Univerzita Hradec Králové

Fakulta informatiky a managementu

SQL databáze a jejich synchronizační komponenty

BAKALÁŘSKÁ PRÁCE

Autor: Radek Vancl
Studijní obor: Aplikovaná Informatika

Vedoucí práce: Ing. Tomáš Nacházel, Ph.D.

Hradec Králové Srpen 2023

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne _____ _____

Podpis

Děkuji vedoucímu bakalářské práce za trpělivost a poskytnutí inspirace i cenných rad při psaní.

Anotace

Tato bakalářská práce se zabývá analýzou a možným vylepšením v praxi použité synchronizační komponenty celopodnikového operačního datového skladu.

Součástí bakalářské práce je rovněž průzkum aktuálních trendů v rámci synchronizace dat a databázových serverových řešení.

Annotation

This bachelor thesis deals with the analysis and possible improvement of used in practice synchronization component of enterprise-wide operational data store.

Part of the bachelor thesis is also research of the current trends of data synchronization and database server solutions.

Obsah

Seznam Obrázků

Seznam Tabulek

Seznam Zkratek

1	.NET	Microsoft .NET Framework
2	Ad-hoc	Pro tento jednotlivý případ
3	API	Application programming interface
4	ASP.NET	Active server pages network enabled technologies
5	AVRO	Apache Avro
6	BLOB	Binary large object
7	C#	C Sharp
8	CDC	Change data capture
9	CFG	Configuration (table)
10	CLOB	Character large object
11	Config	Configuration
12	CPU	Central processing unit
13	D, Def	Definition
14	DB	Database
15	DBMS	Database management systems
16	DDL	Data definition language
17	DML	Data manipulation language
18	DMV	Dynamic management views
19	ETL	Extract transform load
20	HB	Heart Beat
21	Hrs	Hours
22	ID	Identification
23	JDBC	Java database connectivity
24	MEM	Memory

25	MS	Microsoft
26	NFEL	Interní název databáze NFEL
27	NS	Name server record
28	ODS	Operational data store
29	Pcs	Process
30	PcsId	Proces ID
31	Pcss	Processes
32	PL	Procedural language
33	PM4	Process Manager 4
34	Prod	Production
35	Pub/Sub	Publish–subscribe vzor zpráv
36	RDBMS	Relational database management system
37	realtime, near-realtime	V téměř/réalném čase
38	REPL	Read–eval–print loop
39	REST	Representational state transfer
40	Run-time	Doba běhu
41	SC	StartCount – počet spuštění
42	SGA	System global area
43	SQL	Structured query language
44	SQLSP	Sequential least squares programming
45	SSIS	SQL Server Integration Services
46	Tempdb	Temporary DB
47	T-SQL	Transact SQL
48	UDT	User-defined data type
49	UI	User interface
50	V	View

51	VBA	Visual Basic for Applications
52	Workaround	Náhradní řešení
53	WPF	Windows presentation foundation

1 Úvod

V současné době informačního věku má efektivní manipulace s daty klíčový význam pro zajištění plynulosti podnikových procesů a úspěchu organizací. Databáze se staly základním kamenem moderního zpracování informací, poskytujícím spolehlivý způsob ukládání, organizace a správy dat. Se zvyšujícími nároky na velikost, komplexitu, a hlavně dostupnost dat však vzniká potřeba synchronizace těchto informací mezi nejrůznějšími databázovými systémy a aplikacemi.

Tato práce bude zaměřena na analýzu a možné vylepšení v praxi použité synchronizační komponenty celopodnikového operačního datového skladu. Součástí bakalářské práce je rovněž průzkum aktuálních trendů v rámci synchronizace dat a databázových serverových řešení.

V kapitole SQL databáze budou vybrána nejpoužívanější databázová serverová řešení v rámci soukromé sféry, budou dále srovnána s ohledem na jejich rezervovaná slova a datové typy v závislosti na jejich synchronizaci. V návaznosti na tuto část, bude dále představen pojem synchronizační komponenta, metody synchronizace a obecně technologie načítání změn do celofiremního datového skladu. Pátá kapitola bude rovněž obsahovat analýzu existujícího řešení v podniku s více než 250 zaměstnanci.

Poslední kapitola zhodnotí efektivitu řešení synchronizační komponenty s rešerší za účelem možné inovace.

Pro tuto bakalářskou práci budou použity interní materiály firmy s podmínkou, že nebudou použity v celém rozsahu, aby nebyly zneužitelné třetí stranou. Tato podmínka obsahuje i dovětek, že nebude použit konkrétní název firmy.

2 Cíle práce

Cílem práce je zmapování aktuálních možností řešení problematiky synchronizačních komponent pro středně velké až velké podniky, což může sloužit jako vhled do vnitřních procesů firem.

Práce následně nastíní, jakým směrem by se případně mohl vývoj těchto systémů ubírat. Zároveň může sloužit jako motivace pro další sebevzdělávání studentů či jako podnět pro případnou úpravu osnov stávajících předmětů.

3 SQL databáze

Je-li zmiňován databázový server, je tím myšlen softwarový produkt s primární funkcí získávání a ukládání dat dle požadavků jiných softwarových aplikací, které mohou běžet buďto na stejném počítači, nebo na jiném výpočetním zařízení v síti.

Databázových serverových řešení existuje hned několik, a tak je potřeba si nejdříve stanovit jejich nejpopulárnější zástupce v rámci soukromé sféry.

Následující kapitola má za úkol představit dva hlavní zástupce těchto systémů.

3.1 Způsob výběru databází

Výběr je založen na výzkumu publikovaném na webové stránce https://db-engines.com/ . Webová stránka je vytvořena a udržována firmou Solid IT sídlící na adrese Dr.-Kühne-Gasse 5, 1230 Vídeň, Rakousko.

Solid IT o sobě uvádí, že:

"Solid IT je rakouská IT konzultační společnost se zvláštním zaměřením na vývoj softwaru, poradenství a školení pro databázově orientované aplikace."

Skóre DB-Engines se vypočítává na základě následujících faktorů:

- Počtu výsledků ve vyhledávačích Google, Bing a Yandex.
- Frekvence vyhledávání v Google Trends.
- Četnosti technických diskusí na známých stránkách pro otázky a odpovědi související s IT Stack Overflow a DBA Stack Exchange.
- Počtu pracovních nabídek na Indeed a Simply Hired.
- Počtu profilů v profesionálních sítích včetně LinkedIn a Upwork.
- Zmínkách na sociální síti Twitter. [2]

3.2 Výsledek výběru databází

Na základě těchto faktorů se na předních 10 příčkách dlouhodobě drží tato databázová serverová řešení:

Rank	DBMS	Database Model	Score
1.	Oracle	Relational, Multi-model	1260.80
2.	MySQL	Relational, Multi-model	1202.85
3.	Microsoft SQL Server	Relational, Multi-model	944.96
4.	PostgreSQL	Relational, Multi-model	618.00
5.	MongoDB	Relational, Multi-model	477.66
6.	Redis	Relational, Multi-model	176.39
7.	IBM Db2	Relational, Multi-model	157.23
8.	Elasticsearch	Search engine, Multi-model	155.08
9.	Microsoft Access	Relational	146.50
10.	SQLite	Relational	138.87

Tabulka 1 - Výsledek výběru databází, žebříček [1]

Z tabulky vyplývá, že nejpopulárnější databází na světě dle žebříčku DB-Engine je Oracle. Oracle je následován databázemi MySQL, Microsoft SQL Server, PostgreSQL a MongoDB. [2]

Vzhledem k tomu, že MySQL není primárně určena pro korporátní počítačové clustery, budou porovnávána pouze řešení na 1. a 3. příčce, tedy Oracle a Microsoft SQL Server.

3.3 Oracle

Oracle vyvinutý společností Oracle Corporation je dle DB-Engines nejpopulárnější relační databázový systém (RDBMS). Nejen, že se Oracle řadí mezi RDBMS, ale poskytuje také funkce pro Cloud, Document Store, Graph DBMS, úložiště klíč-hodnota, BLOG a úložiště PDF.

Nedávno (Q2 2022) Oracle oznámil vývoj autonomní funkce, která umožní, aby databáze byla inteligentní a samosprávná.

Aktuální verze Oracle Database je 18c. [3]

3.3.1 Zařazení databáze Oracle

Databáze Oracle se řadí mezi relační databáze. Relační databáze ukládají data v tabulkové formě řádků a sloupců. Sloupec databázové tabulky představuje atributy entity a řádky tabulky ukládají záznamy. RDBMS, který implementuje objektově orientované funkce jako jsou uživatelem definované typy, dědičnost a polymorfismus, se nazývá objektově-relační systém správy databází (ORDBMS). Oracle Database rozšířila relační model na objektově relační model, což umožňuje ukládat složité business modely v relační databázi.

3.3.2 Přístup k datům u Oracle

Oracle Database je první databáze navržená pro podnikové tabulkové výpočty (Enterprise grid computing). Enterprise grid computing vytváří velké fondy modulárních úložišť a serverů. S touto architekturou může být každý nový systém rychle zajišťován z fondu zdrojů.

Není potřeba mít extrémně výkonný hardware při pracovní zátěži v pracovní „špičce“, protože kapacitu lze podle potřeby snadno přidat nebo přerozdělit z fondů zdrojů. Jednou z charakteristik RDBMS databáze je, že má oddělené logické struktury a fyzické struktury. Protože fyzické a logické struktury jsou oddělené, fyzické úložiště dat lze spravovat bez ovlivnění přístupu k logickým strukturám úložiště. [3]

3.3.3 PL/SQL

Oracle je přístupná pouze prostřednictvím klientského programu a využívá pro to jazyk PL/SQL, který je při komunikaci s rozhraním tohoto klientského programu k databázi Oracle. Uživatelé databází Oracle označují paměťovou strukturu na straně serveru jako SGA (System Global Area). SGA obvykle uchovává informace o mezipaměti, jako jsou datové vyrovnávací paměti, příkazy SQL a informace o uživateli. Databáze se také skládá z protokolů, které obsahují historii transakcí.

V databázi Oracle je schéma databáze kolekce logických datových struktur nebo objektů schématu. Uživatel databáze vlastní schéma databáze, které má identický název jako je jeho uživatelské jméno. Objekty schématu jsou uživatelem vytvořené struktury, které přímo odkazují na data v databázi. Databáze podporuje mnoho typů objektů schématu, z nichž nejdůležitější jsou tabulky a indexy. Objekt schématu je jeden typ databázového objektu. Některé databázové objekty, jako jsou profily a role, nejsou umístěny ve schématech. [3]

3.4 Microsoft SQL Server

Microsoft SQL Server (dále jen MS SQL Server) je systém pro správu relačních databází vyvinutý společností Microsoft.

Společnost Microsoft uvádí na trh hned několik různých edic MS SQL Server, které jsou zaměřeny na různorodé cílové skupiny a pracovní zatížení od malých aplikací na jednom počítači až po velké internetové aplikace s mnoha paralelními připojeními. [4]

3.4.1 Představení datových typů MS SQL Server

Datové úložiště je kolekce tabulek se zadanými sloupci. MS SQL Server podporuje různé typy dat, včetně primitivních typů, jako je Integer, Float, Decimal, Char (včetně znakových řetězců), Varchar (znakové řetězce s proměnnou délkou), binárních dat – pro nestrukturované Binary Large Objects (dále jako BLOBs), Text (pro textová data) a další. A dále například pro zaokrouhlení plovoucí desetinnou čárkou (dále jako floats) na celá čísla používá buď symetrické aritmetické zaokrouhlování nebo symetrické zaokrouhlení dolů v závislosti na argumentech:

Např. SELECT Round (4.5, 0) vrací číslo 5. [5]

3.4.2 Uživatelsky definované datové typy

Microsoft SQL Server také umožňuje definovat a používat uživatelsky definované datové typy (dále jen jako UDT). Také zpřístupňuje statistiky serveru jako virtuální tabulky a pohledy (nazývané Dynamic Management Views nebo zkráceně DMV). Kromě tabulek může databáze obsahovat také další objekty včetně pohledů, uložených procedur, indexů a omezení spolu s protokolem transakcí.

Databáze MS SQL Server může obsahovat maximálně 231 objektů a může zahrnovat více souborů na úrovni operačního systému s maximální velikostí souboru 2 na 60 bajtů (1 exabajt). [6]

3.4.3 Souběžnost zamykání a pesimistický mechanismus řízení

SQL Server umožňuje více klientům používat stejnou databázi současně. Jako takový potřebuje řídit souběžný přístup ke sdíleným datům tak, aby byla zajištěna integrita dat, když více klientů aktualizuje stejná data nebo když se klienti pokouší číst data, která jsou v procesu změny jiným klientem. SQL Server poskytuje dva režimy řízení souběžnosti: pesimistickou souběžnost a optimistickou souběžnost.
Při použití pesimistického řízení souběžnosti SQL Server řídí souběžný přístup pomocí zámků.

Zámky mohou být sdílené nebo exkluzivní. Exkluzivní zámek poskytuje uživateli výhradní přístup k datům – dokud je zámek držen, žádný jiný uživatel nemá k datům přístup. Při čtení některých dat se používají takzvané sdílené zámky. Z dat uzamčených sdíleným zámkem může číst více uživatelů, ale žádný z nich nezíská exkluzivní zámek. Uživatel by tak musel čekat na uvolnění všech sdílených zámků.[7]

3.4.4 Zámky dle úrovně granularity

Zámky lze použít na různých úrovních granularity tzn. na celé tabulky, stránky nebo dokonce na základě řádků v tabulkách. U indexů může být buď na celém indexu nebo na listech indexu. Úroveň granularity, která se má použít, je definována pro každou databázi správcem databáze. I když jemnozrnný uzamykací systém (fine grained locking system) umožňuje více uživatelům používat tabulku nebo index současně, má vyšší nároky na hardwarové prostředky, což snižuje výpočetní výkon.

SQL Server také obsahuje dvě lehčí řešení vzájemného vyloučení, hrubozrnné zámky (coarse grain locks) a spinlocky, které jsou méně robustní než fine grained locks, ale jsou méně náročné na zdroje. SQL Server je používá pro DMV a další prostředky, které obvykle nejsou zaneprázdněny. SQL Server monitoruje všechna pracovní vlákna, která získávají zámky, aby bylo zajištěno, že neskončí v uváznutí. V případě, že k tomu dojde, SQL Server přijme nápravná opatření, která v mnoha případech spočívají v zastavení jednoho z vláken zapletených do uváznutí a vrácení transakce, kterou zahájil.

Pro implementaci zamykání obsahuje SQL Server Správce zámků. Správce zámků udržuje v paměti tabulku, která spravuje databázové objekty a zámky, pokud existují, spolu s dalšími metadaty o zámku. Přístup k jakémukoli sdílenému objektu je zprostředkován správcem zámku, který buď udělí přístup ke zdroji nebo jej zablokuje. [7]

3.4.5 Optimistický mechanismus řízení

SQL Server také poskytuje optimistický mechanismus souběžného řízení, který je podobný víceverzovému řízení souběžnosti používanému v jiných databázích. Mechanismus umožňuje vytvoření nové verze řádku při každé aktualizaci, na rozdíl od přepsání, řádek je navíc identifikován pomocí ID transakce, která vytvořila verzi řádku.

