IT a anatomie firmy

(Reportingové BI řešení pro plánování výrobních kapacit v závodech)

(pracovní dokument)

Lukáš Rychetský

VŠE Praha, 2024

Obsah

1. Úvod

Pracovní dokument se zabývá konstrukcí obecného řešení architektury reportingu kapacit ve výrobních závodech. Dokument má převážně byznysový charakter, popsané výstupy dokumentu odpovídají byznysově poptávaným produktům.

Autor dokumentu je členem Business Intelligence oddělení, které vyvíjí pro různé subjekty nejrůznější reportingová řešení. Tato řešení jsou zpravidla v pilotní fázi konstruována na míru jednomu zákazníkovi. V případě úspěšnosti produktu vzniká otázka, jakým způsobem by se dalo původní řešení zobecnit tak, aby mohlo být nabízeno dalším klientům, jejichž práce by mohla být podpořena obdobným reportingovým nástrojem.

Jedním takovým řešením je v současnosti reporting výrobní kapacity pro nespecifikovaný výrobní závod, ve kterém pomocí několika dashboardů probíhá sledování vývoje výrobní kapacity napříč hierarchickou strukturou závodu, v čase, a dle různých složek nabídky a poptávky.

2. Kontext původního řešení

Původní reportingové řešení, zobrazující stav nabídky a poptávky po výrobní kapacitě v čase, bylo vyvíjeno jako pilotní řešení dané problematiky. Na základě definice požadavků s původním zákazníkem výrobního závodu bylo řešení v minulosti vypracováno a dále upravováno dle dodatečných požadavků tak, aby plně odpovídalo byznysovým požadavkům.

Architektura původního řešení byla omezena pouze dvěma faktory. Zaprvé, zdrojová data je třeba exportovat ze SAP ERP, systému na plánování podnikových zdrojů (SAP, 2024). Zadruhé, vizualizace musí být realizována v jednom z nejrozšířenějších vizualizačních nástrojů, v Tableau (Tableau, 2024a).

Zákazník poskytuje BI oddělení širší množství dat v mnoha tabulkách, které jsou v různých vzájemných kombinacích potřebné pro zhotovení dalších reportů. Z tohoto důvodu je výchozím úložištěm dat pro původní projekt rovněž ne SAP, ale zdrojová SQL databáze, kam jsou jednotlivé tabulky s daty ukládány a aktualizovány. Tyto tabulky slouží pouze pro ukládání dat, modifikace dat musí následovat až v dalším kroku.

2.1 Produkt

Každý vyvíjený reportingový nástroj je považován za produkt BI oddělení, který je zhotoven nejdříve pro konkrétní zákazníky. Z tohoto portfolia produktů jsou následně vybírány produkty, u kterých je vysoká míra pravděpodobnosti, že pokud je s nimi vysoká spokojenost u jednoho zákazníka, mohly by se osvědčit i u dalších zákazníků. Z tohoto důvodu časem dochází k rozhodnutí o jejich přepracování do obecné podoby, ve které mohou být nabízeny ve vyšším množství různým zákazníkům. Čas spojený s vývojem a údržbou jednotlivých řešení je faktor, ze kterého oddělení profituje – je tedy cílem oddělení mít co nejvíce poskytovaných dashboardů zákazníkům, které může vyvíjet a udržovat.

2.2 Výhody obecného řešení

Pro co nejvyšší efektivitu v rámci vývoje a údržby dashboardů je výhodné, aby obdobná řešení, která fungují u více zákazníků, byla shodná nejen typologicky svými závěry, ale i fakticky, a to použitou architekturou, obecnými postupy a byla tvořena shodnými, či obdobnými nástrojů – tvorba a údržba daného reportu je pak snazší, celý proces tvorby řešení se může opřít o zkušený nadhled vývojářů v dané problematice, kteří na základě bohatého know-how předem mají představu, jak s daným projektem nakládat.

Hlavní výhodou obecného řešení je úspora z rozsahu při tvorbě a údržbě reportů.

V tomto konkrétním případě poslouží obecné řešení jako základ implementací kapacitního reportingu pro další zákazníky. Na základě přecházejících zkušeností bude k dispozici představa o proveditelné architektuře řešení, bude existovat představa o vzhledu dat, bude existovat představa o vzhledu datového modelu a budou definované programy, se kterými se bude pracovat. Tento základ bude obohacen o uživatelské požadavky. Na základě expertního názoru BI oddělení a předpřipravených podkladů pro tvorbu reportů bude v kombinaci se zákaznickými požadavky report vytvořen rychleji a jeho údržba bude snazší.

Obecné řešení se vzorovými daty poskytuje BI oddělení možnost zákazníkům rovnou představit hotový vzorový report, který je modifikovatelný dle představ a možností zákazníků a jejich systémů. Výhoda tohoto postupu se může ukrývat i v úspoře času při debatách se zákazníky, kteří mnohdy nejsou zcela schopni přesně definovat své požadavky a přání vůči nově vytvářeným reportům.

Zákazníka bude na výsledném produktu předně zajímat správnost dat a smysluplnost výsledné vizualizace, nikoliv samotná architektura našeho řešení či ETL transformující data. Právě z tohoto důvodu lze v obecném řešení nadefinovat tyto pasáže skryté budoucím uživatelům a volněji s vyšší individualitou postupovat právě například ve vizualizační fázi, kde obecné řešení přichází pouze s prototypy designu dashboardů. Za správnost dat jsou zodpovědní vývojáři, nikoliv obecné řešení – shodnost postupů a použitých softwarů v obecném řešení však díky zkušenostem vývojářů přispěje i ke zdárnějším výsledkům a rychlejší práci s daty.

2.3 Nevýhody obecného řešení

Obecné řešení samozřejmě obsahuje i určitá předem známá negativa, či hrozby, které je třeba mít na pozoru. Při dostatečně flexibilní konstrukci obecného řešení však bude možné jejich potenciálním dopadům předcházet.

2.3.1 Odlišnosti jednotlivých implementací

Je předem zřejmé, že obecné řešení může sloužit jako výchozí základ celého řešení, lze také chápat jako opora při tvorbě individuálního řešení, nelze jej však aplikovat striktně bez modifikací u různých projektů – ke každému projektu je třeba přistupovat s určitým individuálním přístupem. Každé vyvíjené řešení bude třeba alespoň částečně modifikovat při jednotlivých implementacích, a to zejména ze tří důvodů:

1. Individuální odlišnost zákaznických systémů obsahujících zdrojová data
2. Individuální odlišnost vzhledu zákaznických dat
3. Individuální odlišná přání vůči vizualizacím (přizpůsobení vizualizace)

Odlišné zdrojové systémy zákazníka neovlivníme, ale v případě obecného řešení alespoň můžeme předcházet komplikacím s konektivitou úvahou a analýzou nad obecně používanými systémy, ve kterých zákazníci užívají data, a na základě této analýzy být připraveni na různá řešení z těch běžněji používaných.

Odlišnost, či rozdílnost zákaznických dat, je hlavním potenciálním rizikem celého obecného řešení. Celé řešení je třeba konstruovat tak, aby se v případě výrazné odlišnosti dat oproti původnímu řešení, na jehož základě se učíme, použití celého obecného řešení nepřinášelo více práce než užitku. Na základě úvah o vzhledu dat se však v obecném řešení budeme snažit definovat obecně shodné prvky struktury dat, kterými by data kapacit výrobních závodů měla disponovat, a na jejichž základě bude možná konstrukce obecného datového modelu. V optimálním případě budeme schopni vždy data pomocí ETL transformovat do námi požadované podoby obecného datového modelu. V opačném případě, kdy toho schopni nebudeme, je třeba mít alespoň obecný model připravený v takové podobě, ve které jeho změna bude intuitivní a snadná.

Odlišnost vizualizací, KPIs a použitých grafů vychází rovněž z analýzy potenciálních vlastností vizualizací a z dostupných dat – opět můžeme předpokládat určitou shodu napříč zákazníky. Oblast zaměření je poměrně úzce vymezená, a tak můžeme bezpečně definovat alespoň základní ukazatele. Okrajové či detailní odlišnosti budou vycházet z odlišnosti dat a zákaznických požadavků, a bude tedy záviset jak na flexibilitě obecného řešení, tak na zručnosti vývojáře, aby bylo požadavkům vyhověno.

Celé obecné řešení je tedy třeba stavět tak, aby poskytovalo určitou flexibilní oporu při tvorbě nových implementací, která nebude nová řešení při vývoji brzdit příliš úzkým vnímáním problematiky.

3. Návrh obecného řešení

Kapitola je věnována teoretickému návrhu designu obecného řešení. Z byznysového pohledu již bylo ustanoveno, že obecné řešení bude nabízeným produktem našeho BI oddělení. Cílem tohoto produktu je vždy aplikovat stejnou základní architekturu, na jejímž základě budeme schopni modifikovat detaily tak, abychom každému zákazníkovi byli schopni produkt přizpůsobit na míru dle jeho konkrétních požadavků. Ačkoliv nelze předpokládat shodu uživatelských požadavků (a to zejména co se vizualizace týče), lze předpokládat alespoň částečnou podobnost zdrojových dat, které navíc pomocí datových transformací budeme schopni přepracovat do ještě shodnější podoby. Lze také předpokládat, že architektura řešení bude primárně v dikci strany vývoje řešení, tedy BI oddělení autora – použité programy k uložení a transformaci dat jsou právě těmi milníky, které jsme schopni v rámci obecného řešení definovat předem.

Na základě hodnocení původního řešení, které bylo aplikováno na míru pouze původnímu zákazníkovi, jsme byli schopni vytyčit základní vlastnosti BI řešení hodnotícího výrobní kapacitu a její aspekty, a to zejména funkční požadavky. V této kapitole rovněž zhodnotíme druhou část požadavků, a to technické možnosti a náležitosti obecného řešení, které nebudou vyplívat z původního řešení, ale ze strategicky vhodných systémů, u kterých lze předpokládat využití naším BI oddělením v budoucnu.

3.1 Datové zdroje

Data jsou získávána přímo ze zákaznických systémů a cílem bude je vždy ukládat do naší SQL databáze, odkud s nimi budeme pracovat. Import probíhá různými metodami v závislosti na systémech zákazníka. Nejčastější tři úložiště dat našich zákazníků jsou popsány níže:

3.1.1.1 Primární zdroj, např. SAP

Pokud zákazníci používají ke své práci systém SAP, lze data napřímo exportovat do SQL databáze. Tento proces však pro autora dokumentu není dostupný, s programem SAP z interních byznysových důvodů není umožněno. Data pro nás do SQL vrstvy budou nahrána a my budeme pracovat s neupravenými daty uloženými v tabulkách naší databáze a dále je zpracovávat.

Exporty dat ze systému SAP jsou prováděna dávkově jednou denně v ranních hodinách pomocí automatických orchestrací, které data exportují přímo do předpřipravených tabulkách v SQL databázi.

3.1.1.2 SQL

Pokud zákazník využívá SQL databázi a my k daným tabulkám máme přístup, budeme data získávat přímo z těchto tabulek. Zde můžeme rovněž očekávat dávkový přírůstek dat jednou denně.

V obou doposud popsaných případech se de facto jedná o shodnou situaci, kde pro naše potřeby začíná celá datová architektura SQL databází, ze které můžeme čerpat data a se kterou už jsme schopni pracovat.

3.1.1.3 MS Excel

Stále velmi rozšířeným nástrojem pro práci s daty je mimo datově orientovaný svět MS excel. Pro nejrůznější číselníky závodů, výrobků, či materiálů můžeme očekávat, že tyto číselníky se mohou vyskytovat v excelovských sešitech. Tyto soubory, např. na MS SharePoint či MS OneDrive, budeme schopni pomocí nástroje KNIME a jeho ETL transformací stahovat napřímo na základě přístupu k nim.

V rámci orchestrace KNIME budeme schopni tato data stahovat v libovolné periodicitě – na základě zkušeností ale lze předpokládat, že data tohoto typu budou skutečně jen zřídka měnící se číselníky, jejichž změna bude zákazníky avizována předem, a tak budeme schopni ji řešit ad hoc jednorázově.

3.2 Datová transformace

Po získání přístupu ke zdrojovým datům se bude naše snaha upínat k načtení, transformaci a uložení dat s cílem přepracovat je do podoby takového datového modelu, který bude vhodný pro tvorbu vizualizací.

K transformaci dat budeme používat výhradně nástroj KNIME, s jehož pomocí budeme z dostupných zdrojů data načítat. Tento software je optimálním nástrojem pro ETL v rámci tohoto projektu – jedná se o nástroj, který je k podobným operacím v rámci BI oddělení běžně využíván, a tedy lze

předpokládat schopnost rychle pochopit veškerý proces i dalšími kolegy. Jeho užití je zdarma. Celá transformační workflow je zároveň přehledně graficky členěná a lze ji obohatit i o nejrůznější poznámky.

Následně pomocí dostupných nástrojů v programu provedeme veškeré datové úpravy, které budou dále v dokumentu v předem dostupných mezích definovány. Tato část architektury řešení je jednou z těch, ve které je možné postupovat zcela individuálními postupy v závislosti na požadavcích či omezeních spojených s konkrétním řešením. Cílem transformace je však dosažení co nejpodobnějšího stavu dat s předdefinovaným obecným datovým modelem tvořeným několika tabulkami, které budou následně nahrány do SQL databáze na místo jejich finálního umístění.

Zamýšlená periodicita aktualizace dat je dávkově každý den, a to pomocí automatizace aktualizace celé ETL workflow v programu KNIME.

3.3 SQL skript, úložiště

Souběžně s ETL v KNIME si pomocí předem připraveného SQL skriptu budeme chtít částečně zobecnit tvorbu tabulek pro uložení výsledných transformací tak, abychom opět dosáhli časové úspory a minimalizovali odlišnosti v postupu při vytváření různých adaptovaných verzí reportů různým zákazníkům.

