 BYTE) PATH 'job_id',
"department_id"  VARCHAR2(50 BYTE) PATH 'department_id',
"int_date" VARCHAR2(50 BYTE) PATH    'int_date',
"source"    VARCHAR2(50 BYTE) PATH
'source') xt);
```

příloha 5: Procedura Proc_LOAD_EXT_DATA, Exeterní tabulky a Views

```
PROCEDURE proc_LOAD_IMPDP_DATA(p_filename  IN VARCHAR2,
                               p_input_date IN VARCHAR2,
                               p_source  IN VARCHAR2)
AS
BEGIN
  IF p_filename = 'imp_cli.dmp'
    THEN

      INSERT INTO stage_hr.clients(cli_id,
                                   nm_cl,
                                   sr_cl,
                                   dob,
                                   gen,
                                   email,
                                   valid_from,
                                   location_id,
                                   int_date,
                                   SOURCE,
                                   valid_to)
      SELECT s.cli_id,
             s.nm_cl,
             s.sr_cl,
             s.dob,
             s.gen,
             s.email,
             s.cl_valid_from,
             s.location_id,
             to_date(p_input_date, 'dd.mm.yyyy'),
             p_source,
             s.cl_valid_to
      FROM stage_hr.impdp_clients s;

    DELETE FROM stage_hr.impdp_clients;
ELSIF p_filename = 'imp_cli_emp.dmp'
 THEN

    INSERT INTO stage_hr.clients_employees(client_id,
                                           employee_id,
                                           int_date,
                                           source,
                                           cl_emp_id)
    SELECT e.cl_id,
           e.emp_id,
           to_date(p_input_date, 'dd.mm.yyyy'),
           p_source,
           e.id
    FROM stage_hr.impdp_cli_emp e;

    DELETE FROM stage_hr.impdp_cli_emp;
    END IF;

EXCEPTION
WHEN OTHERS THEN RAISE;
END proc_LOAD_IMPDP_DATA;
```

impdp userid=stage_HR/hr@PDB2 directory=EXEC_AREA dumpfile=imp_cli_emp.dmp remap_schema=hr:stage_hr remap_table=clients_employees:impdp_cli_emp content=data_only

impdp userid=stage_HR/hr@PDB2 directory=EXEC_AREA dumpfile=imp_cli.dmp remap_schema=hr:stage_hr remap_table=clients:impdp_clients content=data_only

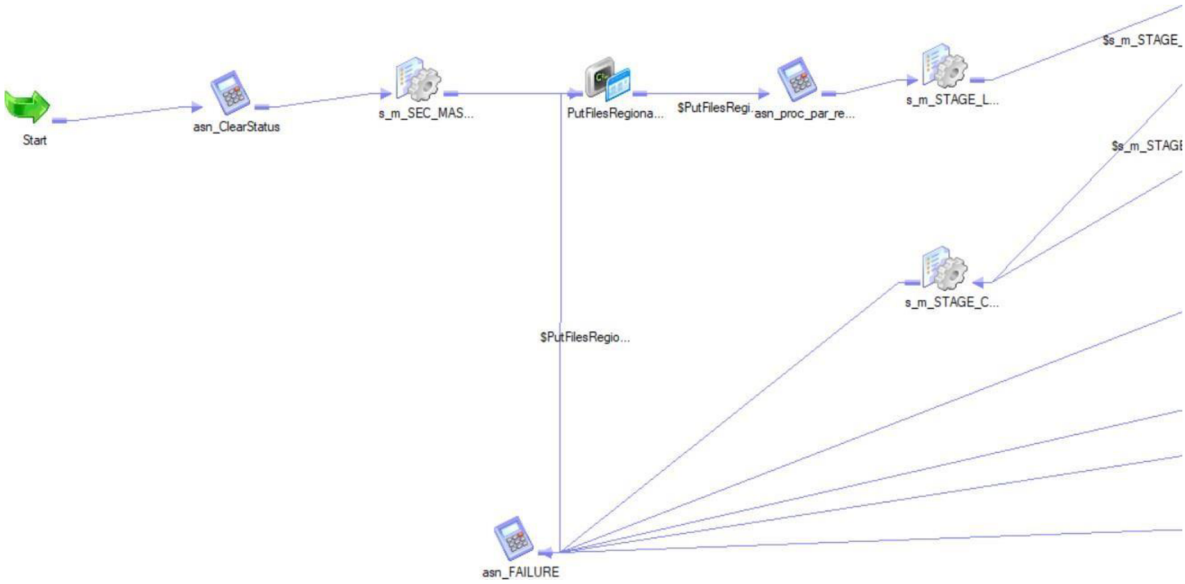
Příloha 6: *Proc_LOAD_IMPDP_DATA*, Datapumpové skripty