Staré i nové verze řádku jsou uloženy a udržovány, ačkoli staré verze jsou přesunuty z databáze do systémové databáze označené jako Tempdb. Když je řádek v procesu aktualizace, žádné další požadavky nejsou blokovány (na rozdíl od zamykání), ale jsou prováděny na starší verzi řádku. Pokud je druhým požadavkem příkaz k aktualizaci, výsledkem budou dvě různé verze řádků – obě budou uloženy v databázi a identifikovány pomocí příslušných ID transakcí. [7]

3.4.6 Načítání dat a programovatelnost

Hlavním způsobem získávání dat z databáze MS SQL Server je jejich dotazování. Dotaz je vyjádřen pomocí varianty SQL nazvané T-SQL což je dialekt, který MS SQL Server sdílí se Sybase SQL Server kvůli jejich společné minulosti. Dotaz deklarativně určuje, co má být načteno. Zpracovává jej dotazovací procesor, který určuje posloupnost kroků, které budou nutné k získání požadovaných dat.

Posloupnost akcí nezbytných k provedení dotazu se nazývá plán dotazů. Může existovat několik způsobů, jak zpracovat stejný dotaz. Například pro dotaz, který obsahuje příkaz spojení a výběr provedení spojení na obou tabulkách a následné provedení výběru na výsledcích, by poskytlo stejný výsledek jako výběr z každé tabulky a následné provedení spojení, ale výsledkem by byly různé plány provádění. V takovém případě SQL Server zvolí plán, u kterého se očekává, že přinese výsledky v co nejkratším čase. To se nazývá optimalizace dotazů a provádí ji samotný procesor dotazů. [8]

3.4.7 Optimalizace dotazů

MS SQL Server obsahuje nákladově orientovaný optimalizátor dotazů, který se snaží optimalizovat náklady, pokud jde o zdroje potřebné k provedení dotazu. Po zadání dotazu se optimalizátor dotazů podívá na schéma databáze, statistiky databáze a zatížení systému v daném okamžiku. Poté rozhodne, která sekvence má přistupovat k tabulkám uvedeným v dotazu, která sekvence má provádět operace a jakou přístupovou metodu použít pro přístup k tabulkám. Pokud má například tabulka přidružený index rozhodne, zda by měl být index použit nebo ne: pokud je index ve sloupci, který není pro většinu sloupců jedinečný, systém nemusí pro přístup k datům index vůbec použít. Zatímco souběžné provádění je z hlediska celkového využitého času procesoru nákladnější, jejich provádění může být ve skutečnosti rozloženo mezi procesory, což může znamenat, že dotaz bude zpracován rychleji.

Jakmile je pro dotaz vygenerován plán dotazů, je dočasně uložen do mezipaměti. Pro další vyvolání identického dotazu se poté využije plán, který je v této mezipaměti uložen. Nepoužívané plány jsou po uplynutí systémově určené doby vyřazeny.[9]

3.4.8 Uložené procedury

SQL Server také umožňuje definovat uložené procedury (Stored Procedures). Uložené procedury jsou parametrizované T-SQL dotazy, které jsou uloženy na samotném serveru a nevydává je klientská aplikace, jako je tomu u obecných dotazů.

Uložené procedury mohou přijímat hodnoty zaslané klientem jako vstupní parametry a odesílat zpět výsledky jako výstupní parametry. Mohou volat definované funkce a další uložené procedury, včetně opakování stejné uložené procedury až do nastaveného limitu. Oproti jiným dotazům mohou uložené procedury selektivně poskytnout přístup a mají přidružený název, který se za běhu používá k následné transkripci do dotazů. Také proto, že kód nemusí být poslán pokaždé od klienta (protože je přístupný podle jména), procedura snižuje provoz v síti a znatelně zlepšuje výkon. Prováděcí plány pro uložené procedury jsou také dle potřeby ukládány do mezipaměti.[10]

3.4.9 T-SQL

T-SQL (Transact-SQL) je proprietární rozšíření procedurálního jazyka společnosti Microsoft pro SQL Server. Poskytuje instrukce REPL (Read-Eval-Print-Loop), které rozšiřují standardní instrukční sadu SQL pro instrukce Data Manipulation (DML) a Data Definition (DDL), včetně nastavení specifických pro SQL Server, zabezpečení a správy databázových statistik. Toto rozšíření zpřístupňuje klíčová slova pro operace, které lze provádět na serveru SQL Server, včetně vytváření a úprav databázových schémat, zadávání a úprav dat v databázi a také monitorování a správy samotného serveru.

Klientské aplikace, které využívají data nebo spravují server, využívají funkčnost SQL Serveru odesíláním dotazů a příkazů T-SQL, které jsou následně zpracovány serverem a výsledky (nebo chyby) jsou na to konto vráceny klientské aplikaci. Za tímto účelem také zpřístupňuje tabulky pouze pro čtení, ze kterých lze zjistit statistiky serveru.

Funkčnost správy systému je vystavena prostřednictvím systémově definovaných uložených procedur, které lze vyvolat z dotazů T-SQL k provedení operace správy. Pomocí T-SQL lze také servery navzájem propojit. Propojení serverů poté umožní jednomu dotazu zpracovávat operace prováděné na více serverech. [11]

3.5 *Rozdílnosti systémů v závislosti na synchronizaci*

Slova vyhrazená pro objekty schématu se mezi Oracle a Microsoft SQL Server liší. Mnoho rezervovaných slov Oracle je platným názvem objektu, popř. názvem sloupce v MS SQL Server.

3.5.1 Rezervovaná slova

DATE je kupříkladu rezervované slovo v Oracle, avšak není rezervovaným slovem v MS SQL Server. Proto žádný sloupec nemůže mít v Oracle název DATE, ale v MS SQL Server sloupec mít název DATE může. Použití vyhrazených slov jako názvů objektů schématu znemožňuje použití stejných názvů napříč databázemi. Uživatel by měl zvolit název objektu schématu, který je jedinečný dle velikosti písmen a alespoň jedné další charakteristiky, aby tak zajistil, že název objektu nebude rezervovaným slovem z žádné z databází. [12]

Zde lze již vidět tabulku rezervovaných slov pro obě řešení:

Oracle	**MS SQL Server**
Database	Database
Schema	Database and database owner (DBO)
Tablespace	Database
User	User
Role	Group/Role
Table	Table
Temporary tables	Temporary tables
Cluster	N/A
Column-level check constraint	Column-level check constraint
Column default	Column default
Unique key	Unique key or identity property for a column
Primary key	Primary key
Foreign key	Foreign key
Index	Non-unique index
PL/SQL Procedure	Transact-SQL (T-SQL) stored procedure
PL/SQL Function	T-SQL stored procedure
Packages	N/A
AFTER triggers	Triggers
BEFORE triggers	Complex rules
Triggers for each row	N/A
Synonyms	N/A
Sequences	Identity property for a column
Snapshot	N/A
View	View

Tabulka 2 - Rezervované názvy objektů, přehled [12]

3.5.2 Datové typy

Datové typy IMAGE a TEXT (Binary Large Objects)

Fyzické a logické metody ukládání dat IMAGE a TEXT dat se mezi Oracle a MS SQL Server liší. V případě MS SQL Server je v tabulce uložen pointer na IMAGE data nebo TEXT, zatímco data IMAGE nebo TEXT jsou uložena samostatně. Toto uspořádání umožňuje více sloupců IMAGE nebo TEXT dat v tabulce.

V Oracle mohou být IMAGE data uložena jako BLOB a pole typu TEXT mohou být uložena v poli typu Character Large Object (dále uváděn jako CLOB). Oracle umožňuje více BLOB a CLOB sloupců na tabulku. BLOBs a CLOBs mohou nebo nemusí být uloženy v tabulce v závislosti na jejich velikosti.

Pokud je sloupec MS SQL Server TEXT takový, že data nikdy nepřesahují 4000 bajtů, uživatel může převést sloupec typu CLOB na datový typ Oracle VARCHAR2. Tabulka Oracle může mít definováno více VARCHAR2 sloupců. Tato velikost TEXT dat je vhodná pro většinu aplikací. [12]

MS SQL Server	Oracle
INTEGER	NUMBER (10)
SMALLINT	NUMBER (6)
TINYINT	NUMBER (3)
REAL	FLOAT
FLOAT	FLOAT
BIT	NUMBER (1)
CHAR(n)	CHAR(n)
VARCHAR(n)	VARCHAR2(n)
TEXT	CLOB
IMAGE	BLOB
BINARY(n)	RAW(n)/BLOB
VARBINARY(n)	RAW(n)/BLOB
DATETIME	DATE
SMALL-DATETIME	DATE
MONEY	NUMBER (19,4)
NCHAR(n)	CHAR(n*2)
NVARCHAR(n)	VARCHAR(n*2)
SMALLMONEY	NUMBER (10,4)
TIMESTAMP	NUMBER
SYSNAME	VARCHAR2(30) and VARCHAR2(128) respectively

Tabulka 3 Rozdílnosti datových typů, přehled [12]

MS SQL Server a Uživatelsky definované typy dat

Toto vylepšení SQL pro Microsoft SQL Server je specifické pro T-SQL a umožňuje uživatelům definovat a pojmenovávat své vlastní datové typy, které doplňují systémové datové typy. UDT lze použít jako datový typ pro libovolný sloupec v databázi. Na UDT lze poté navázat výchozí hodnoty a pravidla (kontrolní omezení), která se na jednotlivé sloupce těchto UDT automaticky aplikují. [12]

4 Synchronizační komponenta

Synchronizační komponenta je součást systému, která umožňuje synchronizaci dat mezi různými zdroji nebo systémy. Hlavním účelem synchronizační komponenty je zajistit konzistenci a aktuálnost dat mezi různými místy a zajišťovat, aby změny provedené v jednom zdroji byly zrcadleny nebo propagovány do jiných zdrojů.

Synchronizační komponenta může být použita například pro:

A. **Synchronizaci mezi databázemi:** Zajišťuje, že data uložená v různých databázích jsou aktuální a konzistentní. To je často důležité v distribuovaných systémech nebo v případech, kdy je třeba zrcadlit data na zálohovacích serverech.
B. **Synchronizaci mezi aplikacemi:** Umožňuje různým aplikacím sdílet a aktualizovat data tak, aby byla vždy aktuální a dostupná pro všechny aplikace.
C. **Synchronizaci mezi cloudovými a lokálními systémy:** Zajišťuje, že data uložená v cloudových systémech jsou synchronizována s lokálními databázemi nebo aplikacemi.
D. **Synchronizaci mezi zařízeními:** Umožňuje synchronizaci dat mezi různými zařízeními jako jsou počítače, mobilní telefony nebo tablety, aby uživatelé mohli pracovat s aktuálními daty na různých zařízeních.

[13]

Synchronizační komponenty mohou využívat různé technologie a přístupy, které jsou podrobněji popsány v následující kapitole. Jejich cílem je zajištění integrity, konzistence a aktuálnost dat mezi různými částmi systému nebo mezi různými systémy například v Operativním datovém skladu (dále jako ODS).

Celofiremní ODS je prostředek, který organizace používají k ukládání, správě a analýze velkého množství dat z různých zdrojů. Načítání změn do celofiremního ODS je důležitý proces, který umožňuje udržovat aktuálnost a relevantnost dat v ODS. Existuje několik metod načítání změn do celofiremního ODS.

4.1 *Metody Synchronizace*

A. **Inkrementální (dále také jako přírůstkové) načítání:** Při této metodě se načítají pouze nové nebo změněné datové sady od posledního načtení. To lze implementovat pomocí časových razítek, identifikátorů změn nebo jiných příznaků. Přírůstkové načítání je efektivní, když je třeba minimalizovat zatížení zdrojových systémů a urychlit proces aktualizace dat v ODS. [14]

B. **Inkrementální (přírůstkové) načítání pomocí logu:** Některé databáze a systémy umožňují sledování logů transakcí a zaznamenávání změn. Tato metoda může být účinná pro zachycení podrobných změn dat. [14]

C. **Batch (dále jako dávkové načítání):** U této metody jsou změny pravidelně zpracovávány ve velkých dávkách. To může být vhodné pro organizace, které mají nižší požadavky na zpracování dat v reálném čase a mohou se spolehnout na pravidelné aktualizace dat. [15]

D. **Change Data Capture (CDC):** CDC je technika, která zachycuje změny v základních zdrojích dat v reálném čase a přenáší tyto změny do cílového úložiště (v tomto případě do celopodnikového ODS). Při této metodě jsou změny dat sledovány a automaticky přenášeny do cílového skladu. [16]

E. **Rozhraní API a webové služby:** Pokud má podnik externí zdroje dat, může k načtení nových nebo změněných dat použít rozhraní API a webové služby. To se často používá k integraci s externími systémy. [16]

F. **Replikace(kopírování) dat:** Tato metoda zahrnuje replikaci dat mezi různými datovými úložišti, aby bylo možné udržovat konzistentní verze dat napříč různými systémy. [17]

G. **Mechanismy zálohování a obnovy:** V případě potřeby provést úplnou obnovu dat z primárního zdroje. [18]

Je důležité zvolit způsob načítání změn, který nejlépe vyhovuje potřebám a požadavkům organizace, včetně požadavků na aktuálnost dat, dostupnost zdrojových systémů a výkonnostní cíle.

4.2 Technologie načítání změn do celofiremního ODS

K načítání změn do celopodnikového ODS se používají různé technologie a systémy, které umožňují zachytit, transformovat a přenést změny z různých zdrojů do cílového úložiště. Tyto systémy a technologie zahrnují, ale nejsou omezeny na:

A. **ETL (Extract, Transform, Load):** Nástroje ETL se široce používají k extrahování dat z různých zdrojů, jejich transformaci do požadovaného formátu a načtení do cílového úložiště, jako je například právě celopodnikový ODS. Mezi oblíbené nástroje ETL patří informační systémy jako Informatica, Microsoft SQL Server Integration Services (SSIS) nebo Talend.[19]

B. **Databázové funkce:** Některé moderní databázové systémy poskytují funkce pro získávání změn nebo záznamů pomocí speciálních operací. PostgreSQL má například funkci logické replikace pro replikaci změn.[20]

C. **Vlastní skripty a procesy:** V některých případech mohou organizace vytvářet vlastní skripty a procesy pro načtení změn do celopodnikového ODS, pokud existují specifické požadavky na transformaci dat nebo zachycení změn.

H. **Nástroje Change Data Capture (CDC):** Nástroje CDC se specializují na zachycování změn v databázích a systémech v reálném čase a na přenos těchto změn do cílového skladu. Příkladem je Oracle GoldenGate, Microsoft SQL Server Change Data Capture nebo Apache Kafka pro streamování dat. [21]

I. **Platformy pro streamování a zasílání zpráv:** Platformy pro streamování dat a zasílání zpráv lze použít k zachycení změn a jejich odeslání do cílového úložiště. Kromě Apache Kafka lze zvážit i další platformy jako Amazon Kinesis, Apache Pulsar nebo Google Cloud Pub/Sub. [21]

J. **API a webové služby:** Pokud získáváte data z externích zdrojů, můžete přes API a webové služby získat nová nebo změněná data a následně je přenést do celopodnikového ODS. [21]

Výběr konkrétního systému závisí na požadavcích, technologickém zázemí a budgetu podniku. Je důležité zvážit faktory jako reálný čas, výkonnost, bezpečnost, kompatibilitu se zdrojovými systémy a celkovou architekturu datového toku při výběru správného systému pro načítání změn do celopodnikového ODS.