Cílem bude příprava jednotného skriptu s využitím proměnných hodnot, který bude pomocí SQL uložených procedur schopen na základě definice názvů objektů v datovém modelu vytvořit odpovídající tabulky s danými vlastnostmi jako jsou názvy tabulek, názvy jejich sloupců, jejich datové typy, nenulovost sloupců s klíči atd. Po dokončení datových transformací bude KNIME schopen data do tabulek nahrát.

3.4 Kompilace datového modelu, vizualizace

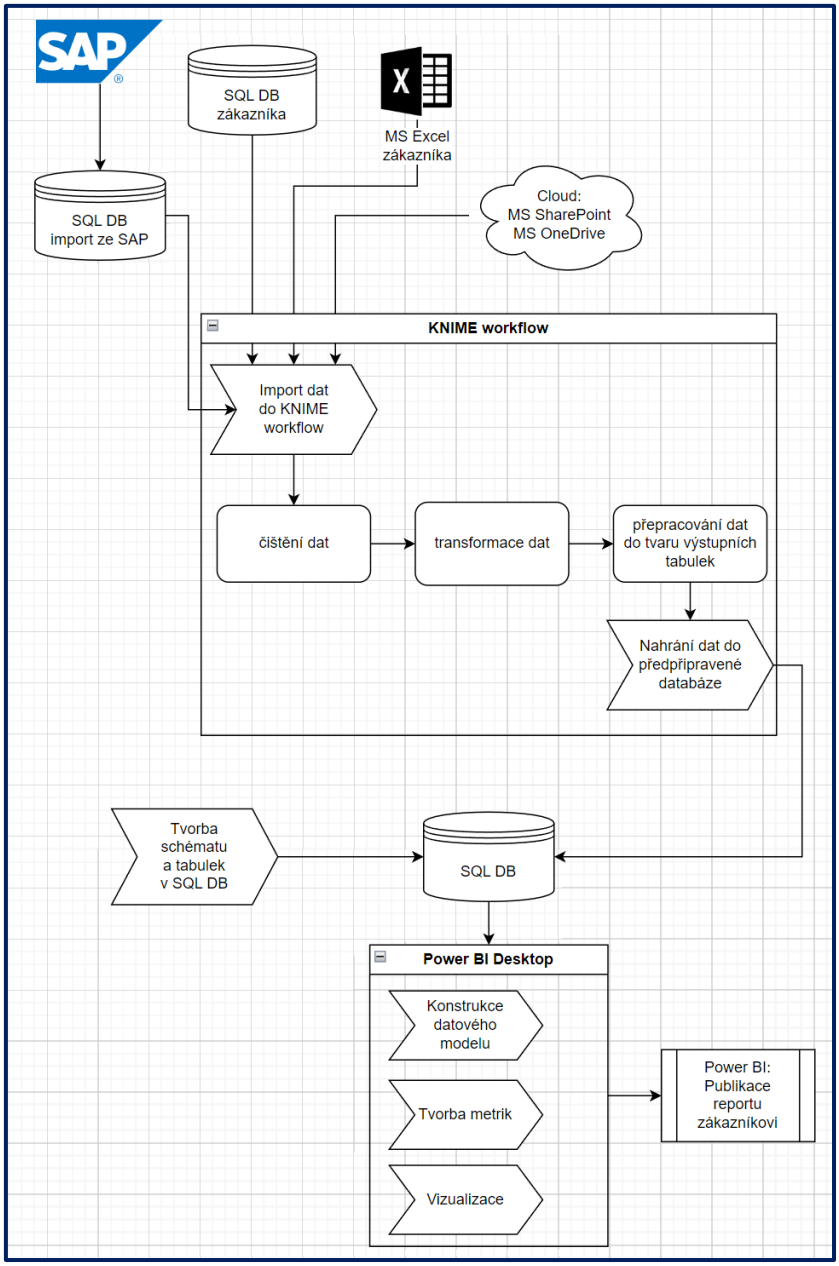
Pro potřeby vizualizace bylo zvoleno Power BI namísto původně využívaného softwaru Tableau, a to zejména díky současné popularitě nástroje jak u vývojářů, tak u zákazníků – s tím jsou spojené i ekonomické důvody, kdy využití Power BI lze předpokládat ve větší míře, než využití Tableau

První fází jeho využití bude import dat z SQL tabulek a tvorba datového modelu přímo ve vizualizačním nástroji, kde lze snadno datový model propojit, případně provádět další úpravy a transformaci dat. V Power BI budeme ve většině případů praktického využití dotvářet metriky pro výpočty KPIs.

Druhou fází využití Power BI je logicky výsledná vizualizace výsledků. V dalších kapitolách se budeme zabývat tvorbou „šablony“ dashboardů, které budeme v rámci obecného vytváření chtít použít a zobrazit, a to opět v takové míře, kterou lze předpokládat jako prospěšnou a podnětnou pro další využití, zároveň však nelimitující volnost nových řešení. Protože se jedná o obecné řešení, můžeme se pokusit nastínit alespoň základní vlastnosti dashboardu, které by v první fázi mohl obsahovat každý konkrétní projekt vzniklý na základě obecného řešení.

Automatizace aktualizace výsledného dashboardu bude rovněž probíhat na denní bázi, a to díky možnostem Power BI Gateway – tedy služby, která umožňuje v Power BI naplánovat plánované aktualizace za pomoci přístupových údajů a specifikace požadované periodicity aktualizace (Iseminger et al., 2023).

3.5 Výsledný návrh



Obrázek 1 – Schéma obecného řešení (autor: vlastní zpracování, https://app.diagrams.net/)

Výsledný návrh architektury obsahuje kompilaci všech popsaných aplikací a úložišť, ve kterých budeme schopni s daty libovolně nakládat za účelem jejich transformace. V každé fázi architektury (zdroje, KNIME, SQL DB, Power BI) budeme schopni data v případě potřeby exportovat do dalších systémů, zálohovat, či odklonit jejich tok jinam.

Model splňuje popsané předpoklady obecného modelu – je vhodným základem pro použití u různých zákazníků, protože poskytuje jednotný přístup k problematice řešení, zároveň jednotlivé fáze mezi sebou obsahují dostatečně variabilní prostor pro modifikaci dat a případné změny u nestandardních, či výrazně odlišných implementací.

4. Tvorba obecného řešení

V této kapitole se budeme zabývat tvorbou obecného řešení, a tedy podkladů pro praktickou aplikaci u budoucích zákazníků. Stěžejním pilířem těchto podkladů je datový model, do něhož musíme přetvořit zákaznická data. Kromě obecného datového modelu, který bude sloužit jako vodítko pro připravované datové transformace, bude třeba vytvořit části architektury, které nám toto umožní. Segmenty, které budeme tvořit, jsou:

- SQL skript pro polo-automatizovanou tvorbu tabulek datového modelu
- Obecnou verzi KNIME workflow pro ETL dat
- Základně naformátovanou verzi reportu v Power BI

4.1 Obsah a typ datového modelu

Nejprve bude třeba nejprve zformulovat, která konkrétní data jsou v rámci současného řešení používána, a to v kontextu struktury závodu a získávaných dat. Pokud tento myšlenkový proces rozšíříme o obecnější úvahu nad strukturou výrobních závodů a datového modelu, který chceme pro takové řešení budovat, budeme schopni vytvořit obecný datový model, který by měl být schopen pokrýt většinu základních otázek ohledně sledování kapacity ve výrobním závodě, a to u jakéhokoliv budoucího zákazníka.

Obecně lze tedy říci, že předpokládáme určitou skupinu vlastností, které se budou v zákaznických databázích s určitou mírou pravděpodobnosti vždy vyskytovat, a které budeme moci použít a zobrazit ve vytvářených reportech. Mezi tyto ukazatele patří:

1. Identifikátory struktury závodu, hierarchické členění (např. závod, oddělení, pracoviště apod.)
2. Identifikátory objektů továrny – členění zaměstnanců, strojů, výrobních linek
3. Seznam vyráběných produktů, jejich hierarchické členění do produktových skupin
4. Dimenze použitých materiálů, případně jejich členění
5. Dimenze času
6. Samotná kapacita – nabídka/poptávka, jejich typy
7. Koeficienty ovlivňující kapacity

Již bylo zmíněno, že nelze předpokládat jakkoli významnou shodu struktury sledovaných dat různých zákazníků. Datový model, který budeme vytvářet, tedy musí být dostatečně obecný a musí být schopen další ukazatele nejen přijmout, ale zároveň být schopen pracovat i v případě absence naopak jiných ukazatelů. Jak proveditelný je to předpoklad bude záviset zejména na kreativitě při tvorbě jednotlivých datových transformací.

Model musí dále splňovat dva základní předpoklady:

1. Intuitivní a jednoduché propojení jednotlivých tabulek modelu
2. Vysoký počet potenciálních ukazatelů, které je možné přidat do modelu

Na první z těchto požadavků bude odpovědí přehledná organizace dat do schématu, které bude propojeno vazbami ideálně kardinalit 1:1 či 1:N (např. když jedno zařízení bude obsluhováno více zaměstnanci, či jedna produktová kategorie bude výlučně zodpovědná za N produktů). Jasně vymezené a přehledné dělení do dimenzí a tabulek faktů je samozřejmostí.

Pro naplnění druhého z předpokladů je podstatné zohlednit jednotlivé typy struktur datových modelů a jednotlivé typy dimenzionálních tabulek.

4.1.1.1 Tabulka faktů

Tabulka faktů obsahuje měřitelné údaje. Zpravidla obsahuje nižší počet sloupců a vysoký počet záznamů, které jsou zpravidla uváděny v nejnižších dostupných jednotkách granularity, kterou jsme schopni sledovat. Sloupce, které neobsahují měřitelná data, zpravidla obsahují kategoriální informace vedoucí ke členění měřitelných výsledků. K jednotlivým identifikátorům lze připojovat dimenze nadřazených struktur, pokud je máme k dispozici. Pokud k dispozici žádná další nadřazená dimenze není, ale stejně dle lze sloupce výsledky klasifikovat, hovoříme o degenerované dimenzi v rámci tabulky faktů.

Pokud v datovém modelu existuje více než jedna z tabulek faktů, může to být dáno rozdílnou granularitou či nekompatibilitou záznamů.

4.1.1.2 Dimenzionální tabulka

Neboli dimenze, obsahuje kategorické informace, pomocí nichž můžeme členit jednotlivé záznamy v tabulce faktů. Tyto číselníky mohou být připojeny přímo k tabulce faktů (schéma STAR), nebo ke kategorii v jiné dimenzi (schéma SNOWFLAKE).

Dimenze může dále být degenerovaná přímo v tabulce faktů. Naopak dimenze, která je propojena s tabulkou faktů pomocí jiné dimenze, nazýváme referenční dimenzí.

4.1.1.3 Schéma STAR

Ve schématu STAR jsou všechny dimenze přímo připojeny k tabulce faktů. Preferované schéma našeho obecného modelu.

4.1.1.4 Schéma SNOWFLAKE

Odlišuje se od schématu STAR možností připojení dimenzí nikoliv k tabulce faktů, ale k dalším dimenzím. V tomto schématu se mohou vyskytovat referenční dimenze.

4.1.1.5 Scénáře rozšíření datového modelu

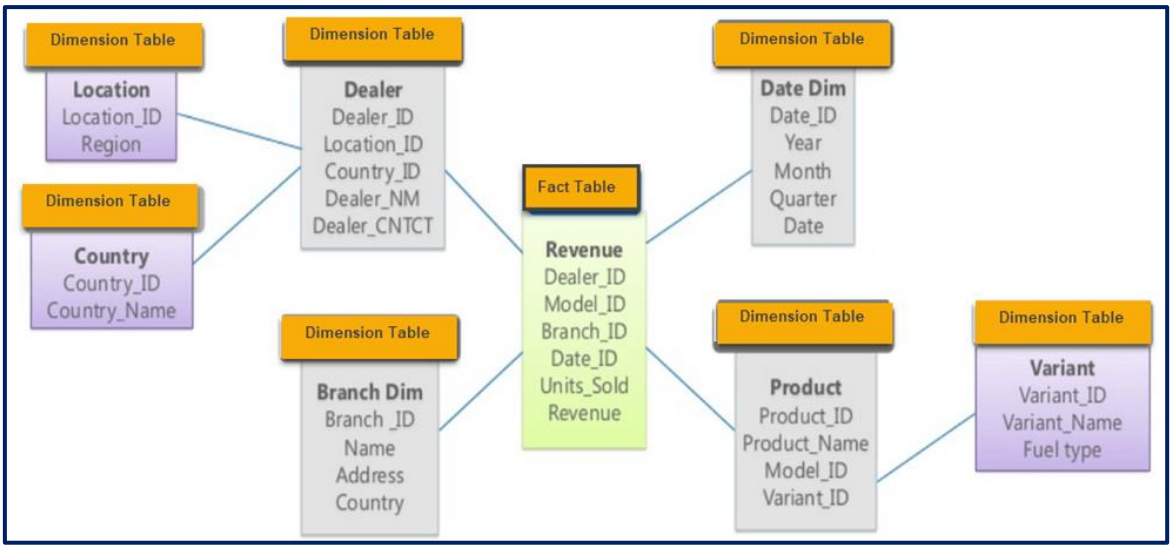
Pokud budeme chtít základní datový model obohacovat o dodatečné informace, je nutné rozlišit, zda se jedná o nový úhel pohledu na data (tedy novou dimenze), či jen o dodatečný stupeň členění dat v již existujících dimenzích.

Pokud budeme data obohacovat nad rámec základního datového modelu o nový úhel pohledu, který disponuje alespoň dvěma úrovněmi, bude vhodné alokovat ho do nové dimenze. Jednoúrovňové členění by bylo sobě samotné klíčem, nemělo by smysl ho vyčlenit do vlastní dimenze, a budeme ho tedy držet v rámci tabulky faktů jako degenerovanou dimenzi.