```
PROCEDURE  proc_STAGE_copy_delete_file(p_old_ora_dir  VARCHAR2,
                                       p_new_ora_dir VARCHAR2,
                                       p_filename  VARCHAR2 DEFAULT NULL,
                                       p_source  VARCHAR2 DEFAULT NULL)
AS

  l_filename VARCHAR2(90);

BEGIN
--copy file from LAND_AREA_REG to EXEC_AREA
IF p_old_ora_dir = 'LAND_AREA_REG' AND p_filename = 'ALL' AND p_source =
'REGIONAL_DEPT'
  THEN
    FOR rec IN (SELECT s.file_name
                FROM stage_hr.md_stage_src_files s
                WHERE s.source = p_source)
     LOOP

      UTL_FILE.FRENAME (p_old_ora_dir,
      rec.file_name, p_new_ora_dir, rec.file_name, TRUE);

    END LOOP;

ELSIF p_old_ora_dir = 'LAND_AREA_HR' AND p_filename = 'ALL' AND p_source =
'HR_DEPT'
  THEN
    FOR rec IN (SELECT s.file_name
                FROM stage_hr.md_stage_src_files s
                WHERE s.source = p_source)
     LOOP

      UTL_FILE.FRENAME (p_old_ora_dir,
      rec.file_name, p_new_ora_dir, rec.file_name, TRUE);
    END LOOP;

ELSIF p_old_ora_dir = 'LAND_AREA_COMM' AND p_source = 'COMMERCIAL_DEPT'
  THEN
    FOR rec IN (SELECT s.file_name
                FROM stage_hr.md_stage_src_files s
                WHERE s.source = p_source
                  AND s.file_name = p_filename)
     LOOP

      UTL_FILE.FRENAME (p_old_ora_dir,
      rec.file_name, p_new_ora_dir, rec.file_name, TRUE);
    END LOOP;
--copy file from EXEC_AREA to HST_AREA
ELSIF p_new_ora_dir LIKE 'HST_AREA%'
  THEN
      l_filename:= to_char(SYSDATE, 'YYMMDD')|| '_hst_' ||p_filename;

      UTL_FILE.FRENAME (p_old_ora_dir,
      p_filename, p_new_ora_dir, l_filename, TRUE);

--copy file from EXEC_AREA to ERR_AREA
```

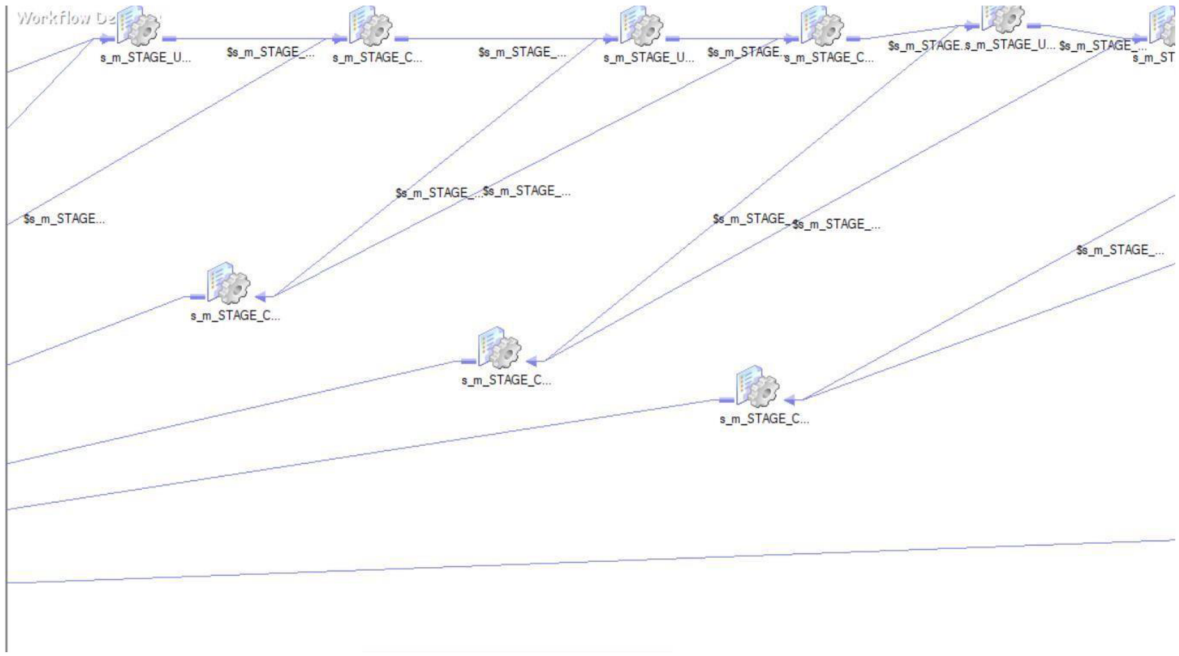
```
ELSIF p_new_ora_dir LIKE 'ERR_AREA%'
  THEN
    l_filename:= to_char(SYSDATE, 'YYMMDD')|| '_err_' ||p_filename;
UTL_FILE.FRENAME (p_old_ora_dir,
    p_filename, p_new_ora_dir, l_filename, TRUE);
END IF;