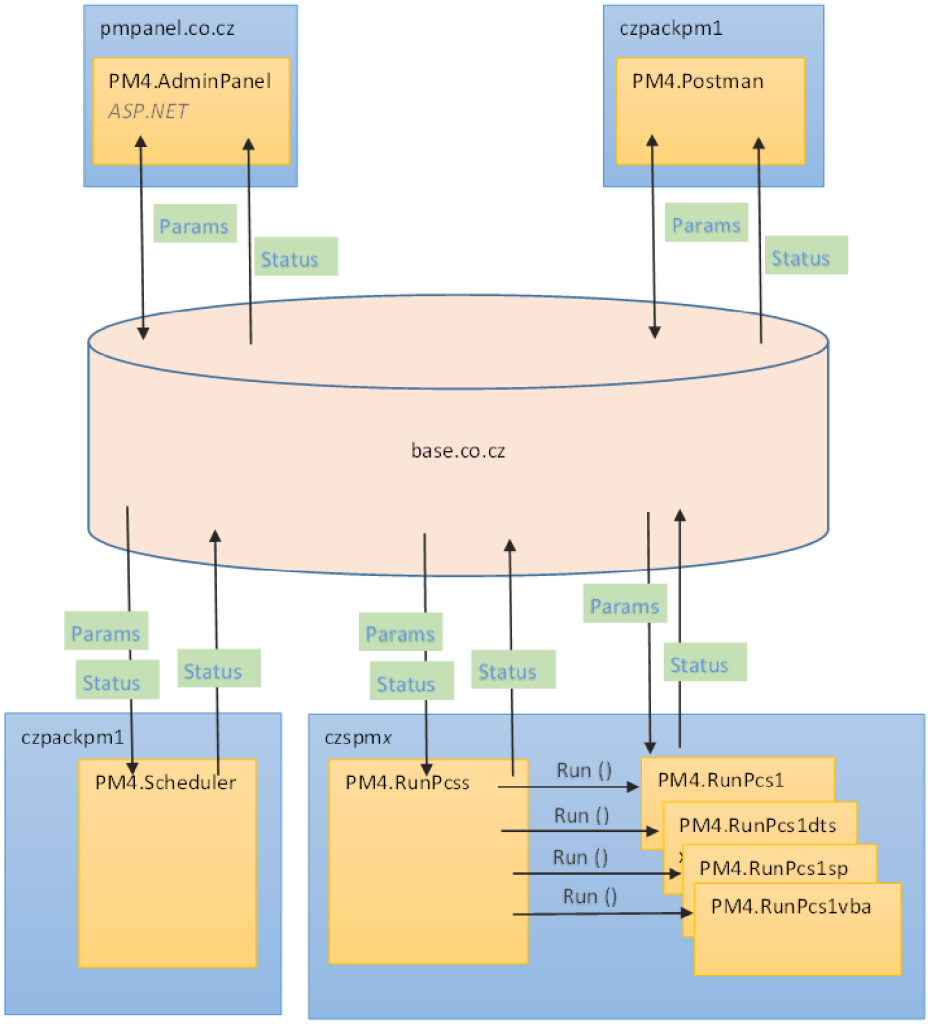
5 Analýza existujícího řešení v podniku

Na základě tématu práce, dále následuje představení již existující funkčního řešení celopodnikového ODS. O hladký chod řešení a jeho správu se stará Process Manager 4 (dále uváděn pod označením PM4), ten spouští jednotlivé procesy (dále jako pcs, popřípadě pcss).

Mezi procesy patří kromě ODS procesů také ostatní ETL procesy pro správu podnikových databází. Stávající řešení je designováno na hodinové až denní synchronizace databáze s ODS.

5.1 PM4

Jedná se o plánovač procesů, zvaný scheduler, psaný v ASP, .NET, Telerik&Kendo UI. Je používán jako interní nástroj pro management ETL procesů nad databázemi podniku.



Obrázek 1 - PM4, diagram [vlastní]

Systém se sestává z modulů

- PM4.Scheduler – Modul pro časové plánování a rozdělování Procesů
- PM4.RunPcss - Více modulů, realizují spouštění Procesů
- PM4.Postman – Modul zajišťující zasílání emailů
- PM4.AdminPanel – FE pro parametrizaci a sledování (webová aplikace)

5.1.1 Typy Procesů

A. C#
   - C# Proces
   - Proces vytvořený pro použití v PM4.
   - PM4 umí zachytit chyby Procesu (unmanaged exceptions), předávat proměnné, atributy, connection stringy, zapisovat do logů apod.

B. SSIS
   - Spuštění: Modul *PM4.RunPcsjs* spustí „obálku“ *PM4.RunPcs1dtsx*, která inicializuje a spouští vlastní .DTSX Package.
   - Obálka zajistí předávání Connection Strings a Variables.

C. Master
   - Master Proces
   - Master Proces obsahuje seznam Procesů, které postupně spouští.
   - Procesy mohou být jakéhokoliv typu.
   - Pro typ „Master“ platí výjimky:
     - Nelze spustit konfiguraci sestavenou tak, že tvoří smyčku (cyklický odkaz).
     - Spuštění Procesu je asynchronní, scheduler nekontroluje další činnost spuštěného master Procesu a v dalším kroku pokračuje na další položku v slave listu.

D. SQLSP
   - SQL Stored Procedure
   - Spuštění: Modul *PM4.RunPcsjs* spustí „obálku“ *PM4.RunPcs1sp*, která předá parametry a provede vlastní proceduru (Exec SP).

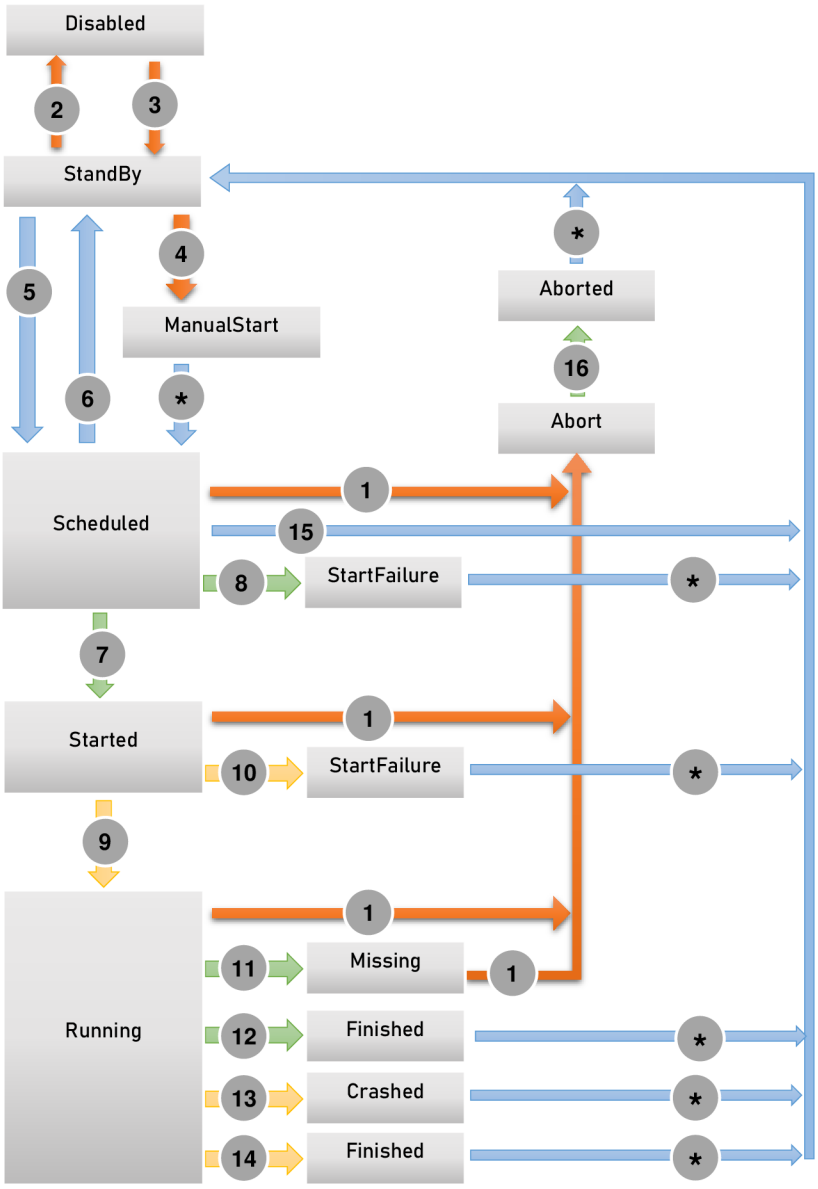
E. VBA
   - Spuštění: Modul *PM4.RunPcsjs* spustí „obálku“ *PM4.RunPcs1vba*, která spustí aplikaci Excel, otevře Workbook a v něm spustí postupně makra dle parametrizace.

5.1.2 Životní cyklus Procesu

Legenda

	Scheduler
	RunPcss
	RunBase / RunPcs1dtsx / RunPcs1sp / RunPcs1vba
	PM4.Admin

Tabulka 4 – Proces, legenda životního cyklu



Obrázek 2 – Proces, Životní cyklus[vlastní]

Podmínky

*		Vždy
1	FE	Abort (ikona)
2	FE	Disable (ikona)
3	FE	Enable (ikona)
4	FE	ManualStart (ikona)
5	Scheduler	Nyní> ManualStart, Nyní> NextSheduledRun (from Cron)
6	Scheduler	RunPcss Timeout Proces byl ve stavu Scheduled více než TimeoutScheduleTo-Running, viz kapitola 5.1
7	RunPcss	Package existuje, parametry jsou OK
8	RunPcss	Package neexistuje nebo má špatné parametry Schedule time> 120 sec
9	RunBase	Init OK
10	RunBase	Init failure Proces je ve stavu Started více než TimeoutStartedTo-Running, viz kapitola 5.2
11	RunPcss	Proces / task není v Process table serveru (mimo External Exe) po více než 3 kontroly (MissingCount> 2)
12	RunPcss	Task není v Process table serveru
13	RunBase	Exited with error code/Throw exception by code/Raised un-managed exception
14	RunBase	Finished in regular way with proper exit code
15	Scheduler	Scheduled Timeout Proces se nachází příliš dlouho ve stavu plánování, plánování je ukončeno. Možné příčiny: Pcs má špatně nastavené nodeid v definition, PM server není k dispozici nebo RunPcss na PM uzlu spadl.

16	RunPcss	Vždy V tomto přechodu ukončí běžící Proces (Kill na process nebo jeho obálku)

Tabulka 5 – Proces, podmínky životního cyklu

5.1.3 Dashboard (Default)

Zobrazení přehledu o stavu v následujících kategoriích:

A. Stav systému
B. Aktuálně běžící úlohy
C. Ukončené úlohy
D. Následující úlohy

A. Stav systému

PM Dashboard : PROD

Module	NodeId	NodeName	Before	Duration	CPU	MEM	Status	Run	SC
PM4.Scheduler	0	czpackpm1	9 sec	708 ms			Alive	hrs	6
PM4.RunPcss	0	czspmx1	8 sec	46 ms	49%	4672 MB	Alive	30 days	3
PM4.RunPcss	1	czspmx2	3 sec	31 ms	0%	4787 MB	Alive	19 days	4
PM4.Postman	0	czpackpm1	19 sec	93 ms			Alive	11 days	3
PM4.Trapender	0	czpackpm1	16 sec	0 ms			Alive	29 days	4

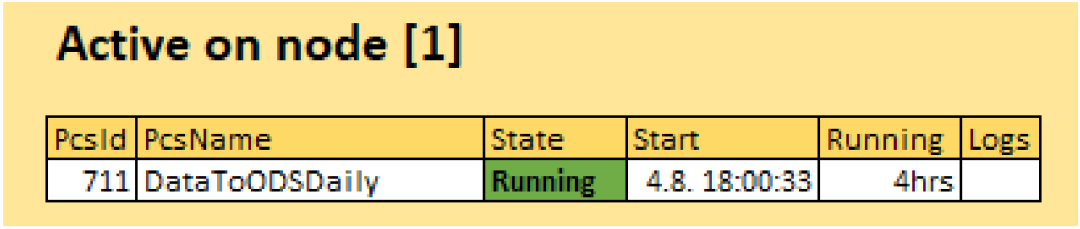
Obrázek 3 – Stav systému, přehled[vlastní]

Pole	Význam
Module	Jméno modulu – RunPcss / Scheduler
NodeId	Node ID … ID v rámci parametrizace systému PM4
Node-Name	Node name … jméno serveru
Last	Značka poslední činnosti
Before	Nyní – Last
Duration	Doba činnosti modulu [ms]
CPU	Zatížení procesoru, pouze pro RunPcss
MEM	Volná paměť před rozšířením stránkování, pouze pro RunPcss
Status	Stav modulu: DEAD Modul je nefunkční (neprobíhá cyklicky)

Alive	Modul pracuje
Run	Doba běhu modulu
SC	StartCount – počet spuštění

Tabulka 6 – Stav systému, popis přehledu

B. Aktuálně běžící úlohy



Active on node [1]

PcsId	PcsName	State	Start	Running	Logs
711	DataToODSDaily	Running	4.8. 18:00:33	4hrs	

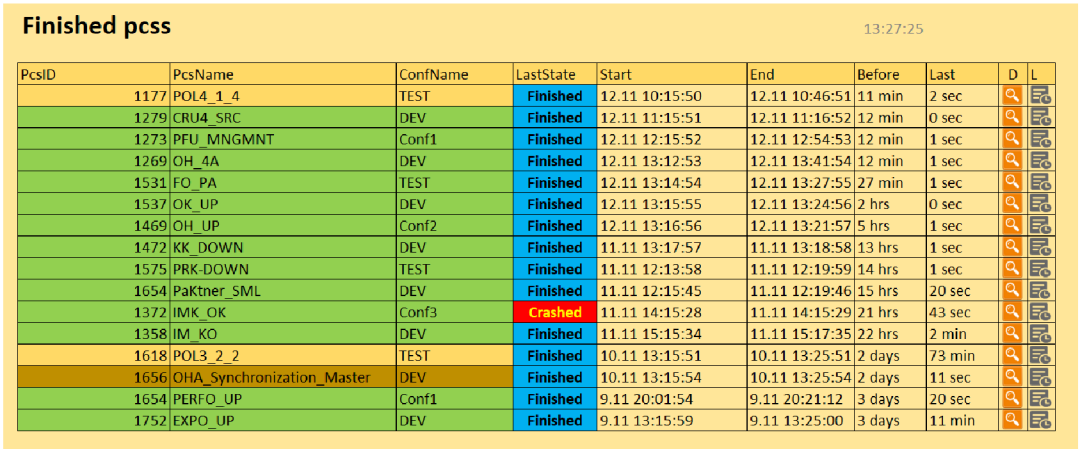
Obrázek 4 - Aktuálně běžící úlohy, přehled[vlastní]

Pro každý uzel s běžícím RunPcss zobrazuje:

Pole	Význam
PcsId	ID úlohy
PcsName	Jméno úlohy (zkrácené)
State	Stav, viz kapitola *5.1.2*
Start	Čas spuštění
Running	Čas nyní – Start
Logs	Odkaz na Logy – nastaví se filtr na PcsId a čas Start/Finish