Při rozšiřování již existujících dimenzí o nové hierarchické úrovně záleží především na počtu těchto úrovní – opět můžeme nové úrovně ponechat v již existujících dimenzích (a tedy tvořit degenerované dimenze v dimenzích). Toto je velmi efektivní z pohledu rychlosti výpočtů v tabulkách, jelikož mezi údaji různé hierarchické úrovně odpadá nutnost JOIN vazby pro získání informací o dalších úrovních, na druhou stranu při častých změnách hierarchických struktur jsou změny dat komplikovanější, protože jedna změna se musí propsat do vyššího počtu řádků.

Naproti tomu vyčlenění nových úrovní do samostatných dimenzí ve tvaru schématu SNOWFLAKE je výhodné, pokud dochází k častým změnám v hierarchickém uspořádání a závislosti prvků. Zároveň také dodržuje pravidla normalizace dat, tedy snahu o minimální redundanci dat. Mezi původními dimenzemi a novými, ve vyšších úrovních, fungují vazby M:1.

Vzhledem k nutné transformaci dat, které budeme do tohoto modelu teprve nahrávat pomocí ETL procesů, bude třeba zvážit tento design modelu v souvislosti s jeho nižší přehledností. Zároveň při nižším počtu záznamů, např. do milionu či nižších jednotek milionů řádků v tabulkách, bude použití SNOWFLAKE schématu spíše práce s nevýznamnou úsporou výpočetních sil.

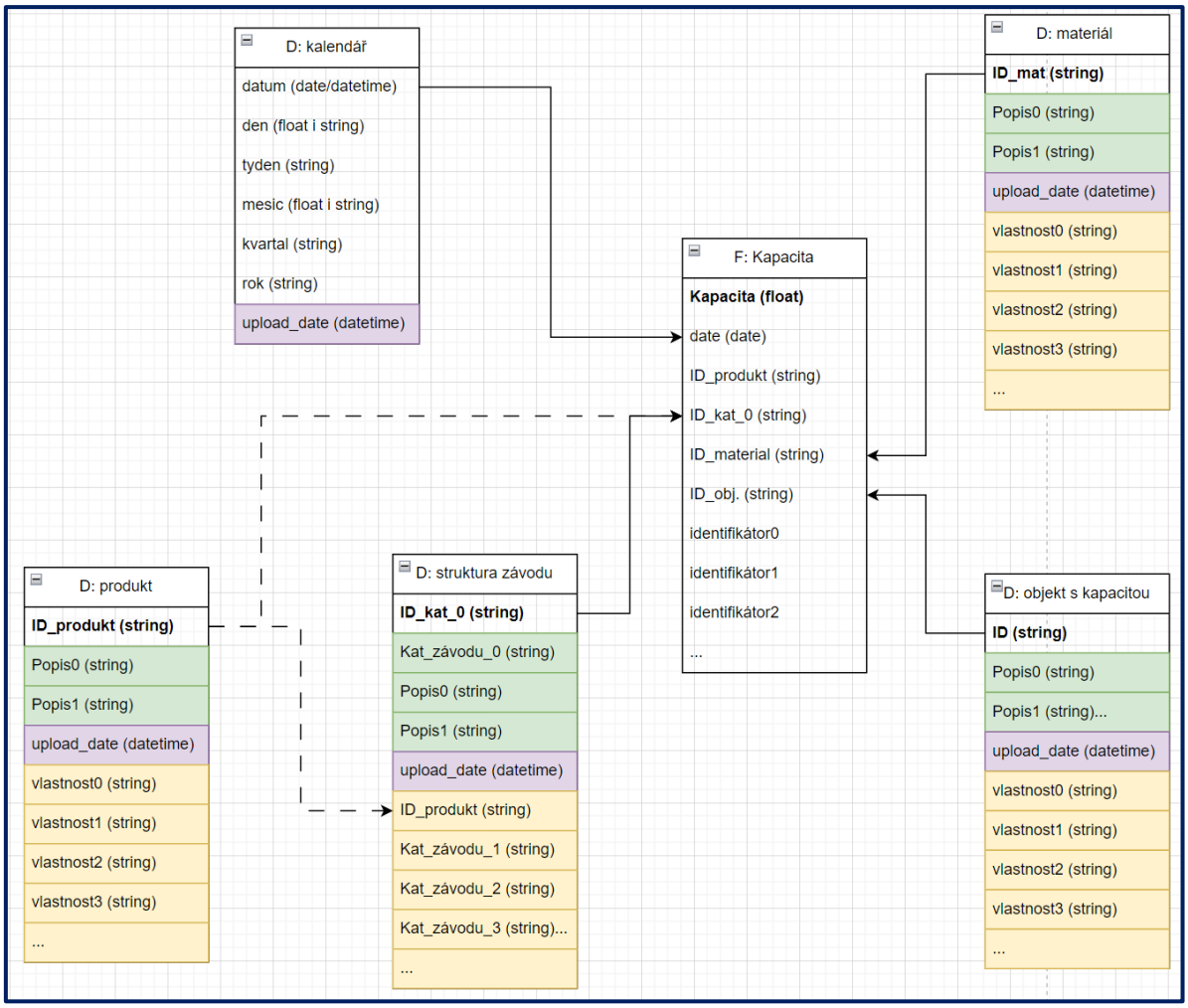


Obrázek 2 – Schéma SNOWFLAKE (autor: Taylor, 2023)

Obrázek 2 znázorňuje jednoduché schéma typu SNOWFLAKE, tedy schéma inspirované tvarem postupně k okrajům větvící se sněhové vločky. Kolem tabulky faktů, která tvoří střed schématu, jsou zaznačeny šedě podbarvené dimenze. Pokud by těmito dimenzemi model končil (z pohledu od faktové tabulky k šedivě podbarveným dimenzím), jednalo by se o schéma typu STAR, tedy základní schéma datového modelování, při kterém je každá dimenze s tabulkou faktů propojena klíčem. Nadstavbou k těmto šedivě podbarveným dimenzím jsou fialově podbarvené dimenze, které z modelu tvoří právě schéma sněhové vločky. Můžeme si například na horním levém rohu dimenzí Location a Country, které rozšiřují model o další informace jako nadstavba dimenze Dealer. Tyto dimenze lze rovněž označit jako referenční dimenze – s tabulkou faktů jsou propojeny pomocí dimenze Dealer (Pour, Maryška, Stanovská, Šedivá, 2018, s. 40).

4.2 Konceptuální návrh datového modelu

V předešlých částech dokumentu byl podrobně popsán původní datový model, resp. proces datové transformace vedoucí ke třem datovým výstupům pro vizualizaci původního reportingového řešení pro konkrétního zákazníka. V popsaných SQL pohledech figurovalo mnoho identifikátorů a dotvářených hodnot za účelem možnosti kategorizace dat ve výsledném reportu. Na základě těchto datových výstupů je tvořen koncept datového modelu pro univerzální použití.



Obrázek 3 – Konceptuální návrh obecného datového modelu (autor: vlastní zpracování)

Při konstrukci obecného datového modelu, který funguje na bázi relačního schématu, je v jednoduchých případech cílem dodržení pouze jedné tabulky faktů, tj. všechny měřitelné hodnoty jsou uvedeny v jedné tabulce.

Naznačené přerušované vazby budou dále vysvětleny v textu věnujícímu se dimenzi produktu. V obecné podobě modelu uvažujeme několik dimenzí vyplývajících ze zdrojových zákaznických dat, které by měly postačit pro pokrytí základních reportingových potřeb. Pro optimalizaci reportingu v rámci času rovněž počítejme s tvorbou uměle vytvořené kalendářní dimenze v nástroji Power BI. V případě nutnosti nebude problém vytvořit další dimenze a pomocí jedinečného identifikátoru a klíče k tabulce faktů připojit. Dodatečné nadřazené úrovně granularity bude možno přidat buď pomocí jednotlivých sloupců v již existujících dimenzionálních tabulkách, či pomocí dodatečných tabulek, a to dle již popsané metodiky tvorby schématu SNOWFLAKE.

Jelikož je cílem tohoto dokumentu obecné řešení vhodné k aplikaci, pomocí které budou schopni zákazníci analyzovat výrobní kapacitu a její dílčí složky, je vhodné vymezit klíčové obecné cíle BI, které nelze opomenout při tvorbě tohoto datového modelu.

Flexibilita dat

Uživatelé se na řešení a data poskytnutá v rámci vizualizace mohou obracet s nejrůznějšími dotazy, které mohou být velice úzce profilované. Je zapotřebí mít k dispozici dostatek vhodných atributů pokrývající vlastnosti, které mohou posloužit k filtrování datové sady. Tyto filtry by pak měly být schopny zobrazit požadovaná data jednoduše, přehledně, a co nejnižší složitostí interpretace výsledků. Právě z tohoto důvodu je tabulka faktů potenciálně tabulkou celého datového modelu co do počtu sloupců, neboť právě ona bude nositelkou všech možných atributů, které budou dále rozvinuty pomocí k ní připojených dimenzí.

Granularita dat

Pokud data budeme chtít zkoumat z určitého detailního pohledu, jistě by nám nestačilo tento detailní požadavek získávat z dat agregovaných za celé kalendářní měsíce. Granularita dat, tedy úroveň detailu poskytnutá v datech, nám s rostoucí mírou dovoluje zobrazovat data detailněji, což logicky umožňuje i analýzu detailních rozdílů v datech. Každá dimenzionální tabulka by měla vždy v každém konkrétním případě obsahovat pochopitelně primární klíče, které budou v tabulce faktů využity jako cizí klíče, a dále další hierarchické hodnoty či hodnoty dále upřesňující strukturu dané dimenze. Zde platí, že primární klíč by měl být hodnotou s nejnižší granularitou členění, jelikož se v dimenzionální tabulce a daném sloupci nemůže atribut stejné hodnoty vyskytovat vícekrát (Pour, Maryška, Stanovská, Šedivá, 2018, s. 31).

Počet jednotlivých sloupců, identifikátorů, popisných sloupců a hierarchických úrovní se v každé dimenzi může měnit u každého zákazníka. Z tohoto důvodu jsou v jednotlivých tabulkách (Obrázek 3) znázorněny potenciální další sloupce znázorněné třemi tečkami.

4.2.1.1 Tabulka faktů

Samotná tabulka faktů by měla obsahovat sloupce s měřitelnými hodnotami nabízené a poptávané pracovní kapacity. Zároveň by kapacita, jakožto jediný kvantifikovatelný prvek této tabulky faktů, měla být obsahem jednoho sloupce. V rámci modelování bude přínosnější daný typ kapacity vždy identifikovat dodatečnými potřebnými sloupci s jejími identifikátory než vytvářet sloupec pro každý z typů kapacity. V případě nutnosti zacházet s hodnotami speciálním způsobem počítáme s možností dotvoření dodatečných sloupců či metrik v Power BI pomocí jazyky DAX a logických podmínek filtrujících hodnoty.

Dále by měla tabulka faktů obsahovat cizí klíče k jednotlivým dimenzionálním tabulkám a poté předem neurčitý počet sloupců pro jednotlivé identifikátory záznamů. Tyto identifikátory budou sloužit nejen k identifikaci jednotlivých záznamů tabulky faktů, ale také mohou sloužit pro propojení s jednotlivými dimenzemi. Jakožto identifikátor záznamů mohou sloužit samy o sobě každý zvlášť, nebo v případné kombinaci např. číslo materiálu + konkrétní datum. Na základě původního řešení a jeho vlastností lze rovněž předpokládat, že tabulka faktů bude obsahovat některé sloupce, které budou ve své podstatě formovat tzv. degenerované dimenze – ačkoliv dle nich bude možné členit tabulku faktů, sloupec s hodnotami bude existovat bez jakékoliv dimenze, které by sloužil jako klíč pro propojení s tabulkou faktů.

Tvorba samotné tabulky faktů bude nejsložitější činností v rámci přípravy dat do požadovaného formátu, neboť v ní budeme potřebovat sloučit data z různých zdrojů a to tak, aby byly vyjádřeny korektně včetně náležitých příslušností a se správným množstvím nabízené a poptávané kapacity.

4.2.1.2 Dimenze hierarchické struktury závodu

Tuto tabulku bude pravděpodobně obsahovat každé řešení, jelikož můžeme prakticky vyloučit, že by výrobní závod nedisponoval žádnou hierarchickou strukturou členění dle výrobních jednotek či dle výrobků zařaditelných pod jednotlivé výrobní linky apod. Toto členění můžeme různě pojmenovávat a členit na výrobní linky, nákladová centra, výrobní oddělení apod. Nejnižší hierarchickou jednotkou pravděpodobně bude ta jednotka, do které budeme schopni v tabulce faktů členit kapacitu, příp. do které bude relevantní sledovat kapacitu.