END proc_STAGE_copy_delete_file;
/
```

příloha 7: proc_STAGE_copy_delete_file

Začátek

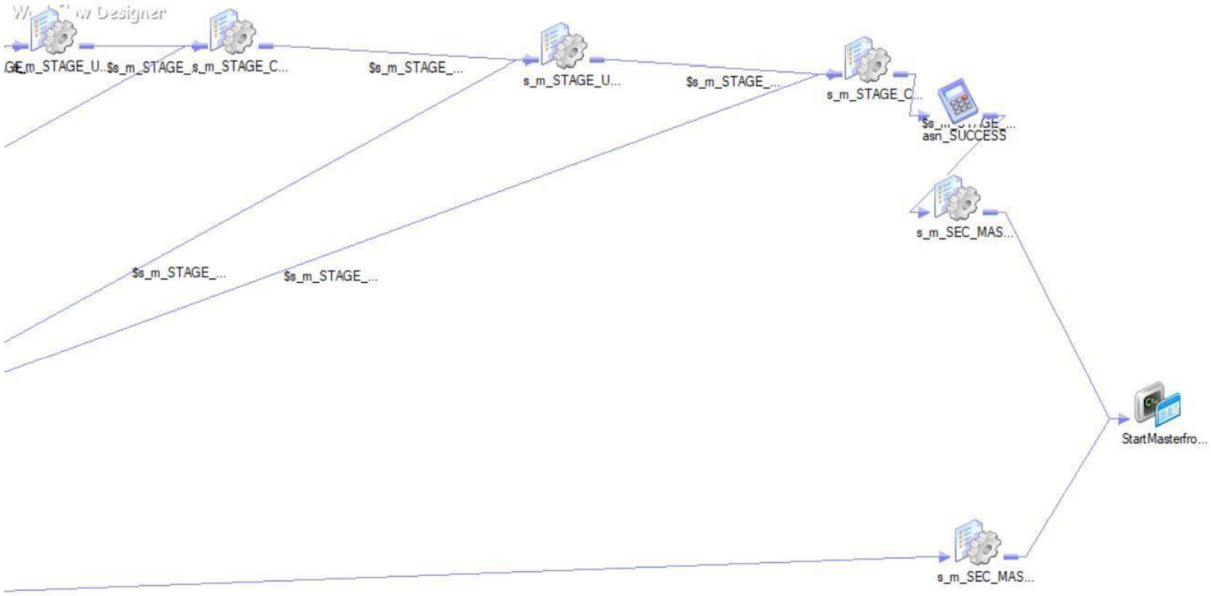


Prostřední část



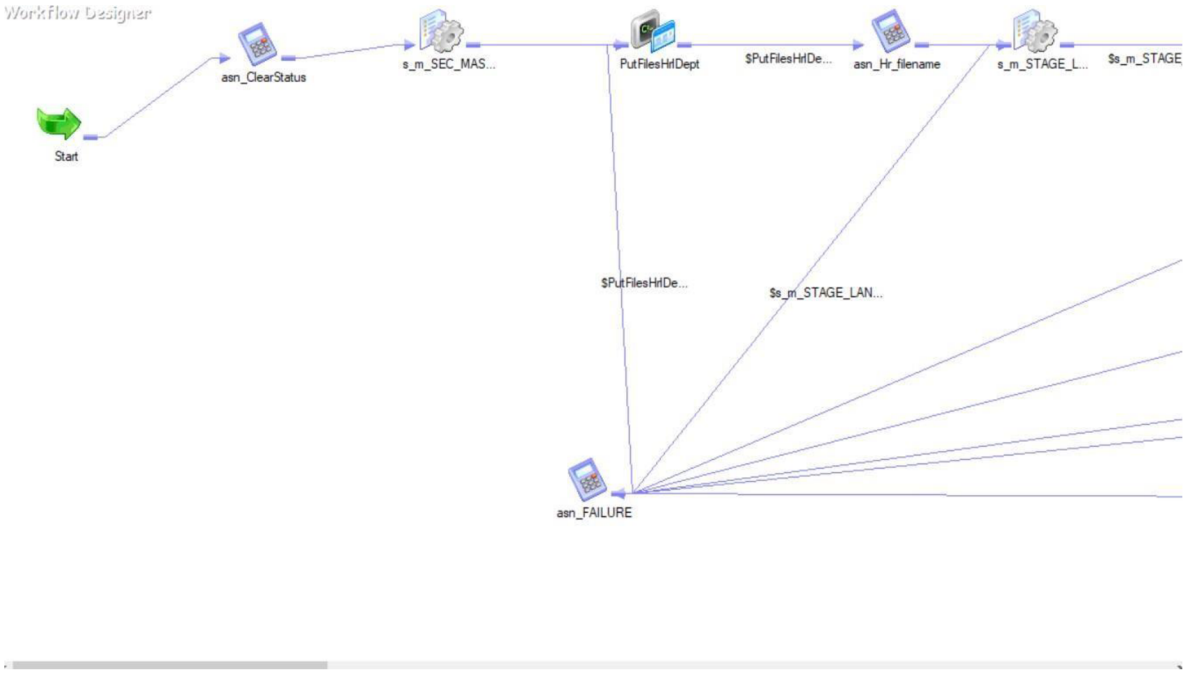
..

Konec



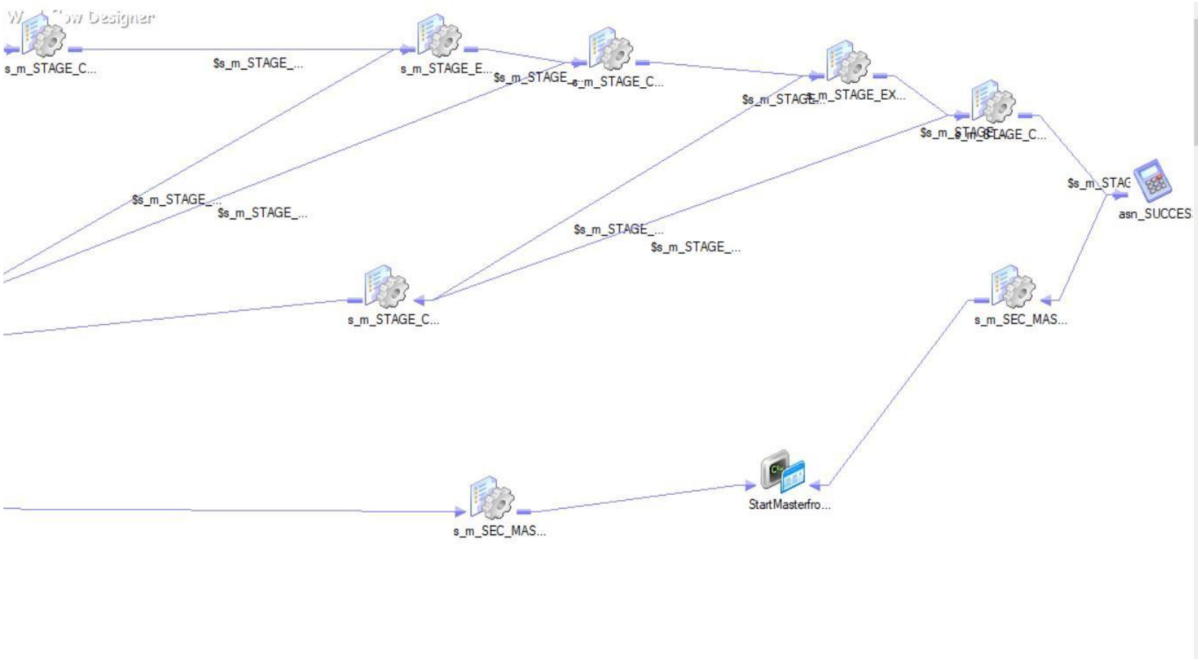
Příloha 8: wf_STAGE_UTL_FILES

Začátek



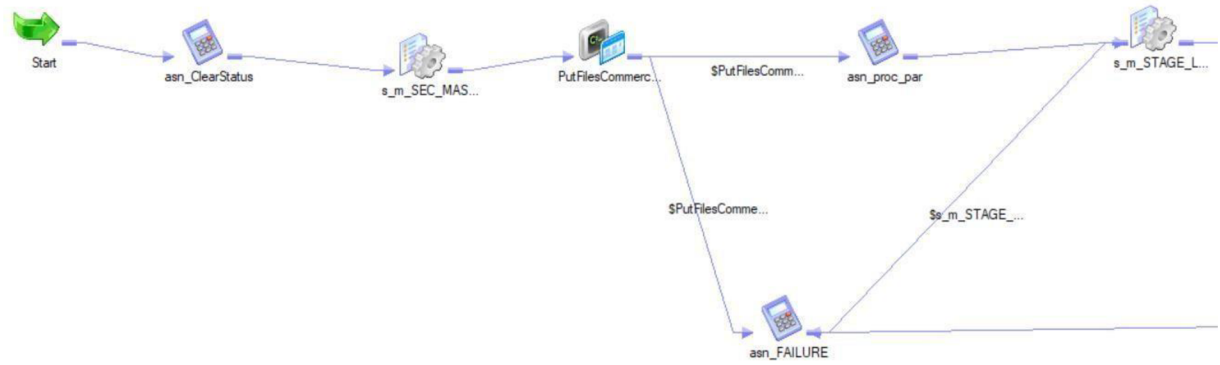
Prostřední část