Obrázek 7 – Aktuálně běžící úlohy, popis přehledu

C. Ukončené úlohy



Finished pcss 13:27:25

PcsID	PcsName	ConfName	LastState	Start	End	Before	Last	D	L
1177	POL4_1_4	TEST	Finished	12.11 10:15:50	12.11 10:46:51	11 min	2 sec		
1279	CRU4_SRC	DEV	Finished	12.11 11:15:51	12.11 11:16:52	12 min	0 sec		
1273	PFU_MNGMNT	Conf1	Finished	12.11 12:15:52	12.11 12:54:53	12 min	1 sec		
1269	OH_4A	DEV	Finished	12.11 13:12:53	12.11 13:41:54	12 min	1 sec		
1531	FO_PA	TEST	Finished	12.11 13:14:54	12.11 13:27:55	27 min	1 sec		
1537	OK_UP	DEV	Finished	12.11 13:15:55	12.11 13:24:56	2 hrs	0 sec		
1469	OH_UP	Conf2	Finished	12.11 13:16:56	12.11 13:21:57	5 hrs	1 sec		
1472	KK_DOWN	DEV	Finished	11.11 13:17:57	11.11 13:18:58	13 hrs	1 sec		
1575	PRK-DOWN	TEST	Finished	11.11 12:13:58	11.11 12:19:59	14 hrs	1 sec		
1654	PaKtner_SML	DEV	Finished	11.11 12:15:45	11.11 12:19:46	15 hrs	20 sec		
1372	IMK_OK	Conf3	Crashed	11.11 14:15:28	11.11 14:15:29	21 hrs	43 sec		
1358	IM_KO	DEV	Finished	11.11 15:15:34	11.11 15:17:35	22 hrs	2 min		
1618	POL3_2_2	TEST	Finished	10.11 13:15:51	10.11 13:25:51	2 days	73 min		
1656	OHA_Synchronization_Master	DEV	Finished	10.11 13:15:54	10.11 13:25:54	2 days	11 sec		
1654	PERFO_UP	Conf1	Finished	9.11 20:01:54	9.11 20:21:12	3 days	20 sec		
1752	EXPO_UP	DEV	Finished	9.11 13:15:59	9.11 13:25:00	3 days	11 min		

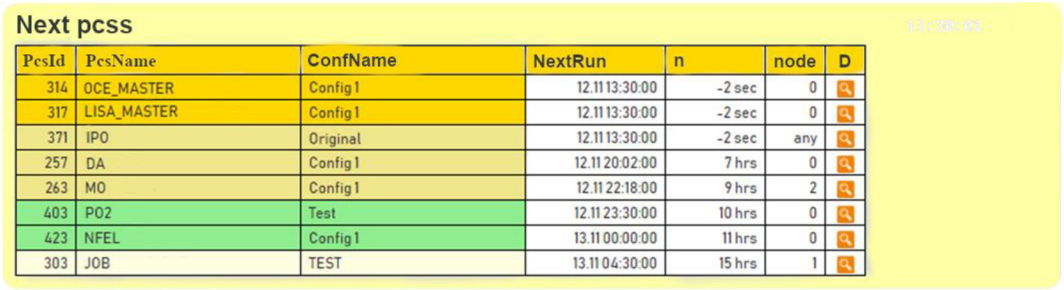
Obrázek 5 – Ukončené úlohy, přehled[vlastní]

Poslední ukončené úlohy:

Pole	Význam
PcsId	ID úlohy
PcsName	Jméno úlohy (zkrácené)
ConfName	Jméno konfigurace
LastState	Stav ukončení, viz kapitola *5.1.2*
Start	Čas spuštění
End	Čas ukončení
Before	Nyní – End
Last	Doba posledního běhu
D	Odkaz na definici Procesu
L	Odkaz na Log Procesu – nastaví se filtr na PcsId a čas Start/Finish

Tabulka 8 –Ukončené úlohy, popis přehledu

D. Následující úlohy


Next pcss

PcsId	PcsName	ConfName	NextRun	n	node	D
314	OCE_MASTER	Config 1	12.11 13:30:00	-2 sec	0	
317	LISA_MASTER	Config 1	12.11 13:30:00	-2 sec	0	
371	IPO	Original	12.11 13:30:00	-2 sec	any	
257	DA	Config 1	12.11 20:02:00	7 hrs	0	
263	MO	Config 1	12.11 22:18:00	9 hrs	2	
403	PO2	Test	12.11 23:30:00	10 hrs	0	
423	NFEL	Config 1	13.11 00:00:00	11 hrs	0	
303	JOB	TEST	13.11 04:30:00	15 hrs	1	

Obrázek 6 –Následujících úlohy, přehled[vlastní]

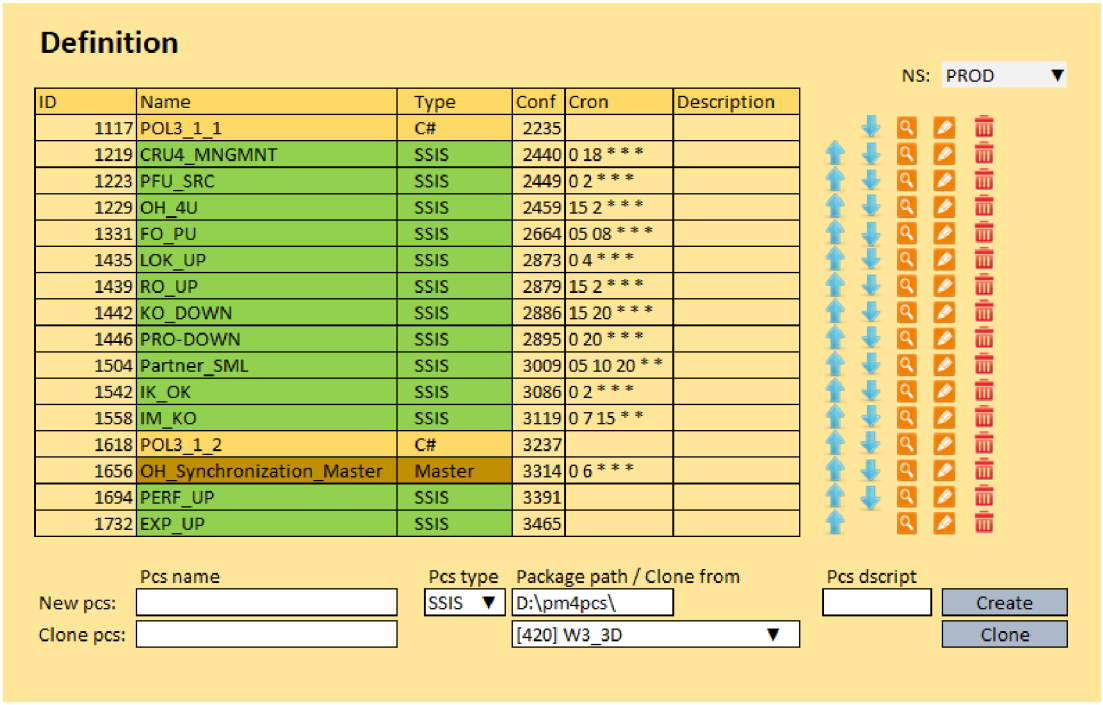
Úlohy, které budou následně spuštěny:

Pole	Význam
PcsId	ID úlohy
PcsName	Jméno úlohy (zkrácené)
ConfName	Jméno konfigurace
NextRun	Čas, kdy budou spuštěny
In	NextRun – Nyní
Node	Uzel nastavený v definici
D	Odkaz na definici Procesu

Tabulka 9 – Následujících úlohy, popis přehledu

5.1.4 Definition

Modul zobrazuje tabulku všech Procesů se základními atributy.



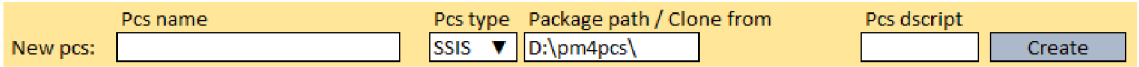
Obrázek 7 - Definition, seznam procesů[vlastní]

Pro každý Proces je k dispozici (v závislosti na právech uživatele) několik nástrojů:

Nástroj	Funkce
	Přesun v tabulce výše (význam pouze pro zobrazení)
	Přesun v tabulce níže (význam pouze pro zobrazení)
	Zobrazení jednoho Procesu v režimu prohlížení – viz kapitola *B*
	Zobrazení jednoho Procesu v režimu editace – viz kapitola *B*
	Smazání Procesu (nevratné)

Tabulka 10 – Definition, seznam procesů, popis funkcí nástrojů

A. Založení Procesu



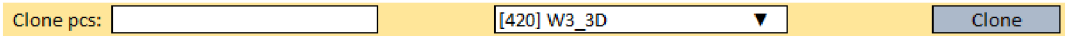
Obrázek 8 - Definition, založení procesu[vlastní]

Pole	Význam
Pcs name	Jméno Procesu
Pcs type	Výběr typu Procesu, viz kapitola ***Error! Reference source not found.***
Package path	Prvotní nastavení jednoho Package, viz kapitola *B*
Pcs dscrpt	Popis Procesu

Tabulka 11 – Definition, založení procesu, popis

Proces je založen v aktuálním NS a položku Pcs type nelze editovat.

B. Kopírování Procesu



Obrázek 9 - Definition, kopírování procesu[vlastní]

Pole	Význam
Pcs name	Jméno Procesu
Clone from	Výběr Procesu, který je kopírován

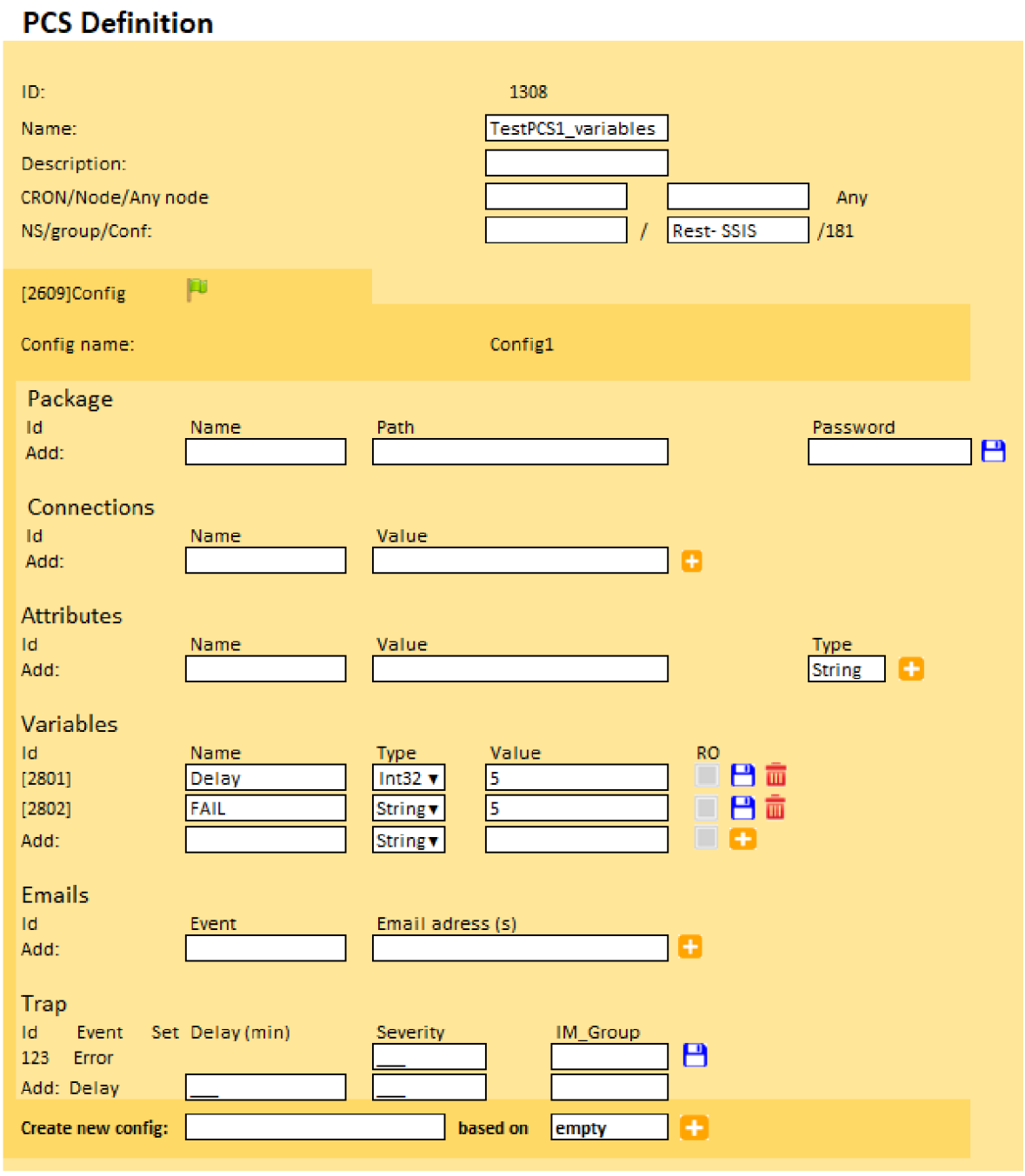
Tabulka 12 – Definition, kopírování Procesu, popis

Proces je založen v aktuálním NS, které lze později změnit.

C. Parametrizace Procesů

Každý Proces je nutno parametrizovat, aby byl PM4 schopen s ním pracovat.
Parametrizace Procesu je závislá na typu Procesu.
S obrazovkou se pracuje intuitivně, každou změnu je třeba individuálně zapsat – ikona .

Zde lze vidět uživatelské rozhraní definice procesu:



Obrázek 10 – Definition, definice Procesu[vlastní]

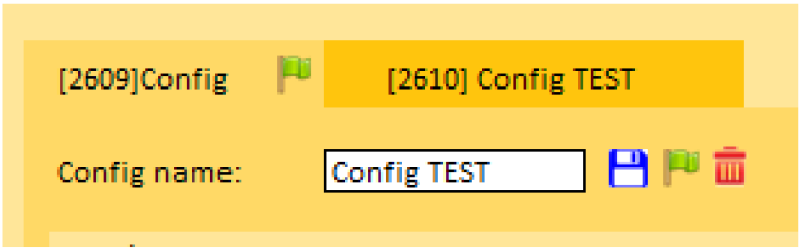
a. Společné parametry

Po-ložka	DB	Popis
ID	PcsId	Unikátní číslo Procesu
Jméno	PcsName	Stručné jméno Procesu
Popis	Descrip-tion	Popis.
Cron	Cron	Specifikuje parametry spouštění v syntaxi cron. Viz např. ZDROJ: https://en.wikipedia.org/wiki/Cron, https://crontab.guru/ ,http://www.cronmaker.com/ atd.
Node	Node	Default Node, na kterém systém poběží Seznam a konfigurace nodů není předmětem nastavování Definition
Any	AnyNode	Pokud je nastaveno, systém PM4 může spustit Proces na jakémkoliv nodu
NS	NS	Jméno NS (přesunutí do jiné NS)
Group	Group	Zařazení Procesu do skupiny (pro účely filtrování)

Tabulka 13 – Definition, společné parametry

b. Konfigurace

Parametrizace Procesu je obsažena v konfiguraci. Proces musí obsahovat alespoň jednu konfiguraci. Ta je automaticky vytvořena při vytvoření Procesu. Právě jedna konfigurace je označena jako aktivní. Konfigurace jsou v definici Procesu zobrazeny jako záložky, přepínání konfigurací se provádí kliknutím na jejich název.



Obrázek 11 - Definition, konfigurace procesu[vlastní]

Znak  vedle názvu konfigurace označuje aktuální konfiguraci. U každé konfigurace lze změnit její název.