4.2.1.3 Dimenze kalendáře

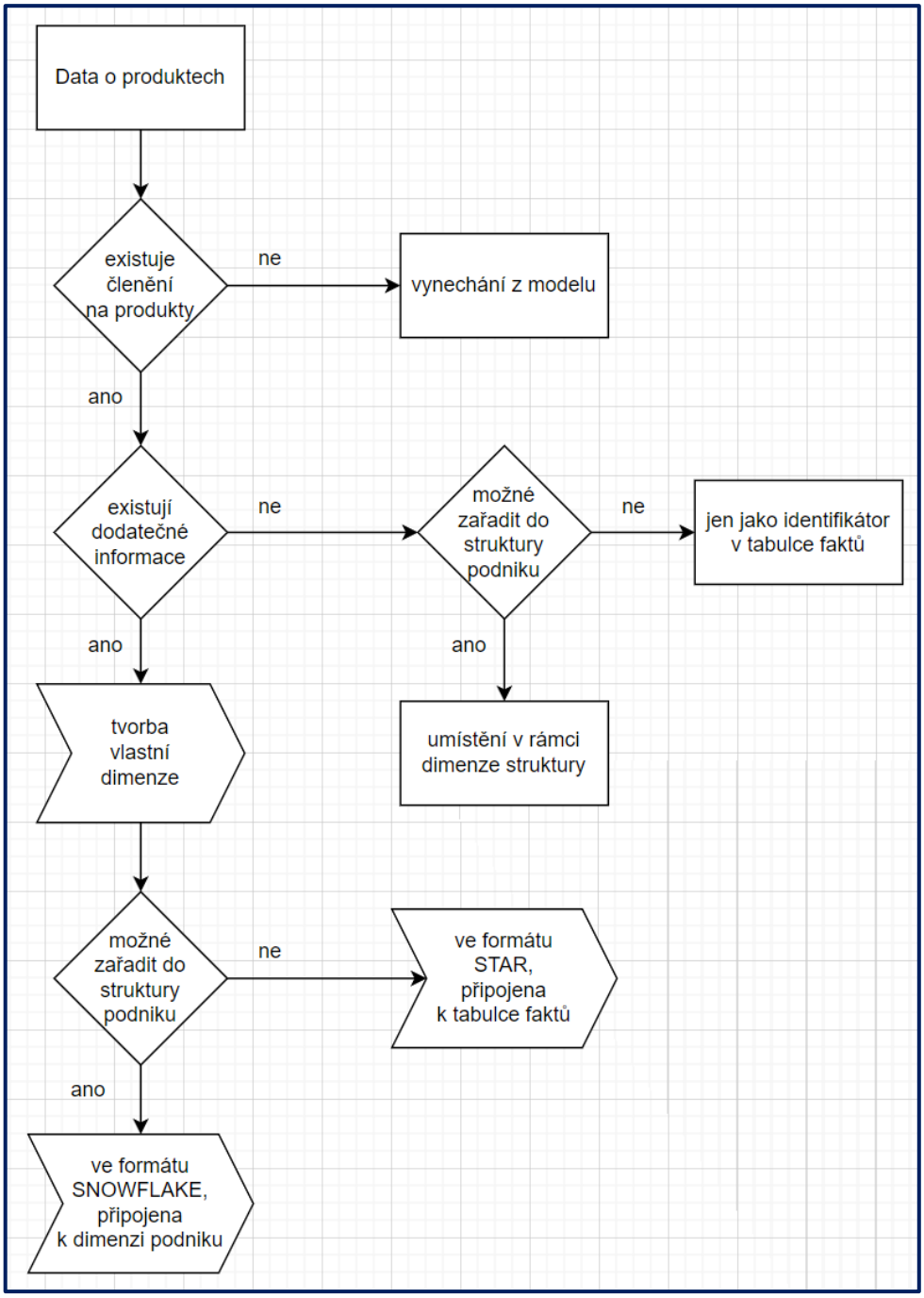
Tuto dimenzi bychom z praktických důvodů měli dotvořit i v případě, že ji nebudeme moci sestavit z poskytnutých dat. Její význam bude v Power BI, a to k efektivnějšímu filtrování a zobrazování dat v čase, časových jednotkách a intervalech. Jelikož jsou časové údaje a příslušné názvy dní, měsíců apod. neměnné, odpadá nám potřeba jakkoliv držet na paměti potenciální potřebu aktualizace či změn této dimenze, pokud ji předem vytvoříme např. s pomocí DAX funkce CALENDARAUTO(), která je schopna automaticky pokrýt celé sledované období.

4.2.1.4 Dimenze produktu

Tato dimenze by měla primárně zobrazovat dodatečné informace o produktu, na jehož vyrobenou jednotku bude třeba X jednotek kapacity. V původním řešení figuruje produkt jako jedna z hierarchických úrovní struktury závodu, protože každému produktu lze přiřadit jedna či více podřízená jednotka struktury, ve které probíhá výroba výhradně tohoto produktu.

Na tomto příkladu je vhodné ilustrovat možnost procesu rozhodování, zda obecný datový model upravíme do podoby SNOWFLAKE připojením vlastní dimenze produktu na dimenzi struktury závodu, nebo o tyto informace pouze obohatíme dimenzi struktury závodu a zůstaneme tedy u schématu STAR, tedy jen jedné dimenze, která je připojena přímo na tabulku faktů.

Dle popsaných možností je vždy třeba zvážit danou kombinaci možností, která bude determinovat samotnou existenci a případné propojení dimenzionální tabulky věnované produktům s dalšími objekty konkrétní verze datového modelu. Z tohoto důvodu lze na schématu celého zamýšleného obecného modelu (Obrázek 3) vidět dvě potenciálně možná naznačená propojení s dimenzí struktury závodu a s tabulkou faktů.



Obrázek 4 – Rozhodovací strom pro řešení otázky dimenze produktů (autor: vlastní zpracování)

Celý rozhodovací proces, který je v tomto příkladu vztažený k potenciální existenci dimenzí produktu a závodu, lze samozřejmě vztahovat na všechny potenciální existující dimenze.

4.2.1.5 Dimenze materiálu

Účelem této dimenze je možnost poskytnout členění materiálů zpracovávaných ve výrobním závodě a kapacit, které je na ně potřeba vynaložit. Pokud každá jednotka materiálu je spotřebována za použití X jednotek kapacity, jsme schopni v reportu zobrazovat různou spotřebu různých materiálů. Pokud data obsahují tyto informace, zcela jistě budeme schopni zahrnout informace o těchto materiálech do tabulky faktů. Pokud budeme mít k dispozici další údaje, díky kterým bychom mohli materiály kategorizovat a shlukovat do skupin, nabízí se pro tyto účely vytvořit samostatnou dimenzi.

4.2.1.6 Další dimenze

Do modelu můžeme přidávat další libovolné dimenze, jejichž základní objekty budou propojeny s tabulkou faktů. Tyto objekty budou mít přiřazenou kapacitu obdobně jako materiály, budeme je též moci různě klasifikovat. Jako příklad můžeme v tomto případě použít jednotlivé zaměstnance, či stroje na daných pracovištích – tyto objekty mohou být přiřaditelné jednoznačně či nejednoznačně k jednotlivým pracovištím a reporting může vyžadovat zohlednění možnosti filtrovat mezi těmito objekty. Dimenze nebude obtížné přidat díky popsanému SQL kódu v dalších kapitolách.

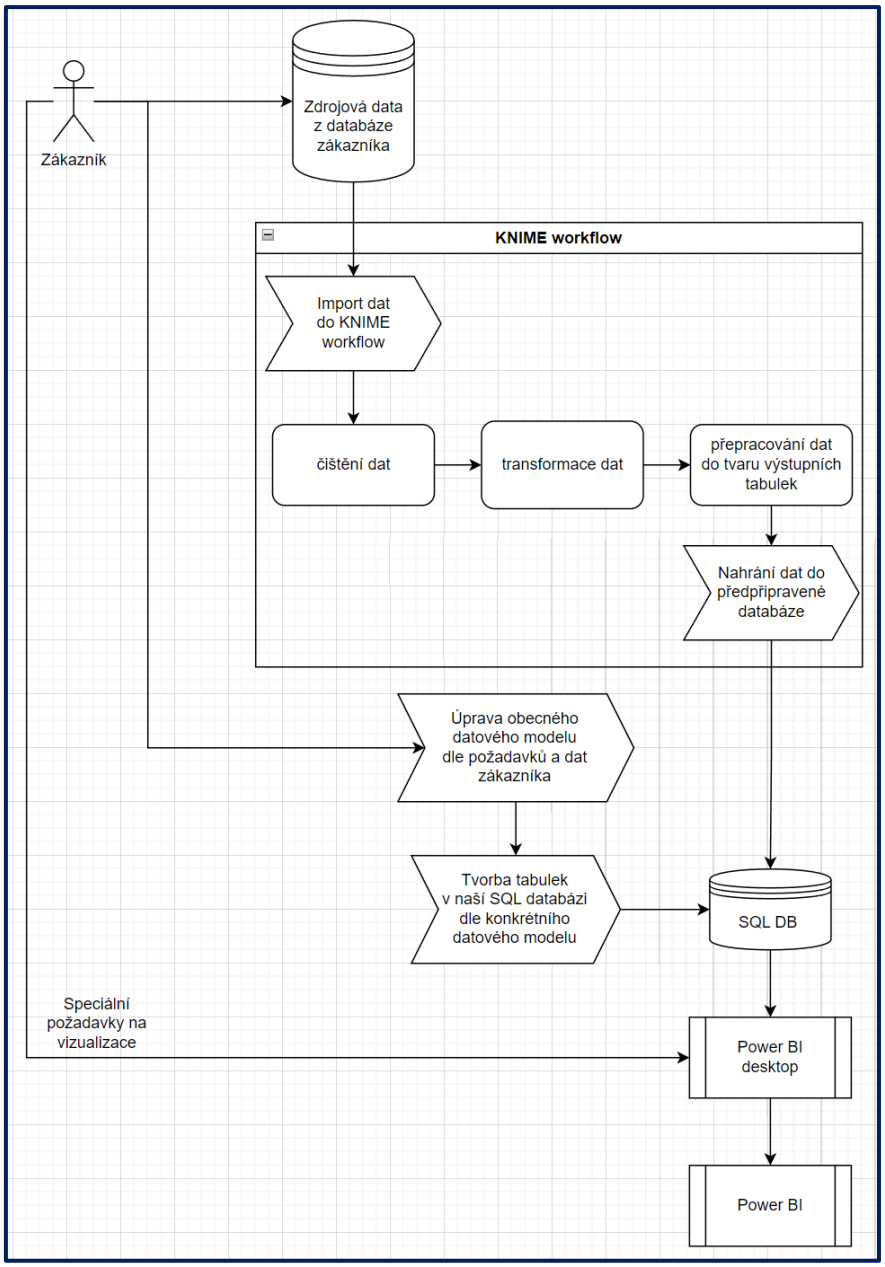
4.2.1.7 Degenerované dimenze

Také do modelu mohou vstupovat další vlastnosti, které budeme evidovat přímo v rámci tabulky faktů, a které již nebude možné dále strukturovaně členit ve vlastní dimenzi. Samotný atribut je tedy zároveň jak klíčem takové dimenze, tak také jedinou její úrovní členění. Jako příklad z původního řešení lze použít koeficienty upravující výši nabídky (AVG FY18-20 apod.), které není třeba vyčlenit do zvláštní dimenze, jelikož by v ní figurovali jako jediný sloupec hodnot.

4.3 Datová architektura a její obecné prvky

Tato část popisuje tvorbu jednotlivých prvků obecného řešení. Vytvořené části řešení budou k využití při každé pozdější implementaci řešení. Schéma procesu implementace je uvedeno na obrázku níže, viz Obrázek 5. Toto schéma je konkrétnější verzí obecného postupu (viz Obrázek 1) – oproti němu se liší o konkrétní kroky, které budou obecné řešení adaptovat konkrétním potřebám.

K tvorbě funkčního řešení modelu budou použity technologie KNIME a Microsoft SQL Server – tedy technologie, které jsou v rámci našeho oddělení běžně k těmto operacím používány.



Obrázek 5 – Diagram procesu tvorby konkrétního řešení pro budoucí zákazníky (autor: vlastní zpracování)

4.3.1 SQL skript

Při konkrétní implementaci řešení budou v KNIME workflow transformována zdrojová zákaznická data do podoby shodné s konkrétním datovým modelem. V SQL databázi budeme chtít během průběhu datové transformace vytvořit prázdné tabulky, které budou naplněny daty na závěr KNIME workflow. Vzhled těchto tabulek (struktura, počet sloupců, datové typy) budou vycházet z konkrétního datového modelu, a my je budeme schopni zjednodušeně vytvořit pomocí předpřipraveného skriptu.

V tomto skriptu budeme počítat s výrobou vlastního schématu a tabulek tak, jak popsal konkrétní datový model, a to záměrně s potenciálně vyšším počtem vytvářených sloupců než menším, protože ve skriptu bude jednodušší sloupce nevytvořit, než dopisovat a dotvářet nové sloupce.

Skript se opírá o funkce DECLARE, SELECT, EXEC a na využití proměnných. Jeho cílem bude v několika úvodních částech skriptu nadefinovat potřebné názvy budoucího schématu, tabulek, jednotlivých sloupců v daných tabulkách a vlastnosti každého
sloupce – jeho datový typ a případně i podmínky, zda se v tomto sloupci mohou vyskytovat *null* hodnoty. Díky užití proměnných budeme schopni takto ve stručnosti definovat vzhled skriptu a zajistíme, že při jednotlivých implementacích řešení bude mezi skripty, a tedy i designy jednotlivých datových modelů, panovat určitá shoda, čímž opět ušetříme čas v budoucnu při potřebě upravovat a spravovat architektury reportů různých klientů.

Skript bude členěn následovně:

1. Tvorba proměnných určených pro názvy schématu, tabulek, datových typů, proměnných nezbytných ke spuštění celého skriptu, jednotlivých sloupců
2. Definice konkrétních vlastností všech proměnných určených pro jednotlivé tabulky a jejich sloupce
3. Kompilace proměnných do vyvolatelného vloženého skriptu
4. Spuštění skriptu a vytvoření tabulek dle nastavených specifikací

Při každé iteraci řešení budou vyžadovány minimální zásahy do první fáze, druhá fáze bude místem pro definování datového modelu, třetí a čtvrtá fáze budou opět prakticky bezúdržbové.

Pro funkčnost tohoto skriptu bude třeba mít práva na úpravu objektů v databázi, v tomto případě pro tvorbu schémat a tvorbu tabulek.

4.3.1.1 Tvorba proměnných

V zájmu minimalizovat potřebu opakovaně upravovat vysoké množství proměnných, budeme pouze na několika místech v textu definovat vlastnosti proměnných, které se budou ve skriptu opakovaně vyskytovat.