Konec

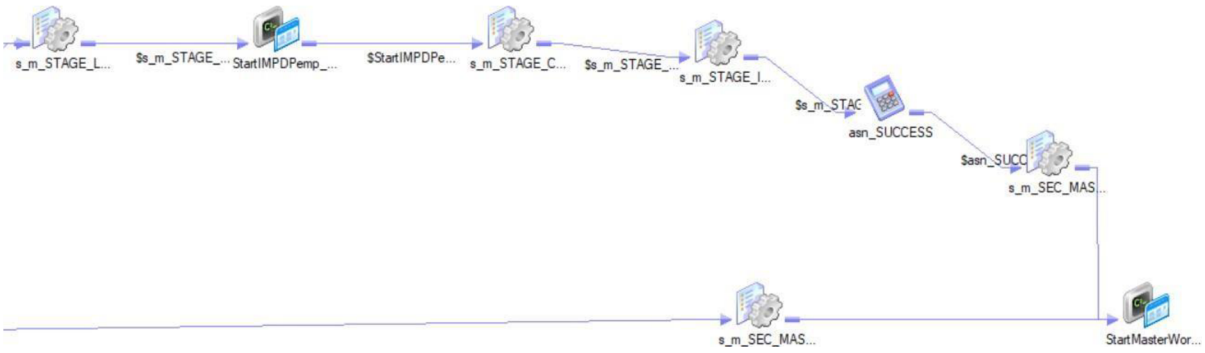


Příloha 9: wf_STAGE_EXT_FILES

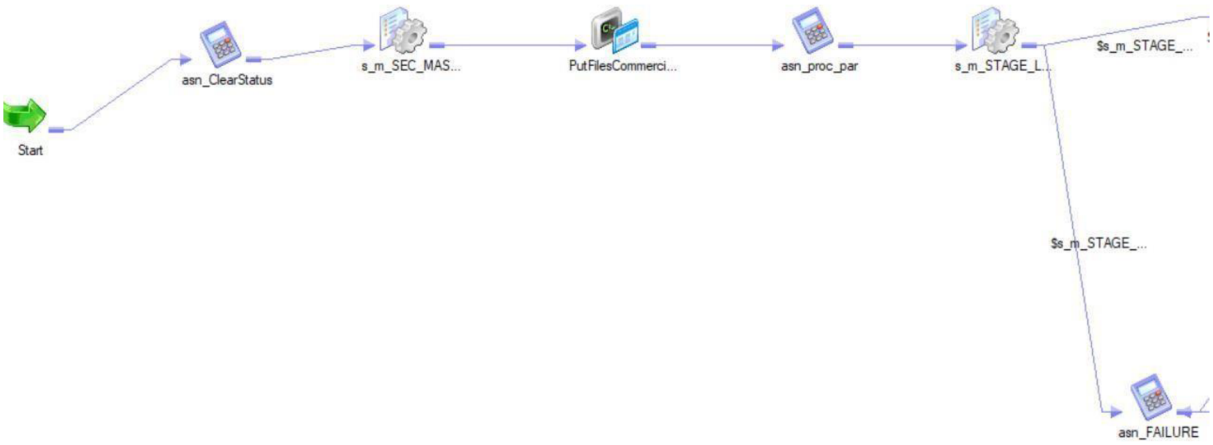
Začátek wf_STAGE_IMPDP_FILES_CLI_EMP



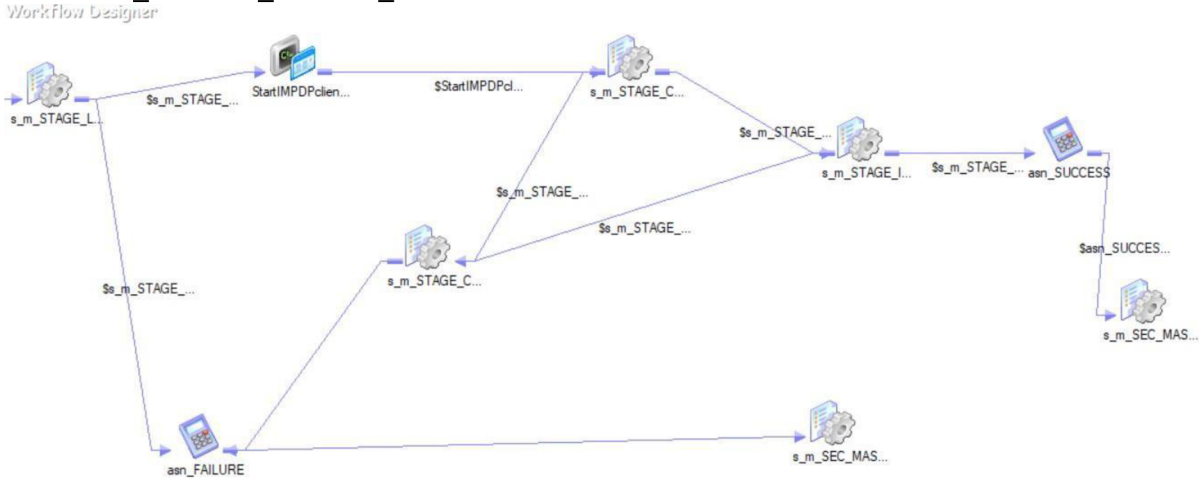
Konec wf_STAGE_IMPDP_FILES_CLI_EMP



Začátek wf_STAGE_IMPDP_CLIENTS



Konec wf_STAGE_IMPDP_CLIENTS



Příloha 10: wf_STAGE_IMPDP_FILES_CLI_EMP a wf_STAGE_IMPDP_CLIENTS

PUT_FILES_COMMERCIAL_DEPT_CLI_EMP.bat
pscp -pw oracle C:\Sources_files\IMPDPSRC\imp_cli_emp.dmp
oracle@192.168.56.104:/home/oracle/landing_area/COMMERCIAL_
DEPT/ exit

PUT_FILES_COMMERCIAL_DEPT_CLIENTS.bat
pscp -pw oracle C:\Sources_files\IMPDPSRC\imp_cli.dmp
oracle@192.168.56.104:/home/oracle/landing_area/COMMERCIAL_
DEPT/ exit

PUT_FILES_REGIONAL_DEPT.bat
C:\pscp -pw oracle C:\Sources_files\UTLSRC*.csv
oracle@192.168.56.104:/home/oracle/landing_area/REGIONAL_
DEPT/

PUT_FILES_HR_DEPT.bat
pscp -pw oracle C:\Sources_files\EXTSRC*.csv.gz
C:\Sources_files\EXTSRC*.xml.gz
oracle@192.168.56.104:/home/oracle/landing_area/HR_DEPT/

příloha 11: Batch skripty