Konfigurace, které nejsou aktivní, lze:

- Smazat příkazem 
- Označit jako aktivní příkazem 
- Uložit příkazem 

Konfigurace obsahuje v závislosti na typu Procesu panely pro nastavení:

- Package
- Connections
- Attributes
- Variables
- SlavePcss

c. C#

C# proces – specifická konfigurace

Package.Path	cesta na c# soubor obsahující makro / makra
Connection[Value]	Connection pro použití Procesem (podle Procesu)
Attributes[]	Atributy, které může Proces pomocí knihoven PM4 číst
Variables[]	Proměnné, které může Proces pomocí knihoven PM4 číst a zapisovat

Tabulka 14 – Definition, C# proces parametry

d. SSIS

SSIS / DTSX proces – specifická konfigurace

Package.Path	cesta na dtsx soubor
Connection[Value]	Connections, které se předávají do DTSX
Variables[]	Proměnné, které se předávají do DTSX

Tabulka 15 – Definition, SSIS / DTSX proces parametry

e. Master

Master proces – specifická konfigurace

Slave Pcss	ID Procesů, které jsou spouštěny

Tabulka 16 – Definition, master proces parametry

- Master Proces obsahuje seznam Procesů, které postupně spouští.
- Procesy mohou být jakéhokoliv typu, mimo včetně MasterProces.
- Pro Procesy typu Master platí následující omezení:

  Jako slave může být uveden další master Proces, který může mít stejně tak jako slave master Procesy, ale žádný z nich se v této řadě nemůže opakovat, aby nedošlo k zacyklení.

  FE umožní takovou konfiguraci uložit, ale zobrazí varování se zobrazením cyklické cesty. Uložení je možné, protože oprava chyby je možná na více místech – ve kterémkoliv Procesu účastnícím se cyklické cesty. V real-time je pak spuštění takového Procesu odmítnuto a přejde ihned do stavu „Startfailure“.

f. SQLP

SQL Stored Procedure – specifická konfigurace

Package.Path	Jméno SP
Connection[Value]	Connection na DB včetně UID a pwd (hesla)
Variables[]	Proměnné – parametry procedury Pokud jsou RO (read-only), jsou *input*, jinak jsou *output*

Tabulka 17 – Definition, SQL stored procedure parametry

Connection string pro MSSQL např:
uid=*user*;pwd=*****;database=*dbname*;server=*ServerSql.co.cz*

g. VBA

VBA makro – specifická konfigurace

Package.Path	cesta na xlsm soubor obsahující makro / makra
Attributes[VBA_MACRO]	makro / makra, která budou spouštěna
Attributes[VBA_SAVE_PATH]	Pokud je nastaveno, uloží se sešit s tímto názvem souboru

Tabulka 18 – Definition, VBA makro parametry

Tento typ umožňuje spustit aplikaci Excel, otevřít Workbook a v něm spustit postupně makra dle parametrizace.

Pro proces RefreshKPI je doplněna možnost nastavit atribut VBA_SAVE_PATH, což zajistí, že po spuštění všech maker je soubor uložen pod jménem uvedeným jako hodnota atributu BA_SAVE_PATH.

D. **Email**

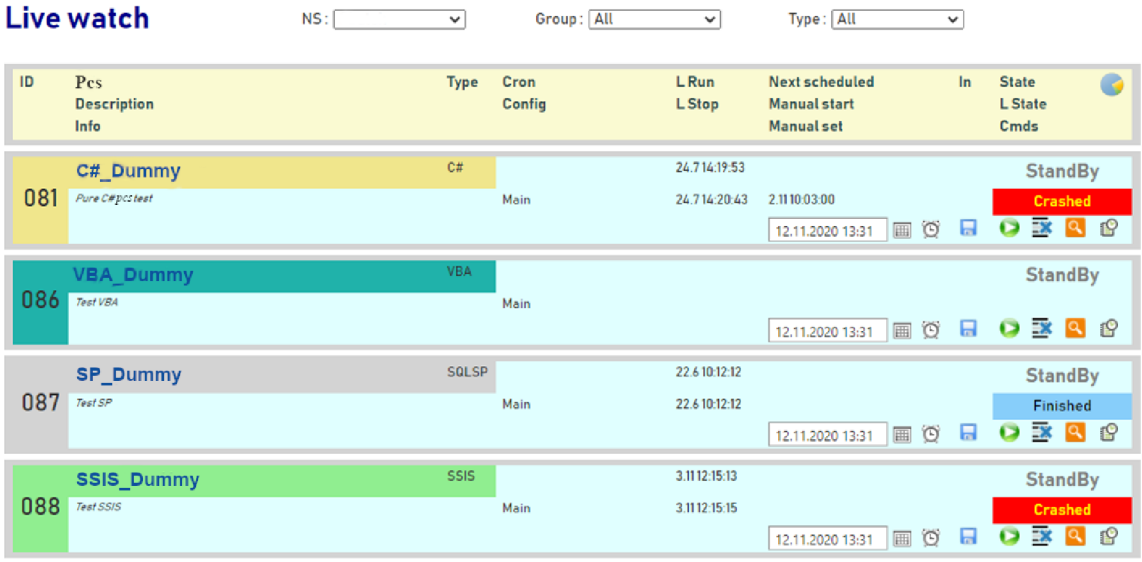
Zasílání email zpráv při změně stavu Procesu

Name	Pojmenování (technicky nemá význam)
Event	Stav, při jehož vzniku je zaslán email (Crashed, Finished, StartFailure…)
Email address (s)	Seznam email adres oddělených středníkem

Tabulka 19 – Definition, email parametry

5.1.5 Watch

Stránka slouží k zobrazení živých informací o aktuálním stavu úloh.



Obrázek 12 – Watch, obrazovka [vlastní]

Údaje:

- ID – PcsId
- Pcs/PcsDescription – krátký a dlouhý název úlohy
- Info – stručné info o běhu
- Cron – plánovací string
- Config – jméno konfigurace
- L.Run – čas posledního spuštění
- L.Stop – čas posledního ukončení nebo přechodu do stavu *StartFailure*
- Next Scheduled – čas následujícího plánovaného spuštění
- Manual start – čas, kdy bude úloha spuštěna mimo plán
- Manual set – pole pro zadání Manual Start
- State – aktuální stav úlohy
- L.State – stav úlohy při poledním ukončení
- In – čas, za který bude úloha spuštěna

Ikony / příkazy:

Ikona	Příkazy
	Jednorázové okamžité spuštění úlohy
	Zastavení běžící úlohy
	Enable úlohy – přechod Disabled → StandBy
	Disable úlohy – nebude dále plánována
	Prohlížení definice úlohy (stránka Definition)

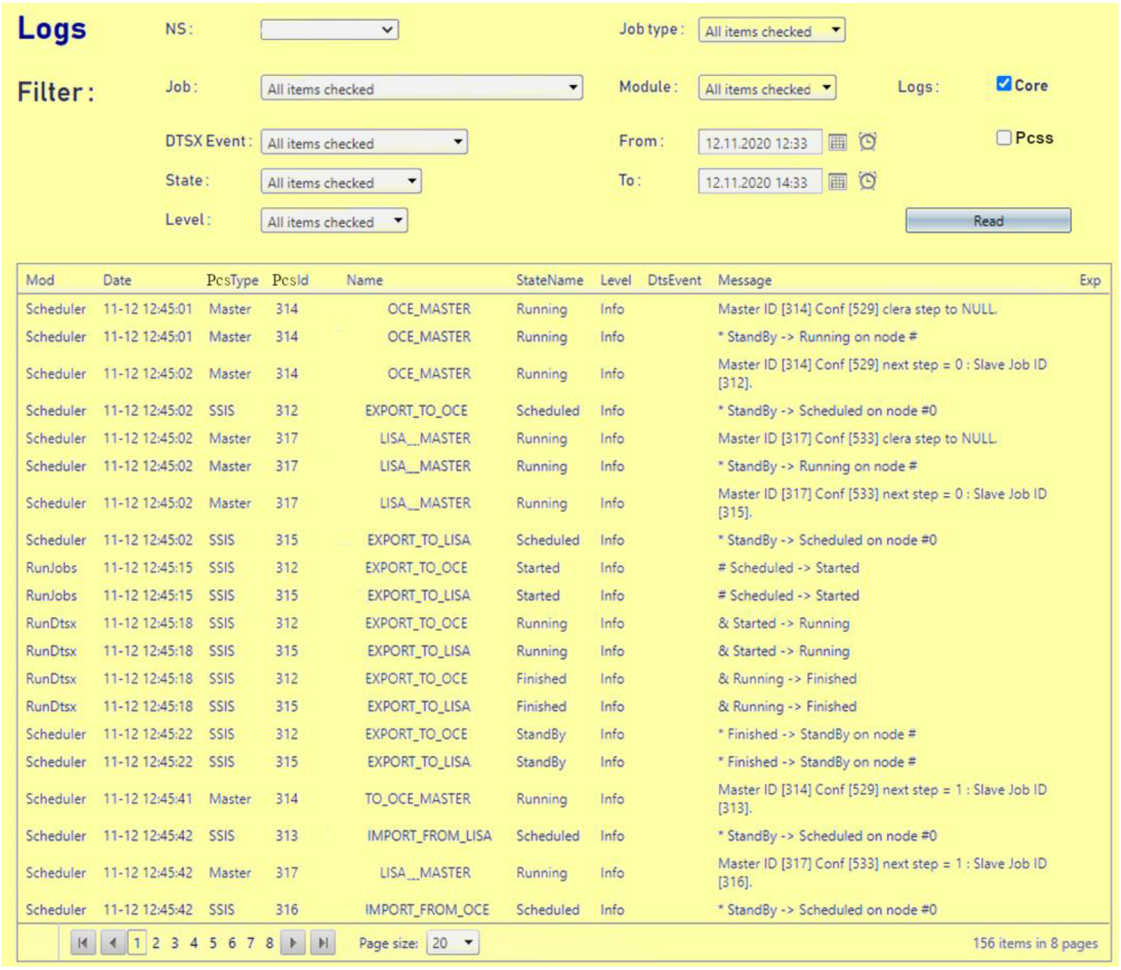
Tabulka 20 – Watch, ikony/příkazy

Obrazovka periodicky obnovuje informace o úlohách.

Seznam úloh je vytvořen v okamžiku zobrazení, a pokud by se výčet úloh změnil, je třeba ji obnovit (při změně definice v jiném okně).

5.1.6 Logs

Stránka slouží k zobrazování historických hodnot a stavů při provádění úloh.



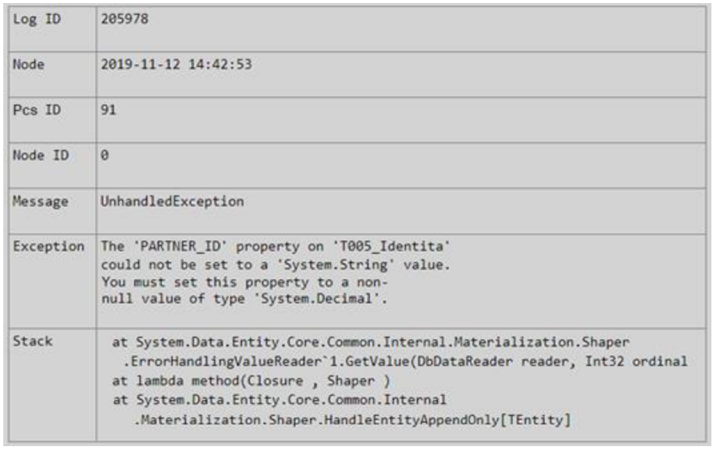
Obrázek 13 – Logs, obrazovka [vlastní]

V záhlaví stránky je umístěn filtr, který umožňuje omezit výpis:

- Module – výpisy pro vybrané moduly
- PcsType – omezení výpisu podle typu úlohy
- Pcs – omezení výpisu na jeden nebo více úloh
- From/To – výběr intervalu od .. do
- DTSX Event – omezení výběru na vybrané události DTS Listener (zprávy DTS programu)
- State – omezení výběru na vybrané stavy Procesu
- Level - omezení výběru na vybrané úrovně logů (Error,Info,Warning,Critical,..)
- Logs :
    - Core – informace PM4 modulů Scheduler, RunPcss a Postman
    - Pcs – informace které produkují
- Events only – omezení typů událostí

Zobrazené informace:

- Mod – modul, který záznam vygeneroval
- Date – čas záznamu
- PcsType – typ úlohy
- PcsId – číslo úlohy
- PcsName – jméno Procesu
- StateName – jméno stavu
- Level – úroveň (charakter) záznamu
- DtsEvent – kategorie zprávy v případě záznamu z DTSX balíčku
- Message – zpráva modulu/úlohy
- Exp – umístění ikony ⚠ je příznak, že daný záznam obsahuje dodatečné informace o chybě (Exception,Stack trace). Po kliknutí se zobrazí tato informace v samostatném okně:



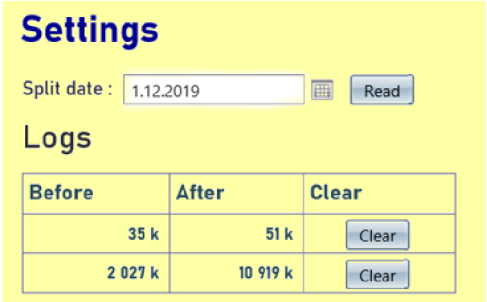
Log ID	205978
Node	2019-11-12 14:42:53
Pcs ID	91
Node ID	0
Message	UnhandledException
Exception	The 'PARTNER_ID' property on 'T005_Identita' could not be set to a 'System.String' value. You must set this property to a non-null value of type 'System.Decimal'.
Stack	at System.Data.Entity.Core.Common.Internal.Materialization.Shaper .ErrorHandlingValueReader`1.GetValue(DbDataReader reader, Int32 ordinal at lambda method(Closure , Shaper) at System.Data.Entity.Core.Common.Internal .Materialization.Shaper.HandleEntityAppendOnly[TEntity]

Obrázek 14 – Logs, dodatečné informace o chybě, obrazovka [vlastní]

5.1.7 Settings

Stránka zobrazuje stav (objem logů).

Uživatel s administrátorskými právy, může vymazat obsah logů do nastaveného data (Split date).



Obrázek 15 - Objem logů, obrazovka [vlastní]

5.1.8 Databáze

Databáze využívá následující tabulky:

Tabulka	Význam
PC01_PcsState	Číselník – Stavy Procesů
PT01_PcsDef	Definice (parametrizace) Procesů
PT02_PcsRun	Runtime informace o Procesu
PT04_PcsAttr	Attributy– určeno pro specifickou parametrizaci vlastních Procesů
PT03_PcsConfig	Config – definice konfigurace / konfigurací Procesů
PT05_PcsPackage	Specifické podle typu Procesu
PT06_PcsConnect	Connection stringy pro jednotlivé konfigurace
PT07_PcsVariable	Proměnné specifické pro danou konfiguraci Procesu
PT08_PcsDepend	Závislosti – definice slave Procesů pro master Proces
PT11_HB	Heart beat – ukládání runtime informací serverů, nesouvisí s jednotlivými Procesy
PT21_LogCore	Log – pro vlastní PM4 core (plánování, spouštění, chyby)
PT22_LogPcss	Log – logy vlastních Procesů
PT23_LogFE	Log – logování zásahů uživatele (změny parametrů, start, abort, …)

Tabulka 21 – Databáze, souhrn tabulek

PC01_PcsState

Číselník stavů.

PT01_PcsDef

Mimo položky vysvětlené v kapitole 0 obsahuje položky:

Sloupec	Význam
Author	Login uživatele, který Proces vytvořil
Created	Datum a čas vytvoření Procesu
ConfId	ID aktuální konfigurace

Tabulka 22 – Databáze, PT01_PcsDef

PT02_PcsRun

Obsahuje runtime informace pro danou úlohu, zejména její stav.

PT03_PcsConfig

Parametrizace konfigurací úloh (pro danou definici).

PT04_PcsAttr

Parametrizace atributů (pro danou konfiguraci).

PT05_PcsPackage

Parametrizace balíčků DTSX pro úlohy typu SSIS (pro danou konfiguraci).