Pro názvy databází a tabulek volíme delší řetězec libovolných znaků do délky 100 znaků.

```
DECLARE

-- ČÁST PRVNÍ - NASTAVENÍ DATOVÝCH TYPŮ VŠECH PROMĚNNÝCH
            -- Není třeba měnit
            -- Existence schémetu, použitých tabulek

                 @SchemaName nvarchar(100)

                ,@FactTableName nvarchar(100)
                ,@DimTable0Name nvarchar(100)
                ,@DimTable1Name nvarchar(100)
                ,@DimTable2Name nvarchar(100)
                ,@DimTable3Name nvarchar(100)

            -- Proměnné spouštějící celý kód - neměnit
                ,@SQLcommand_schema nvarchar(100)
                ,@SQLcommand_tables nvarchar(max)

            -- Datové typy a jejich definovaná minimální délka pro funkčnost
                ,@float nvarchar(9)
                ,@int nvarchar(3)
                ,@varchar nvarchar(11)
                ,@nvarchar64 nvarchar(12)
                ,@nvarchar128 nvarchar(13)
                ,@date varchar(4)
                ,@datetime varchar(8)

            -- Proměnné pro definici existence a typu jednotlivých sloupců
                ,@null nvarchar(4)
                ,@not_null nvarchar(8)
                ,@not_null_unique nvarchar(15)

    -- Názvy sloupců, jejich datové typy, vlastnosti sloupců
        -- Není třeba měnit
            -- Nejprve tabulka faktů (1), pak jednotlivé dimenzionální tabulky (4, indexováno od 0)

            -- Tabulka faktů
                ,@Col0Name nvarchar(50)
                    ,@Col0Type nvarchar(13)
                        ,@Col0_ nvarchar(15)
                ,@Col1Name nvarchar(50)
                    ,@Col1Type nvarchar(13)
                        ,@Col1_ nvarchar(15)
                ,@Col2Name nvarchar(50)
                    ,@Col2Type nvarchar(13)
                        ,@Col2_ nvarchar(15)
```

Obrázek 6 – Deklarace proměnných ve skriptu (autor: vlastní zpracování)

@SQLcommand_schema/tables jsou proměnné, které jsou pro celý skript klíčové. Budou do nich umístěny všechny ostatní proměnné v takové syntaxi a pořadí, aby pomocí jejich spuštění došlo k vytvoření všech objektů databáze. Pro první z nich postačí délka 100 znaků, pro druhou je ale klíčová zvolená délka *nvarchar(max)*, která bude poskytovat dostatečně variabilní maximální délku, kterou nelze vzhledem k obecnosti modelu předpokládat do určitého přesného intervalu.

Dále je třeba nadefinovat jednotlivé proměnné pro jednotlivé datové typy. Těmto proměnným prozatím stanovujeme přípustnou délku budoucích textových řetězců tak, abychom do těchto proměnných mohli dosadit samotné upravitelné definice datových typů, viz 2. část skriptu.

Na spodní části výstřižku kódu je zobrazena struktura záznamu pro samotnou tabulku faktů – v tomto stylu se bude obecně opakovat kód pro tabulku faktů i všechny dimenzionální tabulky – pro každý sloupec dosazení tří informací: názvu sloupce, který může být delší textový řetězec, následně datový typ daného sloupce, závěrem třetí atribut určený ke specifikaci potenciálních hodnot, kterých může sloupec nabývat (*null* atp.). Sloupce v tabulkách, ale i samotné tabulky, jsou indexovány od nuly. Kromě vizuálního aspektu to nenese žádný význam a sloupce by bylo možné číslovat, nazývat a řadit libovolně.

4.3.1.2 Definice konkrétních vlastností

V této části kódu se nejprve budeme zabývat proměnnými, které budou sloužit k určení datového typu. Na základě stanovených délek textových řetězců (předcházející krok) můžeme nyní jednotlivým proměnným přiřadit takový textový řetězec, který bude po spuštění skriptu chápán programem jako funkce pro definici datového typu – např. textový řetězec *‚NVARCHAR(128)‘* bude v další části kódu označovat textová pole s potenciální dlouhou řadou znaků (může být využit například pro označení specifických materiálů, či jiných objektů s dlouhými názvy). Při srovnání s první částí skriptu si lze povšimnout, že pro tuto proměnnou jsme zvolili její textovou délku až 13 znaků, což přesně odpovídá délce textového řetězce *‚NVACHAR(128)‘.*

```
-- ČÁST DRUHÁ
    -- URČENÍ SPECIFICKÝCH HODNOT PROMĚNNÝCH DLE DATOVÉHO MODELU

            SELECT
            -- Nastavení datových typů
            -- Není třeba měnit
                 @float = 'float(30)'
                ,@int = 'INT'
                ,@varchar = 'VARCHAR(32)'
                ,@nvarchar64 = 'NVARCHAR(64)'
                ,@nvarchar128 = 'NVARCHAR(128)'
                ,@date = 'DATE'
                ,@datetime = 'DATETIME'

                ,@null = ''
                ,@not_null = 'NOT NULL'
                ,@not_null_unique = 'NOT NULL UNIQUE'

            SELECT
            -- Nastavení databázových údajů
                -- Název schématu
                 @SchemaName = '[rycl00]'

                -- Názvy jednotlivých tabulek
                ,@FactTableName =  '[F_tabulka]'        -- tabulka faktů
                ,@DimTable0Name =  '[D_produkt]'        -- dimenze 0
                ,@DimTable1Name =  '[D_struktura]'      -- dimenze 1
                ,@DimTable2Name =  '[D_objekt]'         -- dimenze 2
                ,@DimTable3Name =  '[D_material]'       -- dimenze 3
```

Obrázek 7 – Nastavení datových typů a názvů objektů v rámci proměnných (autor: vlastní zpracování)

Výčet nabízených datových typů by měl postačovat potřebám našeho obecného řešení. Pro textové záznamy jsou nabízeny proměnné v kratší i delší verzi, pro záznam data a času předpokládáme dvě proměnné, pro záznamy číselných údajů je rovněž několik proměnných k dispozici. V případě potřeby dotvoření další proměnné je zřejmé, jak na to po vzoru předchozích dotvořených proměnných.

Lze si povšimnout jakým způsobem je řešena proměnná *null* – v později sjednoceném textovém řetězci bude tato proměnná velice užitečná, protože bude dosazovat prázdné uvozovky, tedy žádný text, a to na pozici proměnných ve všech pozicích, které nebude třeba obsadit (např. předpřipravený skript počítá s deseti plně předdefinovanými sloupci, my jich budeme na základě konkrétního datového modelu potřebovat jen šest).

Dále je třeba nadefinovat konkrétní cestu pro umístění do databáze, a to název schématu a názvy budoucích tabulek – pro potřeby obrázku zvoleny ilustrační názvy.

```
SELECT
-- Nastavení názvu, datového typu a vlastností (null / not null / not null unique) sloupců v tabulkách
    -- Nejprve tabulka faktů (1), pak jednotlivé dimenzionální tabulky (4, indexováno od 0)

    -- Tabulka faktů
         @Col0Name = 'Sloupec1'
        ,@Col0Type = @float
            ,@Col0_ = @not_null_unique

        ,@Col1Name = 'Sloupec2'
        ,@Col1Type = @float
            ,@Col1_ = @not_null

        ,@Col2Name = 'Sloupec3'
        ,@Col2Type = @datetime
            ,@Col2_ = @null

        ,@Col3Name = 'ID_produkt'
        ,@Col3Type = @nvarchar128
            ,@Col3_ = @null

        ,@Col4Name = 'ID_kat_0'
        ,@Col4Type = @nvarchar64
            ,@Col4_ = @null
```

Obrázek 8 – Nastavení jednotlivých sloupců v tabulce (autor: vlastní zpracování)

Nejdelší částí celého skriptu je oddíl věnující se definicím sloupců v jednotlivých tabulkách – pro každou tabulku je v základní verzi obecného skriptu alokováno 10 sloupců, u nichž je u každého možnost zvolit název, datový typ a vlastnosti *null, not null,* případně *not null unique*. Každá vytvářená tabulka je definována tímto předpisem, tedy trojicí proměnných pro každý vytvářený sloupec (proměnné definovány dle předcházejících částí kódu, viz *Obrázek 6* a

Obrázek 7).

Opět, tabulka faktů i všechny dimenzionální tabulky jsou tvořeny tímto způsobem. Rovněž lze spatřit využití proměnné *@null*, která je využita při nepotřebnosti sloupce. Obrázek 8 obsahuje ilustrační konfiguraci sloupců s různými vlastnostmi.

4.3.1.3 Kompilace proměnných do skriptu

Proměnná *@SQLcommand* je automaticky osazena textem využívajícím všechny doposud nadefinované informace tak, abychom tento úryvek kódu vůbec nemuseli upravovat. Dochází zde nejdříve k tvorbě schématu, které ponese unikátní název pro pojmenování projektu, např. dle daného zákazníka. Dále již budou vytvářeny všechny tabulky, a to v blocích, z nichž je jeden zobrazen na výstřižku výše.

```
-- ČÁST TŘETÍ
    -- TRANSFORMACE PŘEDCHOZÍHO KÓDU DO PŘÍKAZU, KTERÝ VYTVOŘÍ TABULKY DLE DEFINIC

    ;SET @SQLcommand_schema = 'CREATE SCHEMA ' + @SchemaName + ';'

    ;SET @SQLcommand_tables =
            'CREATE TABLE ' + @SchemaName + '.' + @FactTableName + '(' +
               IIF(@Col0Name = @null, '',       @Col0Name) + IIF(@Col0Name = @null, '', ' ' + @Col0Type) + IIF(@Col
               IIF(@Col1Name = @null, '', ',' + @Col1Name) + IIF(@Col1Name = @null, '', ' ' + @Col1Type) + IIF(@Col
               IIF(@Col2Name = @null, '', ',' + @Col2Name) + IIF(@Col2Name = @null, '', ' ' + @Col2Type) + IIF(@Col
               IIF(@Col3Name = @null, '', ',' + @Col3Name) + IIF(@Col3Name = @null, '', ' ' + @Col3Type) + IIF(@Col
               IIF(@Col4Name = @null, '', ',' + @Col4Name) + IIF(@Col4Name = @null, '', ' ' + @Col4Type) + IIF(@Col
               IIF(@Col5Name = @null, '', ',' + @Col5Name) + IIF(@Col5Name = @null, '', ' ' + @Col5Type) + IIF(@Col
               IIF(@Col6Name = @null, '', ',' + @Col6Name) + IIF(@Col6Name = @null, '', ' ' + @Col6Type) + IIF(@Col
               IIF(@Col7Name = @null, '', ',' + @Col7Name) + IIF(@Col7Name = @null, '', ' ' + @Col7Type) + IIF(@Col
               IIF(@Col8Name = @null, '', ',' + @Col8Name) + IIF(@Col8Name = @null, '', ' ' + @Col8Type) + IIF(@Col
               IIF(@Col9Name = @null, '', ',' + @Col9Name) + IIF(@Col9Name = @null, '', ' ' + @Col9Type) + IIF(@Col
               CHAR(13) + ');' + CHAR(13) + CHAR(13) +

            'CREATE TABLE ' + @SchemaName + '.' + @DimTable0Name + CHAR(13) + '(' +
               IIF(@D0Col0Name = @null, '',       @D0Col0Name) + IIF(@D0Col0Name = @null, '', ' ' + @D0Col0Type) +
               IIF(@D0Col1Name = @null, '', ',' + @D0Col1Name) + IIF(@D0Col1Name = @null, '', ' ' + @D0Col1Type) +
               IIF(@D0Col2Name = @null, '', ',' + @D0Col2Name) + IIF(@D0Col2Name = @null, '', ' ' + @D0Col2Type) +
               IIF(@D0Col3Name = @null, '', ',' + @D0Col3Name) + IIF(@D0Col3Name = @null, '', ' ' + @D0Col3Type) +
               IIF(@D0Col4Name = @null, '', ',' + @D0Col4Name) + IIF(@D0Col4Name = @null, '', ' ' + @D0Col4Type) +
               IIF(@D0Col5Name = @null, '', ',' + @D0Col5Name) + IIF(@D0Col5Name = @null, '', ' ' + @D0Col5Type) +
               IIF(@D0Col6Name = @null, '', ',' + @D0Col6Name) + IIF(@D0Col6Name = @null, '', ' ' + @D0Col6Type) +
               IIF(@D0Col7Name = @null, '', ',' + @D0Col7Name) + IIF(@D0Col7Name = @null, '', ' ' + @D0Col7Type) +
               IIF(@D0Col8Name = @null, '', ',' + @D0Col8Name) + IIF(@D0Col8Name = @null, '', ' ' + @D0Col8Type) +
               IIF(@D0Col9Name = @null, '', ',' + @D0Col9Name) + IIF(@D0Col9Name = @null, '', ' ' + @D0Col9Type) +
               CHAR(13) + ');' + CHAR(13) + CHAR(13) +
```

Obrázek 9 – Složený příkaz pro definitivní tvorbu tabulek v databázi (autor: vlastní zpracování)

Jednotlivé příkazy pro tabulky jsou poskládány ve tvaru „CREATE TABLE“ s následným označením umístění a názvu samotné tabulky. Dále pak dochází k využití nadefinovaných názvů sloupců a jejich datových typů.

Jelikož je možné, že některé dostupné sloupce nemají existovat (dosadili jsme za něj proměnnou *@null* = „“), musíme počítat se situací, že by se v textu vyskytovaly nadbytečné čárky oddělující definice jednotlivých sloupců, které neexistují. Tyto čárky by způsobovaly chyby syntaxe a nefunkčnost skriptu. Aby k tomu nedocházelo a skript fungoval tak, jak má, je nutné využít funkci IIF, (která v SQL slouží shodně jako běžné IF funkce) která bude vždy kontrolovat, zda se název následujícího sloupce rovná prázdné hodnotě, či nikoliv. V případě že ano, daná proměnná bude přeskočena i se všemi svými vlastnostmi. Díky tomuto je zabezpečena funkčnost sekvence kódu bez ohledu na počet vytvářených sloupců v jakékoliv tabulce.

```
498                'CREATE TABLE ' + @SchemaName + '.' + @DimTable3Name + CHAR(13)
499                   IIF(@D3Col0Name = @null, '',       @D3Col0Name) + IIF(@D3Col0
500                   IIF(@D3Col1Name = @null, '', ',' + @D3Col1Name) + IIF(@D3Col1
501                   IIF(@D3Col2Name = @null, '', ',' + @D3Col2Name) + IIF(@D3Col2
502                   IIF(@D3Col3Name = @null, '', ',' + @D3Col3Name) + IIF(@D3Col3
503                   IIF(@D3Col4Name = @null, '', ',' + @D3Col4Name) + IIF(@D3Col4
504                   IIF(@D3Col5Name = @null, '', ',' + @D3Col5Name) + IIF(@D3Col5
505                   IIF(@D3Col6Name = @null, '', ',' + @D3Col6Name) + IIF(@D3Col6
506                   IIF(@D3Col7Name = @null, '', ',' + @D3Col7Name) + IIF(@D3Col7
507                   IIF(@D3Col8Name = @null, '', ',' + @D3Col8Name) + IIF(@D3Col8
508                   IIF(@D3Col9Name = @null, '', ',' + @D3Col9Name) + IIF(@D3Col9
509                   CHAR(13) + ')'
510
511    -- ČÁST ČTVRTÁ
512        -- spuštění obou částí kódu
513
514            ;EXEC sp_executesql @SQLcommand_schema
515            ;EXEC sp_executesql @SQLcommand_tables
516
517        --  ;print @SQLcommand_schema
518        --  ;print @SQLcommand_tables
519
520
```

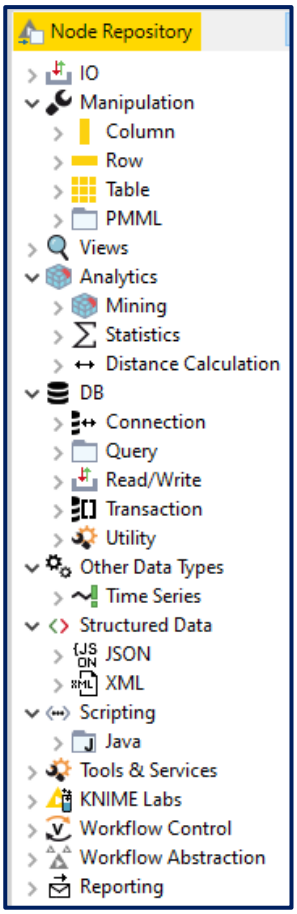
Obrázek 10 – Spuštění příkazů pro vytvoření schématu a tabulek (autor: vlastní zpracování)

V závěru skriptu dojde ke spuštění dvou řádků kódu, které využívají jednak proměnné obsahující všechen kód poskládaný do funkčního pořadí, zároveň ale také k využití SQL uložené procedury s názvem *sp_executesql*. Její klíčová vlastnost pro náš skript je schopnost spouštět variabilní kód uložený v rámci proměnných tak, jako kdyby se jednalo o obyčejný skript. Pro zachování funkčnosti jsou oba „spouštěče“ odděleny středníky (Ghanayem et al., 2023).

V rámci skriptu je možné celkový vytvářený skript taktéž zobrazit pomocí funkce PRINT – z tohoto důvodu je do třetí části kódu, kde se všechny proměnné formují do správné syntaxe, osazena mnohými funkcemi „CHAR(13), které slouží pro rozdělení textu a jeho pokračování na dalším řádku. Pro přehlednost výsledného skriptu je to klíčová vlastnost umožňující revizi uživatelem.

4.3.2 KNIME workflow

Práce v KNIME spočívá ve tvorbě řetězce různých operátorů, nazvaných „nody“, které načítají, transformují a ukládají data. Tento řetězec nazýváme *workflow*, tedy jakýmsi tokem operací, ve kterém se jednotlivé úkony a operace zpracovávají v postupném pořadí v závislosti na úspěšném dokončení předcházející činnosti/činností. KNIME neslouží jen k operacím typu ETL, v nabídce je široké množství nejrůznějších funkcí, včetně analytických a statistických nástrojů.



Obrázek 11 – Strom kategorií nabízených operací v KNIME (autor: vlastní zpracování)

Námi vytvářené KNIME workflow prochází několika stěžejními „etapami“, jejichž výčet je uveden níže. K jednotlivým etapám je nutné se dále vyjádřit, a to zejména z pohledu potenciální náročnosti jednotlivých etap na jejich tvorbu, na jejich údržbu a na jejich individuálně přizpůsobený koncept pro fungování u konkrétních zákazníků, jejichž data budeme zpracovávat se snahou dosazení do co nejbližší varianty obecného datového modelu. Jedná se o fáze ETL:

1) E – Extrakce dat ze zdrojových úložišť
2) T – Transformace a úprava dat do tvaru výstupních tabulek
3) L – Loading dat do jednotného SQL schématu (= předpřipravený datový model)

4.3.2.1 Načtení zdrojových dat ze zákazníkových systémů

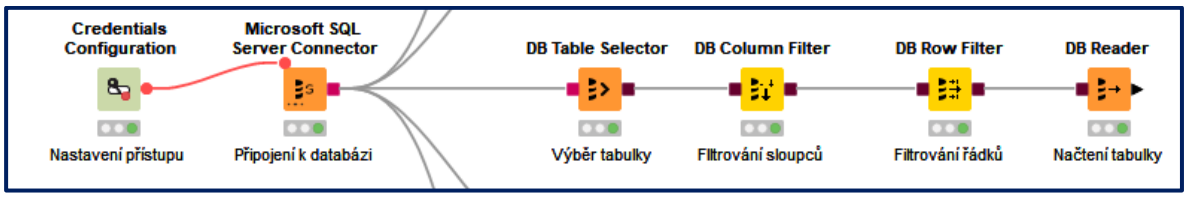
Nejběžnější datové zdroje, jejichž využití můžeme předpokládat, jsme již stanovili v kapitole *3.1.* Předně lze předpokládat, že zákazník má data uložená v SQL databázi, jelikož se jedná o běžně využívaný postup v rámci většiny současných zákazníků. KNIME rovněž disponuje možností připojení k dalším běžně dostupným úložištím byznysových dokumentů, např. MS Excel, MS Sharepoint, OneDrive apod.

Připojení přímo k SAP databázi je možné, avšak za určitých dalších vstupních předpokladů, jejichž vyhodnocení by příslušelo byznysovým požadavkům a podmínkám jednotlivých projektů – připojení k SAP je dodatečným rozšířením KNIME, jehož instalace a používání může podléhat různým bezpečnostním nařízením organizace (KNIME, [b.d.]).

Pokud budou zákaznická data skutečně pocházet ze SAP, počítáme již od návrhu modelu s obohacením stávající architektury o mezistupeň tak, aby KNIME nebylo nutné napojovat přímo na SAP, nýbrž na vložené meziúložiště SQL databáze, které by sloužilo jako staging vrstva pro načtení zdrojových dat. Odtud bychom je načítali do KNIME a dále s nimi pracovali. Takto mimo jiné funguje i původní řešení.

Vstupem do této fáze jsou pro účely našeho workflow zdrojová datová úložiště, výstupem tabulky s daty.

Obecně lze předpokládat připojení k SQL databázím, případně načtení excelových sešitů. Pro tyto typy připojení budeme mít připraveny konektory, které umožní čtení těchto dat. Ke čtení dat přímo z databází SQL bude s nejvyšší pravděpodobností rovněž třeba ověřit přístupová práva z důvodu zabezpečení těchto databází. Tento krok je v KNIME běžně dostupný.



Obrázek 12 – Příklad workflow v KNIME pro načtení dat z SQL, jejich filtrování, prvotní úpravy (autor: vlastní zpracování)

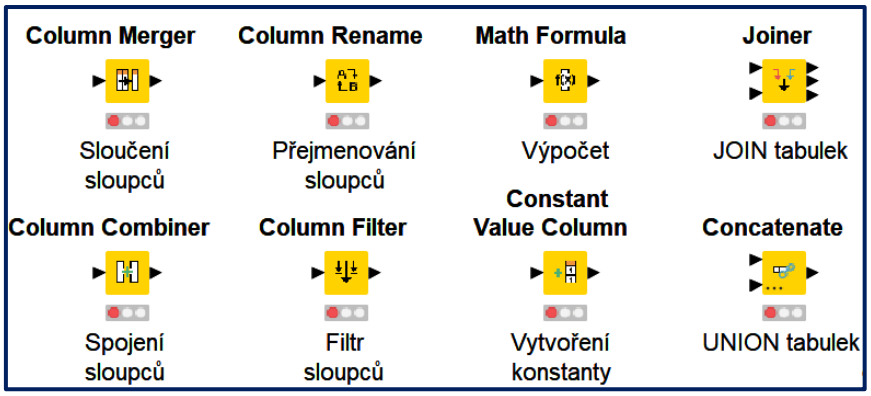
Součástí této úvodní etapy práce v KNIME je rovnou i nastavení datových typů u jednotlivých sloupců načtených tabulek. Dalším krokem, který je možné provést již v této fázi procesu, je veškeré datové zdroje různě filtrovat. Pokud vstupem do této fáze jsou všechny zdrojové tabulky dat, výstupem jsou jen data v tabulkách, která budeme dále potřebovat.

Pokud budeme ze zdrojových tabulek čerpat každý den původní a nová data, je nejvhodnější odfiltrovat dřívější, již zpracované záznamy pryč a pracovat každý den pouze s novými – ty následně připojovat do již existujících tabulek. Tohoto lze docílit různými způsoby filtrování zdrojových dat, například dle jejich časového příznaku připsání do databáze, či dle identifikátoru času záznamu. Konkrétní způsob závisí na každé implementaci zvlášť.

4.3.2.2 Transformace dat

Tato fáze bude z hlediska náročnosti na zhotovení KNIME workflow jednak nejsložitější, zároveň ale bude u každého budoucího zákazníka vyžadovat prakticky individuální přístup, kterou můžeme usnadnit jen zevrubnou úvahou o možných operacích. Jedná se o kompilovanou etapu z hlediska nutnosti porozumění vstupním datům, požadovanému tvaru jejich výstupu a všem potenciálním operacím, které propojí tyto sva stavy. Rovněž je klíčová zručnost v programu KNIME.

V rámci přípravy KNIME workflow je vhodné identifikovat možné potenciální operace, které k práci s daty můžeme využít. K tomuto bude sloužit know-how získané v rámci tvorby dokumentace – během transformace dat původního řešení ve vytvořených SQL pohledech docházelo především k filtrování použitých sloupců, dotváření konstantních hodnot v dotvářených sloupcích, sumace a průměrování hodnot kapacit a samozřejmě byly použity UNION a JOIN operace mezi jednotlivými tabulkami. Všechny tyto operace a operace jim podobné můžeme v KMINE workflow připravit do schématu pro pozdější rychlejší vyhledání a použití.



Obrázek 13 – Příklad KNIME nástrojů pro datovou transformaci (autor: vlastní zpracování)

Při slučování tabulek obdobně jako při SQL funkci UNION je nutné využít CONCATENATE node. Při tomto kroku je nutné držet na paměti že jen sloupce se shodným názvem se sloučí – pokud nemají

sloupce v obou tabulkách zastoupení se stejným názvem, sloupec se sice k výsledné tabulce připojí, do řádků patřících původní tabulce bez tohoto sloupce KNIME neumístí nic, budou zde tedy null záznamy. Předpokládejme rovněž i nutnost tvorby JOIN vazeb mezi jednotlivými tabulkami, či tvorbu dodatečných kalkulovaných sloupců s výpočty kapacit. Pro tyto kroky rovněž předpokládejme nutnost případných změn datových typů jednotlivých sloupců, pokud nejsou ve správném formátu pro tyto operace připraveny již z předcházejících kroků.

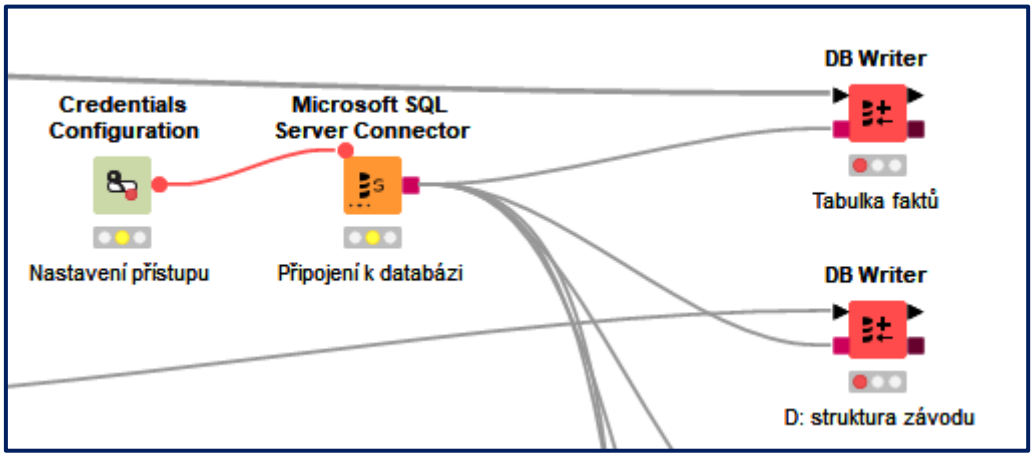
Koncem procesu v této transformační etapě by měly být tabulky odpovídající tabulkám popsaným v předcházející části *4.2*. Jelikož jsme navrhli datový model v jednotné struktuře s cílem využít ho v Power BI s minimálním potřebným počtem změn mezi jednotlivými zákazníky (tedy s cílem poskytovat vizuál vždy na základě co nejobdobnějšího datového modelu), bude třeba data předpřipravit do jednotné struktury.