PT06_PcsConnect

Parametrizace Connection Strings (pro danou konfiguraci).

PT07_PcsVariable

Parametrizace proměnných (pro danou konfiguraci).

PT08_PcsDepend

Parametrizace podřízených úloh pro úlohy typu *MasterProces*.

PT11_HB

Realtime informace o činnosti modulů PM4: Scheduler a RunPcss. Jejich zobrazení – viz kapitola ***Error! Reference source not found.***.

PT21_LogCore

Obsahuje historické informace o činnosti jádra a modulů PM4.

PT22_LogPcss

Obsahuje historické informace, které ukládají Procesy při svém běhu.

Podle typu Procesu jsou to:

- C# Informace zapisuje Proces pomocí funkcí z dll knihoven PM4.Core a PM4.RunBase
- SSIS Informace, které obálka Procesu obdržela metodou DtsxEventListener
- SQLSP Neumožňuje zápis
- VBA Neumožňuje zápis
- Master Nemá význam

PT23_LogFE

Obsahuje historické informace o činnosti prostřednictvím FE – PM4.Admin:

- Změny parametrizace (založení, mazání Procesů, změny jejich parametrů)
- Realtime zásahy (Disable/Enable Proces, start, Stop)

C#

Programování se provádí standardními nástroji .NET programování a jsou použity knihovny .NET Framework 4.6.2 (nebo vyšší).

Typicky je Proces typu Console application nebo WPF.

Kontrola systému

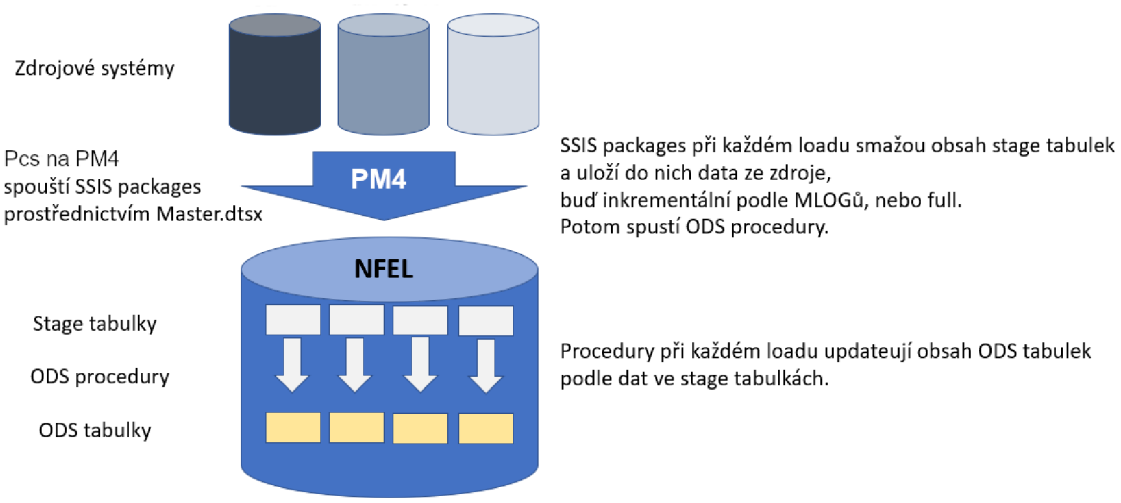
Základní kontrola systému a jeho konfigurace se provádí FE rozhraním PM4.Admin

Základní předpoklady chodu PM4 jsou:

- Dostupnost databáze vč. síťových služeb
- Funkční server, na kterém je spuštěn PM4.Scheduler
- Alespoň jeden funkční server, na kterém je spuštěn PM4.RunPcss
- Funkční servery, na kterých má spuštěn PM4.RunPcss a které jsou definovány jako výhradní servery pro jakýkoliv Proces (tj. má-li některý Proces nastaven NodeId na tento server a nemá nastaveno AnyNode).
- Správná parametrizace Procesů

5.2 Proces pro synchronizaci zdrojových systémů ODS

Následující kapitola popisuje proces pro synchronizaci zdrojových systémů ODS.



Obrázek 16 - Proces pro synchronizaci zdrojových systémů ODS, diagram [vlastní]

Načítání dat (load) se spouští pomocí několika procesů umístěných v Process Manageru 4.

Každý proces načítá jinou množinu tabulek, podle frekvence běhu a zdrojového systému.

V procesech je potřeba nastavit tyto proměnné:

- **Box** – určuje množinu tabulek které bude proces načítat. Kombinace frekvence loadu, zdrojového systému, a volitelně ještě pole box v tabulce CFG_DIRECTOR. např. load tabulek ze systému TIA, s frekvencí jednou denně značíme: TIA_D. viz pole SYSTEM_FREQ_BOX ve view V_CFG_DIRECTOR

- **Stg_packages_path** – adresa složky, kde jsou umístěné ssis balíčky na serveru. Hodnota by měla být shodná s cestou na úložiště, například: D:\PM4jpcs\NFEL\

- ***extra_where*** – nepovinná proměnná spouštěného balíčku Master.dtsx, kterou můžeme použít při (zejména manuálním) testování, aby proběhlo pouze načtení např. jedné tabulky. správný tvar je např. tento: and table_id in (65,66)

Na test serveru jsou naplánované například tyto procesy:

- NFEL daily (box TIA_D)
- NFEL hourly (box TIA_H)
- NFEL full loads daily (box TIA_D_FULLLOADS)
- NFEL full loads hourly (box TIA_H_FULLLOADS)
- Každý job spouští integration package Master.dtsx.

5.2.1 Master.dtsx

Provádí následující úkony:

I. V tasku *Get list of tables to load* vybere seznam *table_ids* z tabulky *CFG_DIRECTOR* podle hodnoty variable box.

II. Následně se spustí for each loop, který pro každou tabulku:

   a. Průběžně loguje průběh loadu do tabulky *LOAD_LOG*, včetně časů jednotlivých kroků, počtu řádků změněných v jednotlivých krocích a posledního načteného *SEQUENCE$$* u inkrementálně načítaných tabulek.

   b. Script task *run Staging package* spustí načtení ze zdrojového systému do stage tabulky. Pro tento load je pro každou staging tabulku vytvořen jeden SSIS balíček. Tyto balíčky jsou uložené ve stejné složce na serveru jako *Master.dtsx* (adresa této složky je daná proměnnou Stg_packages_path v jobu).

   Více informací o těchto balíčcích následuje níže.

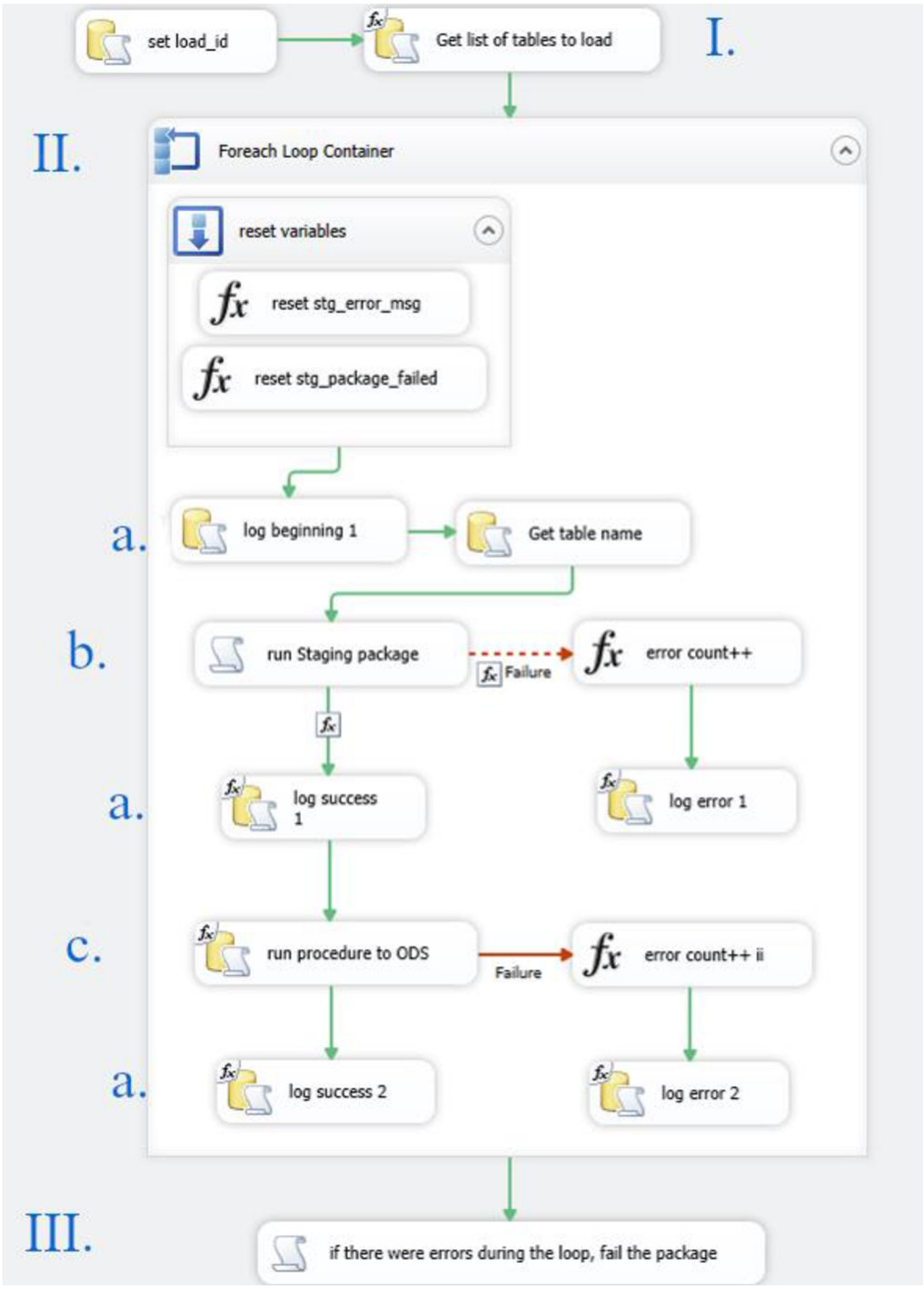
Pozn. Process Manager 4 nepodporuje ssis projekty, proto bohužel není možné použít Execute package task a je nutné použít script task.

c. *execute sql task run procedure to ODS* plní ODS tabulky ze stage tabulek. Vybere proceduru ke spuštění v závislosti na *table_id/table_name*, použije *log_id* jako vstupní parametr a počet smazaných řádek s počtem řádek vložených jako parametr výstupní.

   Více informací o těchto procedurách následuje níže.

III. Pokud některý z kroků selhal, for each loop postoupí k další tabulce, nicméně na závěr, v tasku *if there were errors during the loop, fail the package* je zajištěno, že celý proces nahlásí chybu. V chybové hlášce je uveden počet tabulek, jejichž načtení selhalo. Pro podrobnější informace o chybách je možné se podívat do tabulky *LOAD_LOG*.

Jak lze vidět na následujícím obrázku:



Obrázek 17 – Master.dtsx, obsah .dtsx [vlastní]

5.2.2 Staging balíčky

Základní funkce staging balíčků je načíst query z tabulky V_CFG_QUERYSTRINGS a podle něj naloadovat do stage tabulek inkrementální data ze zdrojového systému.

Pozn. Bylo ovšem potřeba naimplementovat několik workaroundů, vyplývajících mimo jiné z omezení na Process Manageru 4.

Jednotlivé kroky:

I. Script task *read connection files* načte connection strings ze souborů uložených na serveru (ve stejné složce jako staging balíčky, tato složka je specifikovaná v proměnné *path2connections_files*).

   Pozn. toto je nutný workaround protože PM4 nepodporuje project connection managers v ssis a to z důvodu kódování hesla.

II. *execute sql task Get Last loaded id* načte nejvyšší hodnotu SEQUENCE$$ pro danou tabulku z tabulky LOAD_LOG do proměnné seq_id proto, aby se z MLOG tabulek z databáze načetla jen data přidaná od posledního úspěšného loadu.

III. *Get Variable values* načte query z tabulky V_CFG_QUERYSTRINGS, ve kterém vymění placeholder za hodnotu seq_id.

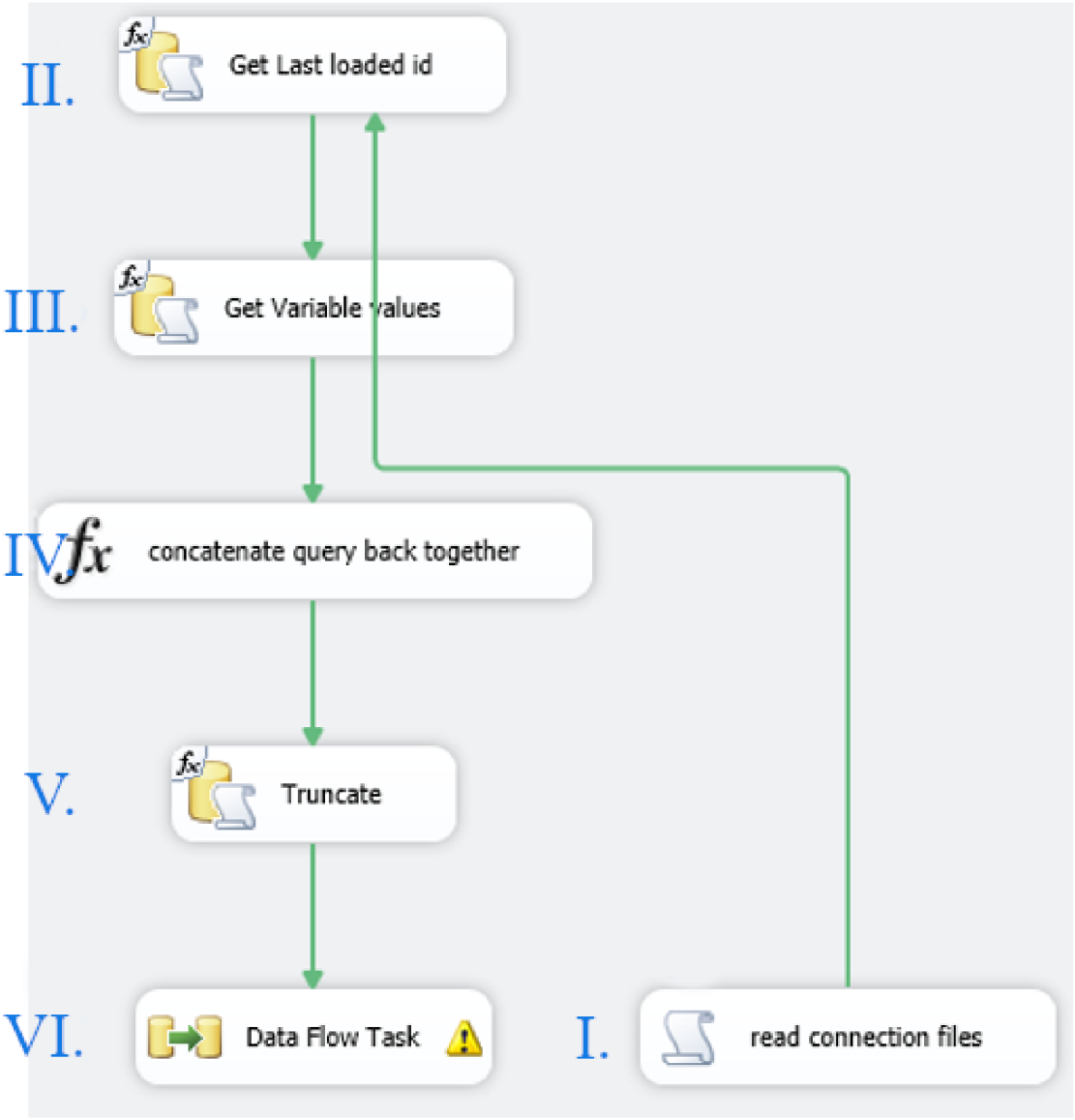
   Pozn. implementovaný workaround: toto query se načte do několika proměnných, protože není možné předat z Oraclu do proměnné v SSIS textový řetězec delší než 4000 znaků.