4.3.2.3 Nahrání dat do předpřipravené SQL databáze

Posledním krokem, který budeme v KNIME provádět, je nahrání tabulek do SQL databáze – tentokrát do námi připraveného úložiště, které bude sloužit pro uložení dat z KNIME a zároveň pro načtení dat do Power BI. Každou z předpřipravených tabulek budeme chtít nahrát do databáze, kde tou dobou musí existovat předpřipravená tabulka shodných vlastností – počtu sloupců, jejich datových typů.

Vstupem i výstupem pro tuto fázi workflow jsou tabulky odpovídající svou strukturou těm tabulkám, které pro daného zákazníka budeme dále nahrávat do SQL za účelem zobrazení v Power BI.

Přestože KNIME umí komunikovat napřímo s Power BI, a to za pomoci rozšíření dostupného zdarma v aplikaci KNIME (Knime docs, [b.d.]), není z pohledu údržby žádoucí přímo posílat data mezi těmito dvěma softwary. Zaprvé chceme výsledné tabulky v SQL ukládat v připravené podobě, zadruhé chceme mít možnost data ad hoc analyzovat.



Obrázek 14 – Příklad schéma pro nahrání dat do dvou tabulek SQL databáze (autor: vlastní zpracování)

4.4 Power BI

Následující podkapitoly se budou zabývat teoretickými úvahami o možnostech vizualizace a jejich vhodného použití s cílem co nejvhodnějšího přetvoření doposud popsaných fází BI řešení do závěrečných vizualizací. Praktická ukázka vizualizace není předmětem této podkapitoly – vizualizace a jejich konstrukce budou představeny v rámci praktické aplikace nástroje.

4.4.1 Obecné předpoklady pro tvorbu vizualizace

Před samotnou tvorbou vizuálů, grafů a tabulek je třeba připomenout si některé body, jejichž zanedbání sníží kvalitu námi dodaného řešení.

4.4.1.1 Interaktivita dashboardu

Dashboard by měl být schopen nabídnout co možná nejširší možnost komplexně nahlížet na data z různých úhlů a při různých kombinacích filtrů a granularity jednotlivých ukazatelů. Jelikož předpokládáme určitou míru interaktivity dashboardu na základě různých úrovní granularity dat, hierarchických číselníků a filtrů dle nejrůznějších měřítek, je třeba vhodně sestavit dashboard tak, abychom se nepřipravili o možnost některou z kombinací filtrů a úrovní dat zobrazit. Pokud bychom měli sebelepší datový model nahraný do Power BI a okleštili bych potenciál jeho informační hodnoty špatně zvolenými vizualizacemi či filtry, dopustili bychom se zbytečné chyby vedoucí ke snížení míry sdělení informací.

4.4.1.2 Srozumitelnost dashboardu

V kontrastu s předcházejícím bodem, výsledný produkt nesmí být pro zákazníka nesrozumitelný. Přespřílišná komplexita vizualizací by mohla působit nejen odpudivě z grafických hledisek, zejména ale právě z hlediska srozumitelnosti dat. Složitost produktu by neměla odpovídat zručnosti tvůrce, nýbrž být adekvátní schopnostem zákazníka.

Celkový vizuál by měl působit neutrálně a srozumitelně tak, aby skupina budoucích uživatelů byla schopna s řešením v základní podobě pracovat. Opět je třeba mít na paměti, že výsledný produkt je bezpodmínečně nutné později přizpůsobit na míru zákazníkovi. Lze počítat s tím, že čím profesně výše je zákazník postavený ve výrobním závodě, tím méně obecnější úroveň detailu bude chtít na první pohled monitorovat. Vedoucí pracovníci a představitelé managementu budou zpravidla potřebovat rychle a na první pohled získat informace, zda všechno ve výrobním závodě funguje vše zhruba tak jak má, a nebudou se zabývat detailními informacemi jednotlivých pracovišť. Mezi informace, které by takto měly být sdělovány, bude patřit například zda je kapacita ve všech základně dělených částech výrobního závodu dostatečná, kde ne, jak se nedostatečnost/přebytek vyvíjí v čase.

Detailní informace musí být samozřejmě k dispozici také, lze ale předpokládat, že se jimi budou vedoucí pracovníci zabývat ne vždy, případně až později po shlédnutí prioritních základních ukazatelů, případně bude detailní informace sledovat někdo zcela jiný, kdo naopak bude primárně zodpovědný za sledování detailnějších hodnot. Je tedy třeba ideálně dashboardy rozdělit vždy tak, aby bylo předem zřejmé, zda se jedná o základní, či detailní přehled.

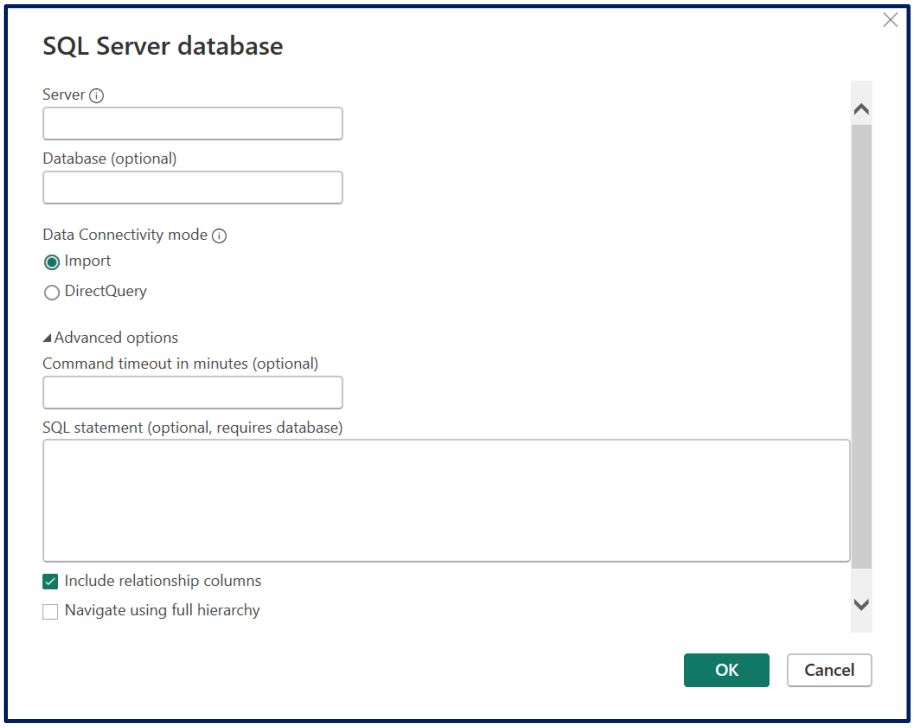
Na základě dosavadních zkušeností s reportingem výrobních závodů lze předpokládat, že základní dashboardy by měly obsahovat primárně grafické zobrazení, které bude doplněno maximálně určitými KPIs, detailní zobrazení by mělo kombinovat grafické a tabulární zobrazení dat tak, aby v detailních úrovních závodu měli uživatelé reportu možnost dohledat přesně každé číslo.

4.4.1.3 Servis dashboardu

Dodatečné modifikace a údržba řešení v chodu je pro potenciální zákazníky samozřejmostí. Ať již se bude jednat o drobnosti jako přizpůsobení barev vizuálů, přechodové prvky mezi jednotlivými dashboardy, či přímo úprava/tvorba sledovaných ukazatelů, každý zákazník by měl mít jistotu, že jemu určené řešení bude zhotovené na míru jeho potřebám, bude optimálně nastavené a bude obsahovat aktuální data. Tento dokument neposkytuje univerzální návod, jak každému zákazníkovi doručit stejný dashboard, nýbrž návod a pomocné soubory pro snížení náročnosti tvoření individuálních řešení obdobné tématiky.

4.4.2 Nahrání dat

Data do Power BI budeme v obecném případě vždy čerpat z SQL databáze, do které jsme v předcházejícím kroku celého řešení nahráli transformovaná data z KNIME.



Obrázek 15 – Menu pro nahrání dat z SQL databáze do Power BI (autor: vlastní zpracování)

Power BI umožňuje dva typy připojení k databázi. Prvním z nich je „Import“, který slouží k jednorázovému, okamžitému připojení a načtení dat. Pozdější aktualizace probíhá opět jako jednorázový, okamžitý úkon, který lze automatizovat za použití složitějších mechanismů v rámci Power BI serveru. Tento typ připojení je vhodný pro naše potřeby, kdy budeme chtít dávkově v určité periodicitě aktualizovat data. Po nahrání dat jsou informace uloženy v paměti Power BI a práce s nimi je v tomto ohledu poměrně rychlá a efektivní.

Druhý typ připojení, DirectQuery, pracuje na principu živého připojení ke zdroji dat, které v našem případě nemá smysl využít. Jeho využití by bylo na místě, pokud bychom byli napojeni například přímo na zákazníkovy systémy, které produkují data prakticky neustále, a my bychom chtěli v okamžitém čase tento vývoj sledovat. Námi předpokládaná aktualizace dat po 24 hodinách bude dostatečně pokryta prvním popsaným způsobem načtení
dat – dávkovými importy a aktualizacemi.

4.4.3 Konstrukce datového modelu

Datový model sice v obecné rovině zkonstruovaný máme, vždy bychom před nahráním dat z SQL měli mít představu o jeho vlastnostech a klíčích, zároveň ale až teprve v Power BI, a to po nahrání všech tabulek, budeme datový model konstruovat.

Několik obecných předpokladů pro jeho konstrukci:

4.4.3.1 Typ vazby mezi tabulkami

Na základě klíčů budeme chtít jednotlivé tabulky propojit, a to vždy ideálně vazbou 1:N. Použití vazby M:N je výrazně komplikovanější, zejména z hlediska správnosti a interpretace. Ani vazba M:N, ani vazba 1:1 v představovaném relačním modelu nenacházejí žádné uplatnění – nebudou tedy použity.

4.4.3.2 Směr filtrování tabulek

V rámci datového modelu lze nastavit směr filtrování mezi tabulkami tak, aby při omezení jedné tabulky filtry došlo zároveň k automatickému omezení hodnot i v druhé tabulce, která je s ní propojena. Běžně pro naše potřeby modelu bude stačit jednosměrné filtrování mezi tabulkami, a to tak, abychom pomocí filtrování dimenzionální tabulky omezovali hodnoty v tabulkách faktů. Obousměrné filtrování dat mezi tabulkami se bude zhotovovaných modelů týkat jen v případě složitějších a obsáhlejších datových modelů.

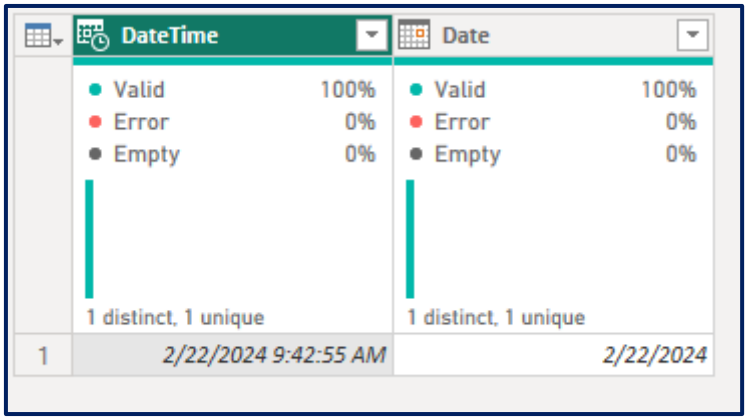
Power BI automaticky zneaktivní vazby mezi tabulkami v případě, kdy se dopustíme potenciálního cyklického vztahu mezi tabulkami na základě těchto směrů filtrování tabulek.

4.4.3.3 Tabulky nepocházející z dat

Každý zhotovovaný report v Power BI bude vhodné obohatit o tabulky, které nepocházejí přímo z dat, ale dělají dashboardy srozumitelnějšími.

První z takových tabulek bude kalendářní tabulka, která bude sloužit jako dimenze času. Ačkoliv se dá předpokládat existence atributu času ve faktových tabulkách, tato časová řada nemusí být vhodná pro řízení času v reportu. Vygenerováním automatického kalendáře pomocí DAX funkce CALENDARAUTO() zabezpečíme existenci souvislé časové řady v rámci celého vytvořeného datového modelu, kterou můžeme použít pro filtrování datového modelu na základě data, či času.

Další tabulkou, kterou by naše reporty měly běžně obsahovat, bude tabulka stahující aktuální čas v systému, z něhož provádíme aktualizaci reportu. Tento údaj budeme využívat pro zobrazení poslední aktualizace reportu tak, aby zákazník měl představu o aktuálnosti dat. Aktuální datum a čas získáme v rámci PowerQuery, a to s pomocí funkce DateTime.FixedLocalNow(). Dodatečně lze tento údaj poupravit na pouhé datum, tedy odstraněním času. Ve výsledných reportech bychom chtěli mít možnost zobrazovat oboje, proto bude tato tabulka o velikosti 2 sloupců a 1 řádku vždy načítána. Tuto tabulku není třeba připojovat k datovému modelu.



Obrázek 16 – Tabulka zobrazující poslední aktualizaci dat v Power BI (autor: vlastní zpracování)

Výpis 4.1 – Kód v PowerQuery vytvářející tabulku sledující systémový čas (vlastní zpracování)

```
let
    Source = DateTime.FixedLocalNow(),
    #"Converted to Table" = #table(1, {{Source}}),
    #"Renamed Columns" = Table.RenameColumns(#"Converted to Table",{{"Column1", "Date-
Time"}}),
    #"Changed Type" = Table.TransformColumnTypes(#"Renamed Columns",{{"DateTime", type date-
time}}),
    #"Duplicated Column" = Table.DuplicateColumn(#"Changed Type", "DateTime", "DateTime -
Copy"),
```

```
    #"Extracted Date" = Table.TransformColumns(#"Duplicated Column",{{"DateTime - Copy", Da-
teTime.Date, type date}}),
    #"Renamed Columns1" = Table.RenameColumns(#"Extracted Date",{{"DateTime - Copy",
"Date"}})
in
    #"Renamed Columns1"
```

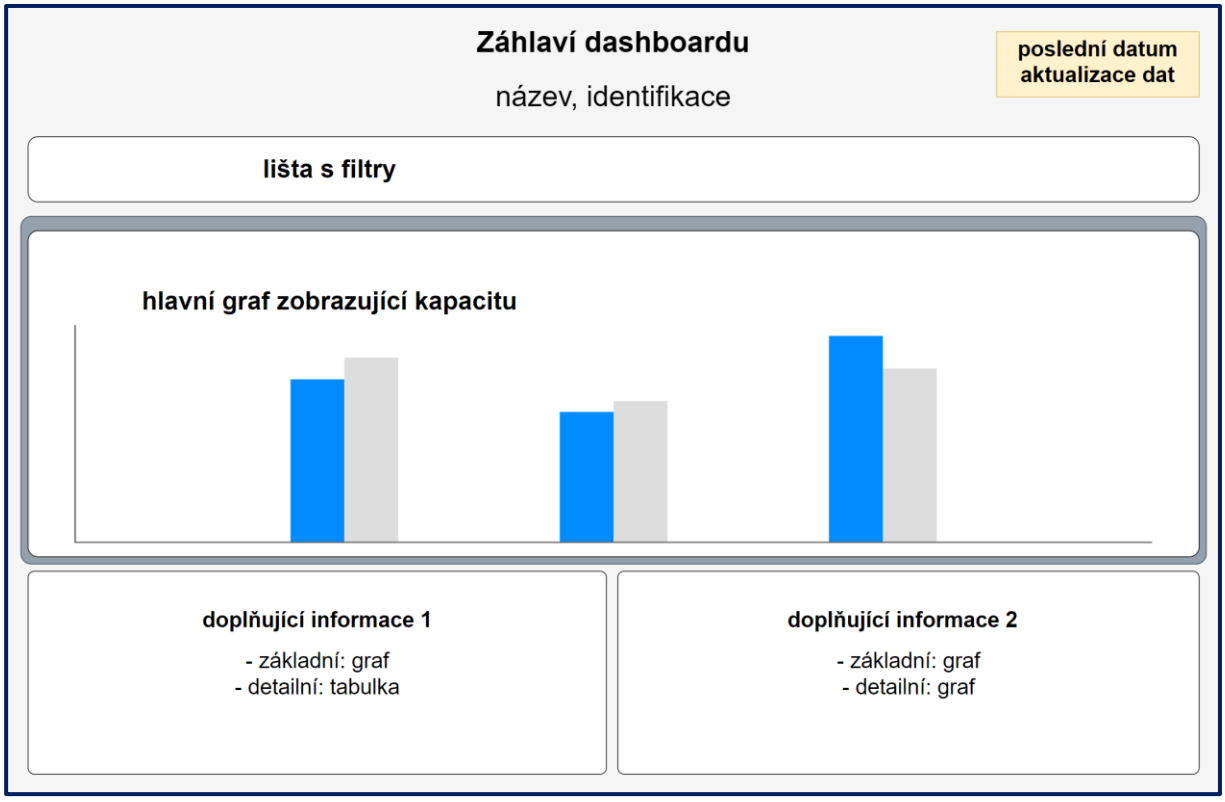
Vzhled a počet dashboardů

Obecné řešení nabízíme jako produkt, proto je vhodné automaticky přicházet s návrhem dashboardů tak, abychom s každým potenciálním zákazníkem nemuseli vytvářet nové návrhy dashboardů – v prvním návrhu řešení budeme chtít zákazníkům poskytnout alespoň částečnou představu o reportingu, který zamýšlíme.

Předpokladem je existence dat, které budou umožňovat vznik těchto dashboardů.

4.4.3.4 Obecné vlastnosti dashboardů

Samozřejmostí každého dashboardu je jeho název v záhlaví a lišta s filtry, ve které budeme chtít ovlivňovat data na základě dimenzionálních tabulek – předpokládáme primárně filtr času a dimenzionální struktury, dále vlastností dle konkrétních dat.



Obrázek 17 – Základní design dashboardu s rozvržením prostoru (autor: vlastní zpracování)

V souladu s obecnými postupy tvorby dashboardů a reportingu by měl být dashboard orientovaný vodorovně, a to ideálně tak, aby přesně kopíroval plochu obrazovek uživatelů. Při jeho otevření je rovněž důležité, aby uživatel okamžitě upřel svou pozornost na klíčový prvek dashboardu, který přenáší základní a nejdůležitější informace – v tomto případě samotný vývoj kapacit v čase a srovnání nabídky s poptávkou. Pro zajištění okamžitého upoutání pozornosti je hlavní graf největším prvkem dashboardu a je zvýrazněn rámečkem.

Pro hlavní prvek dashboardu, celkový graf kapacit, lze předpokládat použití sloupcového grafu, ve kterém bude srovnávána nabídka s poptávkou v jednotlivých časových instancích na ose X. Použití sloupcového grafu je ideální, neboť nabídka i poptávka jsou kvantitativní hodnoty, jejichž srovnání probíhá již na první pohled pomocí výšky sloupců. Tento typ grafu je jednoduchý, uživatelé ho instinktivně chápou, mohou se tedy zaměřit rovnou na jeho obsah (Potančok, Pour, Chramostová 2020, s. 114).

V případě různých typů kapacit, které se podílejí na celku, lze rovněž použít sloupcový skládaný graf, pomocí něhož rozšiřujeme předávanou informaci na nejen výšku celkového sloupce v porovnání s jinými sloupci, ale také přidáváme velikost a podíl složek podílejících se na celkové výšce sloupce.

Pokud uživatel odhlédne od hlavního grafu, a zaměří svou pozornost do vedlejších vizualizací ve spodní části dashboardu, nalezne dva prostory s doplňujícími informacemi, jejichž typologie se bude odvíjet od požadované míry sdělovaného detailu. Lze předpokládat, že pro základní dashboard se může jednat o různou kombinaci grafů, naopak detailní pohledy na data by měly obsahovat tabulku, ve které budou informace členěny v textové podobě tak, aby byl uživatel schopen dohledat konkrétní údaj.

4.4.3.5 Základní přehled kapacit

Tento dashboard bude sloužit primárně k získání představy o fungování závodu na první pohled, bez zbytečných detailů. Primárním zdrojem informací je vizuální stránka věci, kapacita je zobrazována pomocí sloupcových grafů. Cílovou skupinou uživatelů je primárně management / výše postavení zaměstnanci monitorující výrobní kapacitu.

Snahou je předat informace o celkovém stavu nabídky a poptávky a jejich výši, popřípadě upozornit na místa s největšími rozdíly v nabídce a poptávce.

4.4.3.6 Detailní přehled kapacit

V tomto dashboardu bude cílem zobrazit opět základní sloupcový graf, ve spodních detailech ho ale obohatit o značně detailnější informace, ze kterých bude těžit personál bližší operativní úrovni, který musí sledovat dění i na nižších úrovních struktur závodu, potenciálně až do nejnižších jednotek struktury.

Cílem je zobrazení detailních hodnot nabídky i poptávky, jejich rozdílů a stavu dostatečnosti kapacity ve všech dostupných úrovních závodu tak, aby uživatel mohl všemi úrovněmi procházet a zkoumat komplexně kapacitu ze všech úhlů pohledu. K dispozici jsou mu jak grafické ukazatele, tak také tabulka obsahující agregované hodnoty dle míry požadované agregace a filtrace dat.

4.5 Aktualizace dat

V budoucnu bude samozřejmě nutné zajistit, aby u jednotlivých implementací docházelo k aktualizaci dat v celém řešení. Popisované řešení není jednorázovým procesem, je třeba aby zákazník mohl skrze popisovanou architekturu řešení posílat nová data, která se mu budou automaticky objevovat v dashboardech na konci celého řešení. Při každé budoucí aplikaci popisovaného postupu je třeba zvážit, na čí pokyn budou data aktualizována a s jakou periodicitou.

Dalším bodem týkajícím se aktualizací dat je samotný způsob aktualizace. Bez ohledu na časový interval mezi aktualizacemi budeme vždy preferovat aktualizaci dat dávkově – za přenos dat je v popisované architektuře primárně zodpovědný software KNIME, ve kterém máme nastavený proces kompletního stažení dat v momentě, ve kterém je vydán řídícím uživatelem pokyn.

Danou aktualizaci dat budeme provádět nejprve v rámci KNIME workflow, poté budeme chtít aktualizovat dashboardy v Power BI. Pro automatizaci aktualizace dat se využije KNIME Server, který umožňuje automatické spouštění workflow na základě nastaveného času, jako tzv. „scheduled job“ (Knime docs, [b.d.]b).

KNIME Server a jeho služby jsou zpoplatněny (na rozdíl od samotného KNIME), a přístup k nim je udělen na základě licence, kterou disponují jen někteří zaměstnanci BI oddělení. Power BI vyžaduje pro nastavení aktualizací rovněž zpoplatněnou verzi licence, kterou drží jen někteří zaměstnanci. Předpokládaný postup v případě budoucích implementací je ten, že zpracované řešení se všemi podklady bude předáno kolegům v rámci oddělení, kteří jsou držiteli těchto licencí, jsou v komunikaci s danými zákazníky, a tedy mají přístup k těmto metodám automatizace procesů a mohou je nastavit na základě dohody se samotnými zákazníky.

5. Zdroje

GHANAYEM, Mark, et al., 23.05.2023. *sp_executesql (Transact-SQL).* Online. Learn.microsoft.com. Dostupné z: https://learn.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/sp-executesql-transact-sql?view=sql-server-ver16. [citováno 27.02.2024].

INTELLIPAAT, 27.04.2024. *Power BI vs Tableau: Which Data Visualization Tool is Better.* Online. IntelliPaat. Dostupné z: https://intellipaat.com/blog/power-bi-vs-tableau-difference/. [citováno 26.02.2024].

ISEMINGER, David, et al., 10.11.2023. *Configure scheduled refresh.* Online. Learn.microsoft.com. Dostupné z: https://learn.microsoft.com/en-us/power-bi/connect-data/refresh-scheduled-refresh#scheduled-refresh. [citováno 27.02.2024].

KNIME, [b.d.]. *Easy Access to your SAP Data.* Online. Knime. Dostupné z: https://www.knime.com/easy-access-to-your-sap-data. [citováno 29.02.2024].

KNIME DOCS, [b.d.]. *KNIME Extensions 5.2 – KNIME Power BI Integration User Guide.* Online. Knime Docs. Dostupné z: https://docs.knime.com/latest/powerbi_integration_user_guide/index.html. [citováno 29.02.2024].

KNIME DOCS, [b.d.]. *KNIME Server 4.16 – KNIME Server User Guide.* Online. Knime Docs. Dostupné z: https://docs.knime.com/latest/server_user_guide/index.html. [citováno 29.02.2024].

LARSEN, Greg, 2022. When to use CHAR, VARCHAR, or VARCHAR(MAX). Online. Red Gate. Dostupné z: https://www.red-gate.com/simple-talk/databases/sql-server/learn/when-use-char-varchar-varcharmax/. [citováno 23.02.2024].

MICROSOFT, 13.10.2023. *Views.* Online. Learn.microsoft.com. Dostupné z: https://learn.microsoft.com/en-us/sql/relational-databases/views/views?view=sql-server-ver16. [citováno 23.02.2024].

POTANČOK, Martin, Jan POUR a Veronika CHRAMOSTOVÁ, 2020. *Business analytika v Praxi.* 1. vyd. Praha: Vysoká škola ekonomická v Praze, Nakladatelství Oeconomica. ISBN 978-80-245-2382-8.

POUR, Jan, Miloš MARYŠKA, Iva STANOVSKÁ a Zuzana ŠEDIVÁ, 2018. *Self service business intelligence: jak si vytvořit vlastní analytické, plánovací a reportingové aplikace*. 1. vyd. Praha: Grada Publishing. ISBN 978-80-271-0616-5.

POWER BI, 2024. *Ceny Power BI.* Online. Power BI. Dostupné z: https://www.microsoft.com/cs-cz/power-platform/products/power-bi/. [citováno 26.02.2024].

RICHARDSON, Ben, 14.05.2020. *Differences between the M Language and DAX in Power BI.* Online. SQL Shack. Dostupné z: https://www.sqlshack.com/differences-between-the-m-language-and-dax-in-power-bi/. [citováno 24.02.2024].

SAP, 2024. *Enterprise Resource Planning.* Online. Sap.com. Dostupné z: https://www.sap.com/products/erp.html. [citováno 04.03.2024].

SIEMENS, 2024. *Global Business Services (GBS).* Online. Siemens.com. Dostupné z: https://www.siemens.com/global/en/products/services/gbs.html. [citováno: 04.03.2024].

SLÁNSKÝ, David, 2018. *Data a analytika pro 21. století: Architektura a governance.* 1. vyd. Průhonice: Professional Publishing. ISBN 978-80-88260-22-6.

SPARKMAN, Maggie, et al., 14.11.2022. *Create small multiples in Power BI.* Online. Learn.Microsoft.com. Dostupné z: https://learn.microsoft.com/en-us/power-bi/visuals/power-bi-visualization-small-multiples. [citováno: 03.03.2024] .

TABLEAU, 2024. *What is Tableau?* Online. Tableau. Dostupné z: https://www.tableau.com/why-tableau/what-is-tableau. [citováno 04.03.2024].

TABLEAU, 2024. *Pricing.* Online. Tableau. Dostupné z: https://www.tableau.com/pricing/teams-orgs. [citováno 24.02.2024].

TAYLOR, David, 30.12.2023. *Snowflake Schema in Data Warehouse Model.* Online. Guru99. Dostupné z: https://www.guru99.com/snowflake-schema-in-data-warehouse-model.html. [citováno 27.02.2024].

RYCHETSKÝ, L.: Reportingové BI řešení pro plánování výrobních kapacit v závodech. DP, VŠE, 2024.