IV. *concatenate query back together* spojí řetězce získané z předchozího kroku zpět do proměnné query.

V. Truncate staging table

VI. Data Flow Task načte data ze zdroje podle proměnné query do cílové stage tabulky.

Jak lze vidět zde:



Obrázek 18 – Staging balíčky, obsah .dtsx [vlastní]

5.2.3 Procedury pro ODS tabulky

Jsou automaticky vygenerované pomocí procedury *SP_CFG_WRITE_ODS_TABLES_PROCEDURES*, která spouští funkci *FN_WRITE_ODS_TABLES_PROCEDURE*.

Fungují na principu *DELETE+INSERT* – jedná se o efektivnější řešení než *MERGE*, který v tomto případě není potřeba, protože ODS tabulky nejsou rozdělené do faktových tabulek a dimenzí.

Procedury mají následující parametry:

- vstupní parametr *log_id* – cizí klíč do tabulky *LOAD_LOG*
- výstupní parametry vracející počet smazaných řádek a počet vložených řádek.

5.2.4 Evidence a dokumentace databázových objektů

V NFEL databázi se vyskytují následující 3 základní typy tabulek:

A. **Konfigurační tabulky a pohledy** – používané k řízení ETL procesu. Mají předponu CFG (s výjimkou tabulky LOAD_LOG).

   CFG_DIRECTOR

   Tabulka CFG_DIRECTOR obsahuje všechny načítané tabulky. Plní se ručně (nebo eventuálně ad-hoc napsaným insertem) a je používána při načítání.

   - TABLE_ID – automaticky inkrementované číslo
   - SYSTEM_ID – cizí klíč do tabulky CFG_SYSTEM
   - TABLE_NAME – jméno stage tabulky
   - LOAD_FULL – určuje, jestli se tabulka má načíst celá (pokud má pole hodnotu 1). Pole se nepoužívá v run-time při loadu, ale využívá ho např. procedura SP_CFG_QUERYSTRINGS – viz dále.

     Pozn. Toto je pouze pro potřeby ddl procedur: pole může mít i hodnotu 2, ve speciálním případě tabulek, které se sice nemohou loadovat celé, ale nemají MLOG, mají proto vyřešenou inkrementální logiku indiviuálně.

- LOAD_FREQUENCY – pole se používá v run-time při loadu rozhoduje, jestli se tabulka bude loadovat každý den nebo každou hodinu (tj. podle hodnoty pole tabulku bude loadovat job naschedulovaný denně či každou hodinu). Hodnoty mohou být 'D' nebo 'H'.

- BOX – pole se používá v run-time při loadu zatím nevyplněné, určené k umožnění vyšší flexibility/paralelizace procesů.

- FLAG_DISABLED – v případě že má hodnotu 1, tabulka se nebude loadovat (není třeba vyplňovat, prázdná hodnota má stejný výsledek jako 0).

V_CFG_DIRECTOR – přidává navíc pole SYSTEM_FREQ_BOX, které provádí concate pole system_name, load_frequency, a box pro jednodušší výběr tabulek loadovaných ve specifickém procesu.

V_CFG_TABLES – uživatelsky přívětivé zobrazení, obsahuje všechny důležité informace ke všem tabulkám na jednom místě.

CFG_QUERYSTRINGS

Tabulka je plněná ad-hoc, pomocí procedury SP_CFG_QUERYSTRINGS.

Obsahuje dotazy používané v run-time při loadu tabulek ze zdrojových systémů do stage tabulek.

- TABLE_ID – cizí klíč do tabulky CFG_DIRECTOR

- FLAG_MANUAL_INPUT – v případě, že je hodnota 1, procedura

- SP_CFG_QUERYSTRINGS – nebude měnit hodnotu tohoto řádku

- QUERYSTRING_MINUS – pro použití při každodenním loadu

- QUERYSTRING – slouží pouze pro použití při tvorbě SSIS balíčků. Oproti QUERYSTRING_MINUS má navíc u každého pole CAST AS, kvůli tomu, aby SSIS správně validovalo metadata.

- QUERYSTRING_INITIAL_LOAD – pouze pro použití při počátečním loadu. Oproti QUERYSTRING_MINUS neobsahuje where podmínku, a především vybírá data z "hlavních" tabulek, zatímco QUERYSTRING_MINUS vybírá data z MLOG tabulek.

  Pozn. SSIS neumí pracovat s Oracle datovými typy CLOB a BLOB. CLOB pole (dlouhý textový řetězec) bylo proto potřeba zkrátit na NVARCHAR2(4000), a BLOB pole (binary object) je zcela vynechané z loadu.

V_CFG_QUERYSTRINGS – používá se v run-timeu při loadu. Slouží pro specifikaci, které z polí QUERYSTRING_INITIAL_LOAD, QUERYSTRING_MINUS se použije při loadu. Při eventuálním počátečním loadu je potřeba změnit toto view aby vybíralo pole QUERYSTRING_INITIAL_LOAD, jinak se vždy vybírá pole QUERYSTRING_MINUS.

CFG_SYSTEMS

Plní se ad-hoc ručně.

- SYSTEM_ID
- SYSTEM_NAME – použije se při tvorbě jmen tabulek – stage i ODS

LOAD_LOG

Tabulka se plní SSIS během loadu balíčkem Master.

- LOG_ID – automaticky inkrementované číslo
- TABLE_ID – cizí klíč do tabulky CFG_DIRECTOR
- TS_START – čas začátku loadu tabulky ze zdroje do stage
- TS_STAGE – čas dokončení loadu tabulky ze zdroje do stage
- TS_END – čas dokončení loadu tabulky ze stage do ODS
- SUCCEEDED
- LAST_LOADED_ID – nejvyšší SEQUENCE$$ (v případě TIA) naloadované do stage tabulky
- ROWCOUNT_STAGE
- ROWCOUNT_ODS_DELETE
- ROWCOUNT_ODS_INSERT
- ERROR_MESSAGE – případný error message z loadu tabulky ze zdroje do stage (pro load mezi stage a ODS v tuto chvíli není naimplementováno).
- LOAD_ID identifies one run of the load

V_LOAD_LOG – uživatelsky přívětivé zobrazení logu

B. **Stage tabulky** – „pracovní tabulky“ sloužící pro úpravu dat před vložením do finální tabulky. V databázi mají předponu STG.

Jmenná konvence: STG_ + zkratka zdrojového systému, např. TIA_ + název tabulky ve zdroji například tedy STG_TIA_TCP_INET_STAT_PLACE

Tabulky mají identickou strukturu jako odpovídající tabulky na zdrojovém systému, v případě TIA navíc obsahují následující sloupce vztahující se k MLOG tabulkám:

- SEQUENCE$$ - incremental id z MLOG tabulky
- OLD_NEW$$ - signifies DML action. O for deleted rows, U for old value of updated rows, N for new rows, or new value of updated rows
- M_ROW$$ - ROWID zdrojové tabulky, nikoliv MLOGu

C. **ODS tabulky –** finální tabulky určené pro použití uživateli databáze.

Jmenná konvence: zkratka zdrojového systému_ + název tabulky ve zdroji například tedy TIA_TCP_INET_STAT_PLACE.

Tabulky mají identickou strukturu jako odpovídající tabulky na zdrojovém systému přidávají se následující sloupce:

ODS_LOG_ID – cizí klíč na LOAD_LOG

M_ROW$$ - ROWID ze zdrojové tabulky také funguje jako primární klíč u tabulek které nemají primární klíč na zdroji.

6 Zhodnocení efektivity řešení

Vzhledem k použitým technologiím je třeba využít mnoha workaroundů, jak bylo již poznámkami v minulé kapitole ostatně podotknuto. Prezentované řešení je funkční, avšak není ideální.

Hlavním úskalím je využití SSIS, potažmo .dtsx balíků, pro Oracle databázové řešení. Nynější verze používaného SSIS v daném podniku je z roku 2012. Mnoho funkcí z nejnovější verze Management Data Studia tudíž není v PM4 podporovaných.

Vzhledem k povaze nalezených nesrovnalostí nepovažuji za ekonomické rozhodnutí využít zdroje k úpravě PM4, aby vyhovoval požadavkům tohoto řešení. Doporučil bych obměnu použité technologie za komplexní řešení, které odpovídá nynějším trendům a zvyšujícím se nárokům na ODS. Stávající řešení je designováno na hodinové až denní synchronizace databází s ODS, což neodpovídá požadavkům dnešní doby.

Vzhledem k faktu, že v rámci daného podniku je cílem zajistit do budoucna near-realtime synchronizaci změn všech produkčních databází s ODS, doporučuji v tomto případě úplné vynechání PM4 z ODS, avšak jeho ponechání v provozu a případnou opravu pro ostatní ETL procesy.

Doporučuji také změnu procesu pro synchronizaci zdrojových systémů. Změna zahrnuje implementaci Kafka clusteru s využitím Kafka topiců a jejich propojení s databázemi pomocí odpovídajících connectorů, to už však nastíním rešerší v následujících kapitolách.

6.1 *Realtime replikace datových změn*

Cílem je připravit návrh řešení pro replikaci změn ze zdrojových databázových tabulek do tabulek ve zvolené cílové databázi v reálném čase. Navržené řešení je třeba realizovat na vybraných tabulkách a ověřit správnost.

Rešerše se vztahuje k Debezium verze 1.9.0.Final. V budoucnu při realizaci/upgradu na novější verzi nemusí být některé informace relevantní a bude nutné navrhované řešení porovnat a aktualizovat podle oficiální dokumentace. [22]

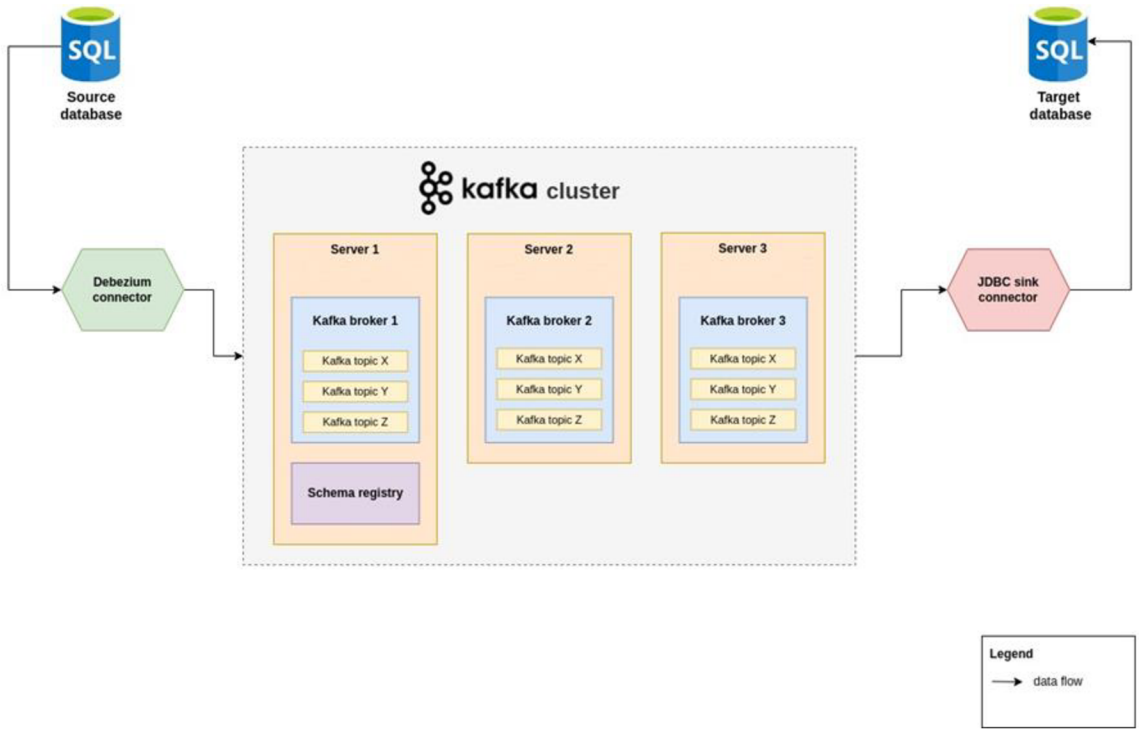
6.2 Komponenty použité v návrhu

Mezi technologie použité v návrhu patří Kafka cluster a jeho komponenty, které poskytují funkci Kafka topiců a jsou dále spojeny se zbytkem systému pomocí odpovídajících konektorů, tudíž Debezium connector a JDBC Sink Connector.

- Debezium [23]
- Debezium connector [24]
- JDBC Sink Connector [25]
- Kafka cluster a komponenty clusteru [26]
- Kafka topic [27]

6.3 *Solution model*

Takto vypadá celkový pohled na řešení replikace změn ze zdrojových databázových tabulek do cílové databáze:



Obrázek 19 - Solution model, diagram [vlastní]

Model se skládá z následujících komponent:

- **Source database**

Zdrojová databáze s tabulkami, ze kterých budou změny replikovány do tabulek v cílové databázi.

- **Target Database**

Cílová databáze s tabulkami, kam budou změny replikovány.

- **Kafka cluster**

Představuje seskupení několika message brokerů. Tito brokeři obsahují množinu topiců, které si lze zjednodušeně představit jako fronty zpráv. K jednotlivým topicům jsou pak navázáni producenti, kteří zajišťují zasílání zpráv do topiců a na druhé straně konzumenti, jejichž úkolem je čtení zpráv z topiců. [26]

Pro optimalizaci velikosti přenášených zpráv se zprávy transformují do binární podoby. Aby bylo možné provést serializaci/deserializaci zpráv z/do binární podoby a jejich validaci, přenáší se tato schémata zpráv v AVRO formátu [28] do Schema registry.

- **Debezium connector**

Stěžejní komponentou celého řešení je Debezium connector, který se stará o čtení změnových logů ve zvolených tabulkách ze vstupní databáze a zápis změnových logů (modifikovaných/nově vzniklých záznamů tabulek) do Kafka topiců. Z pohledu názvosloví Kafky je tento connector Kafka producentem. [24]

Detailní představení Debezia lze nalézt v oficiální dokumentaci projektu. [29]

- **JDBC sink connector**

Pro přenesení změnových záznamů z Kafka topiců do cílových databázových tabulek může být využit například JDBC sink connector. Jde tedy o Kafka konzumenta. [25]

6.4 Realizace

6.4.1 Instalace Debezia

Instalace Debezium connectoru může být společně s instalací Kafka clusteru zanesena například do Ansible skriptů, jejichž automatizovanou instalaci poté zajistí správce Kafka clusteru.

A. Time-based retenční politika Kafka topicu

Tato politika je založena na časovém hledisku mazání zpráv z Kafka topicu. Nastavení/úpravu časové retence na úrovni topicu je možné provést přidáním volby *--config* s těmito parametry:

retention.ms=604800000

segment.ms=604800000

Výchozí hodnota obou parametrů bývá nastavena na 604800000 ms (7dní)

B. Size-based retenční politika Kafka topicu

Další možností retence zpráv v Kafka topicu je stanovení limitní velikosti topiců. Ta se opět nastavuje při vytváření topicu (případně lze upravit na již vytvořeném topicu).
retention.bytes - Stanovuje maximální velikost na 1 partition Kafka topicu. Pro zjištění maximální velikosti topicu je třeba tuto hodnotu vynásobit počtem partitions nastavených pro daný topic. Výchozí hodnota je nastavena na neomezeno.
segment.bytes- Nastavuje maximální velikost segment souboru u topic partition. Výchozí hodnota je stanovena na 1 GB.

- **Výpočet velikosti Kafka topicu**

  Velikost Kafka topicu nelze striktně omezit. V případě, kdy budou do topicu data zasílána rychleji, nežli zvládne LogCleaner zpracovat, bude tato hodnota stanovený limit převyšovat.

  Při vytváření Kafka topicu je vhodné zvolit celkovou velikost topicu (standardně je velikost stanovena na 5 GB). Může se však hodit v některých situacích zvolit jinou velikost.

Jelikož nelze nastavit celkovou velikost topicu, velikost se nastavuje na úrovni partitions. Při výpočtu celkové velikosti Kafka topicu se tedy musí zohlednit počet partitions a výpočet by měl vypadat následovně:

1. Určení požadované celkové velikosti topicu - např. 3 GB.
2. Určení počtu *partitions* pro topic - např. 50.
3. Velikost parametru *retention.bytes* se udává v bajtech, proto se hodnota z bodu 1 převede na bajty, tedy např. 3 GB -> (1024 x 1024 x 1024 x 3)
4. Hodnota z kroku 3 se vydělí počtem *partitions* a výsledná hodnota se nastaví v parametru *retention.bytes* při vytváření nebo úpravě Kafka topicu. Např. 1024 x 1024 x 1024 x 3 / 50
5. Posledním krokem je nastavení hodnoty pro parametr *segment.bytes*. Zde je možné využít výslednou hodnotu z kroku 4 a vydělit ji vhodným koeficientem v závislosti zvolené hodnotě pro parametr *retention.bytes*. Například *retention.bytes* -> / 3 tedy 1024 x 1024 x 1024 x 3 / 50 / 3

[30]

6.4.2 Úprava parametrů existujícího Kafka topicu

V některých případech je třeba v čase změnit některé z parametrů existujícího Kafka topicu. Jako například po přenesení zálohy ze zdrojové DB, kdy nejprve nastavíme časovou retenci záznamů v topicu tak, aby nebyly záznamy z topicu smazány po dobu provádění zálohy a restore procesu ze zdrojové do cílové tabulky.

Na konci scénáře pak již není třeba mít u topicu nastavenu takto dlouho časovou retenci, a tak se hodnota může snížit. Obdobně může vzniknout potřeba přenastavit i jiné parametry.Podobně jako při vytváření topiců je možné i pro úpravu parametrů Kafka topicu využít Kafka-UI. [27]

6.4.3 Vytváření instancí connectorů

Dalším krokem v procesu replikace dat je vytvoření instancí connectorů. Jak bylo na Obr. 19 vyznačeno, řešení se skládá ze dvou typů connectorů. Pro zajištění variability (ve smyslu specifických nastavení) je vhodné vytvářet instance connectorů pro každou tabulku zvlášť. Tzn., že každá tabulka bude mít 2 instance connectorů (JDBC sink a Debezium).

Kompletní popis konfigurační parametrů se nachází v oficiální dokumentaci. [31]

Endpointy pro management instancí connectorů

Management connectorů se realizuje přes REST API vystavené na Kafka serverech. Na těchto serverech společně s Kafka brokerem musí být spuštěna i služba Kafka connect, která poskytuje REST.

6.4.4 Monitoring

V rámci monitorování stavu Kafka topiců a stavu connectorů by bylo vhodné napojit pomocí REST API Kafka a Kafka connect na centrální logovací systém firmy. Tím bude možné detekovat pády/chyby v běžících konektorech a zjednat včasnou nápravu.

6.4.5 Troubleshooting

V případě, že nastane nějaký problém, bude ho třeba řešit v závislosti na jeho konkrétním typu. Obecně lze ale při investigaci problému postupovat takto:

1. Je detekován problém v monitoringovém systému. Případně aktivní manuální kontrolou GET /connectors.
2. Z předchozího kroku pomocí REST API endpointu zjistíme, jaký konektor a na jakém brokeru selhal.
3. Na daném serveru pomocí příkazů z kapitoly Kontrola logů zkontrolujeme logy a identifikujeme problém.
4. Podle typu/povahy chyby, která může být konfigurační/databázová/aplikační rozhodneme, zda dokážeme vyřešit, případně se obrátíme na správce příslušné oblasti.

7 Shrnutí výsledků

Rešerše ukázala, že PM4 je vhodným nástrojem pro ETL procesy a synchronizaci dat mezi různými databázovými systémy. Poskytuje komplexní prostředí pro tvorbu, správu a automatizaci těchto procesů.

Na druhé straně bylo představeno Kafka Systems jako robustní platforma pro streamování dat, která umožňuje synchronizovat data v reálném čase. Je mimořádně vhodná pro rychlé a nízko-latentní synchronizace, tudíž například právě pro celopodnikový ODS.

Na základě této analýzy byla navržena možná strategie pro implementaci synchronizačních procesů v Kafka systems komplexním řešení. Tato strategie zahrnuje využití silných stránek PM4 pro ETL procesy a využití Kafka pro synchronizaci v reálném čase mezi rozličnými systémy.

8 Závěr

Na základě vypracované rešerše lze konstatovat, že stávající řešení využívající interního systému PM4 pro synchronizaci dat mezi interními databázovými systémy není optimálním. A to zejména vzhledem k omezením plynoucím z nedostatečné podpory nejnovějších funkcí, zejména pro Oracle databáze. Analýza ukázala, že vzhledem k narůstajícím požadavkům na near-realtime synchronizaci dat v celopodnikovém ODS je třeba hledat modernější a efektivnější řešení.

Byla navržena strategie, která využívá kombinaci dvou nástrojů: stávajícího PM4 pro ETL procesy a Kafky pro rychlou a nízko-latentní synchronizaci v reálném čase mezi systémy. Tímto způsobem by bylo možné dosáhnout komplexního řešení, které by spojilo ověřené funkce PM4 s moderními vlastnostmi Kafky. Tento přístup by umožnil progres v synchronizaci a splnil požadavky na dynamické datové operace.

Strategie, která navrhuje kombinaci interního PM4 pro ETL procesy a Kafka pro rychlou synchronizaci v reálném čase, by měla být podrobněji prozkoumána a analyzována v kontextu aktuální infrastruktury, technologických schopností a požadavků podniku. Implementace tohoto řešení vyžaduje plánování, testování a správu změn, aby byl úspěšně proveden plynulý přechod.

V následujících krocích by měl být vypracován plán postupné migrace a přechodu, který minimalizuje čas přerušení běhu stávajících procesů a zajistí tak hladký upgrade na nový hybridní model.

Je vhodné také zvážit aspekty týkající se školení zaměstnanců a týmů, které budou pracovat s novými technologiemi, aby bylo zajištěno efektivního využití předností obou z nástrojů.

Lze říci, že navrhovaná strategie představuje kruciální krok směrem k vyšší efektivitě synchronizace dat a schopnosti reagovat na požadavky moderního podnikového prostředí. Přechod na nový hybridní model kombinující silné stránky stávajícího stabilního PM4 a agilních možností Kafky by mohl přinést řadu výhod a zlepšení v oblasti synchronizace dat, což by následně mohlo pozitivně ovlivnit celkový výkon a konkurenceschopnost podniku.

Literatura

[1] *Db-engines* [online]. Vídeň, Rakousko: Solid IT, 2022 [cit. 2022-08-06]. Dostupné z: https://db-engines.com/en/ranking

[2] *About us* [online]. Vídeň, Rakousko: Solid IT, 2022 [cit. 2022-08-31]. Dostupné z: https://solid-it.at/

[3] *Oracle DB* [online]. United States of America: Oracle Corporation, 2022 [cit. 2022-08-31]. Dostupné z: www.oracle.com/database/

[4] *MS SQL Server Features* [online]. United States of America: Microsoft, 2019 [cit. 2022-06-20]. Dostupné z: https://www.microsoft.com/cs-cz/sql-server/sql-server-2019

[5] *ROUND (Transact-SQL)* [online]. United States of America: Microsoft, 2022 [cit. 2022-08-31]. Dostupné z: https://docs.microsoft.com/en-us/sql/t-sql/functions/round-transact-sql?view=sql-server-ver16¨

[6] *Transact-SQL Reference* [online]. United States of America: Microsoft, 2022 [cit. 2022-08-31]. Dostupné z: https://docs.microsoft.com/en-us/sql/relational-databases/databases/create-a-user-defined-data-type-alias?view=sql-server-ver16

[7] *Transaction locking and row versioning guide* [online]. United States of America: Microsoft, 2022 [cit. 2022-08-31]. Dostupné z: https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-transaction-locking-and-row-versioning-guide?view=sql-server-ver16

[8] *Transact-SQL Reference (Transact-SQL)* [online]. United States of America: Microsoft, 2008 [cit. 2022-08-31]. Dostupné z: https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2005/ms189826(v=sql.90)?redirected-from=MSDN

[9] *Configure the cost threshold for parallelism Server Configuration Option* [online]. United States of America: Microsoft, 2022 [cit. 2022-08-31]. Dostupné z: https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/configure-the-cost-threshold-for-parallelism-server-configuration-option?view=sql-server-ver16

[10] *Stored Procedures (Database Engine)* [online]. United States of America: Microsoft, 2021 [cit. 2022-08-31]. Dostupné z: https://docs.microsoft.com/en-us/sql/relational-databases/stored-procedures/stored-procedures-database-engine?view=sql-server-ver16

[11] *Transact-SQL Reference (Database Engine)* [online]. United States of America: Microsoft, 2022 [cit. 2022-08-31]. Dostupné z: https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver16

[12] Oracle Compared [online]. United States of America: Oracle Corporation, 2020 [cit. 2022-08-10]. Dostupné z: https://docs.oracle.com/cd/E10405_01/app-dev.120/e10379/ss_oracle_compared.html

[13] ESTUARY. What is data synchronization? Purpose, types, & methods. Estuary [online]. 20. June 2023 [cit. 2023-07-30]. Dostupné z:: https://estu-ary.dev/data-synchronization/

[14] *JÖRG, Thomas and Stefan DESSLOCH*. Towards generating ETL processes for incremental loading. Towards Generating ETL Processes for Incremental Loading [online]. 2008 [cit. 2023-07-30]. Dostupné z: https://www.resear-chgate.net/publication/221524731_Towards_genera-ting_ETL_processes_for_incremental_loading

[15] *Operational Data Store Event Loader* [online] [cit. 2023-07-30] Dostupné z: https://docs.informatica.com/data-integration/data-integration-hub/10-5/administrator-guide/dashboard-and-reports-management/operational-data-store-event-loader.html

[16] No More Silos: How to Integrate your Databases with Apache Kafka and CDC | Confluent. *Confluent* [online] [cit. 2023-07-30] Dostupné z: https://www.con-fluent.io/blog/no-more-silos-how-to-integrate-your-databases-with-apache-kafka-and-cdc/

[17] LUTKEVICH, Ben. database replication. Data Management [online]. 2022[cit. 2023-07-30]. Dostupné z: https://www.techtarget.com/searchdatama-nagement/definition/database-replication

[18] *Backing Up Your Database* [online]. 17. November 2005 [cit. 2023-07-30]. Dostupné z: https://docs.oracle.com/cd/B16351_01/doc/server.102/b14196/backrest003.html

[19] DEARMER, Abe. Top 5 Microsoft SQL ETL Tools for Data Integration. Integrate.io [online]. 2023 [cit. 2023-07-30]. Dostupné z: https://www.integrate.io/blog/microsoft-sql-etl-tools/

[20] THEODORE.JORDAN@ENTERPRISEDB.COM, PostgreSQL replication and Automatic Failover tutorial. EDB [online] [cit. 2023-07-30]. Dostupné z: https://www.enterprisedb.com/postgres-tutorials/postgresql-replication-and-automatic-failover-tutorial

[21] No More Silos: How to Integrate your Databases with Apache Kafka and CDC | Confluent. Confluent [online] [cit. 2023-06-25]. Dostupné z: https://www.con-fluent.io/blog/no-more-silos-how-to-integrate-your-databases-with-apache-kafka-and-cdc/

[22] Reference documentation. Debezium [online] [cit. 2023-06-25]. Dostupné z: https://debezium.io/documentation/

[23] Debezium Architecture: Debezium Documentation [online] [cit. 2023-06-25]. Dostupné z: https://debezium.io/documentation/reference/stable/architecture.html

[24] *Debezium Connector for Oracle: Debezium Documentation* [online] [cit. 2023-06-25]. Dostupné z: https://debezium.io/documentation/reference/stable/connectors/oracle.html

[25] *JDBC Sink Connector for Confluent Platform | Confluent Documentation* [online] [cit. 2023-06-25]. Dostupné z: https://docs.confluent.io/kafka-connectors/jdbc/current/sink-connector/overview.html

[26] Part 1: Apache Kafka for beginners – What is Apache Kafka? - CloudKarafka, Apache Kafka Message streaming as a Service. *CloudKarafka* [online]. 19. March 2020[cit. 2023-06-25]. Dostupné z: https://www.cloudkarafka.com/blog/part1-kafka-for-beginners-what-is-apache-kafka.html

[27] SUPPORT, Dattell – Kafka &Amp; Elasticsearch. What is a Kafka Topic? *Dattell* [online]. 2022[cit. 2023-06-25]. Dostupné z: https://dattell.com/data-architecture-blog/what-is-a-kafka-topic/

[28] Documentation *Apache Avro*[online] [cit. 2023-06-25]. Dostupné z: https://avro.apache.org/docs/

[29] *Debezium Documentation* [online] [cit. 2023-06-25]. Dostupné z: https://debezium.io/documentation/reference/1.8/

[30] *IBM Cloud Docs* [online] [cit. 2023-06-25]. Dostupné z: https://cloud.ibm.com/docs/EventStreams?topic=EventStreams-ES_understanding_reserved_disk_usage

[31] *Debezium Connector for Oracle: Debezium Documentation* [online] [cit. 2023-06-25]. Dostupné z: https://debezium.io/documentation/reference/stable/connectors/oracle.html#oracle-example-configuration

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor: **Radek Vancl**

Studium: I1800242

Studijní program: B1802 Aplikovaná informatika

Studijní obor: Aplikovaná informatika

Název bakalářské práce: **SQL databáze a jejich synchronizační komponenty**

Název bakalářské práce AJ: SQL databases and their synchronization components

Cíl, metody, literatura, předpoklady:

Cíl práce: Seznámení s SQL databázemi, typy synchronizací a vysvětlením co je to synchronizační komponenta, dále práce detailně popíše konkrétní v praxi používanou synchronizační komponentu a zhodnotí její výhody a nevýhody.

Osnova:

1. Úvod
2. SQL databáze
3. Typy synchronizací
4. Synchronizační komponenta
5. Popis konkrétní synchronizační komponenty
6. Zhodnocení efektivity řešení
7. Závěr

Zadávající pracoviště: Katedra informačních technologií,
Fakulta informatiky a managementu

Vedoucí práce: Ing. Tomáš Nacházel, Ph.D.

Oponent: Ing. Barbora Tesařová, Ph.D.

Datum zadání závěrečné práce: 21.1.2020