VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

APLIKACE PRO ZOBRAZOVÁNÍ OLAP DATOVÝCH KOSTEK

APPLICATION FOR DISPLAYING OLAP DATA CUBES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ KISS**
AUTHOR

VEDOUCÍ PRÁCE **Ing. BARTÍK VLADIMÍR, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Kiss Tomáš**
Program: Informační technologie
Název: **Aplikace pro zobrazování OLAP datových kostek**
Application for Displaying OLAP Data Cubes
Kategorie: Databáze

Zadání:

1. Seznamte se s problematikou datových skladů, datových kostek a OLAP analýzy.
2. Analyzujte požadavky na aplikaci, která bude pracovat s databází PostrgeSQL, bude provádět jednoduchou etapu ETL a zobrazovat datové kostky ve vhodné formě s podporou základních OLAP operací.
3. Navrhněte aplikaci dle požadavků z bodu 2, využijte k tomu vhodné modelovací techniky. Návrh konzultujte s vedoucím.
4. Navrženou aplikaci implementujte a otestujte její funkčnost na vhodném vzorku dat.
5. Zhodnoťte dosažené výsledky a diskutujte další možné pokračování tohoto projektu.

Literatura:

- LABERGE, Robert. Datové sklady: agilní metody a business intelligence. Brno: Computer Press, 2012. Expert (Grada). ISBN 978-80-251-3729-1.
- LINOFF, Gordon. Data analysis using SQL and Excel. Second edition. Indianapolis, IN: John Wiley, 2016, 978-04-700-9951-3.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 14. května 2020
Datum schválení: 16. října 2019

Abstrakt

Cieľom tejto bakalárskej práce je poskytnúť informácie o základnej problematike dátových skladov, dátových kociek a OLAP analýzy. V teoretickej časti práce sa predstavuje etapa predspracovania dát a jeho elementárne časti prevedené pred ich samotným vkladaním do dátových skladov. Okrem toho sú stručne uvedené základné typy OLAP operácii a možnosti zobrazenia dátových kociek. Získané poznatky sú využité k vytvoreniu hlavného výsledku práce, ktorá má podobu aplikácie pre zobrazovanie OLAP dátových kociek vo vhodne zvolenej podobe. Vytvorené riešenie poskytuje možnosť prevedenia základných operácii OLAP analýzy nad výslednými dátovými kockami.

Abstract

The main aim of this thesis is to provide information about the basic issues of data warehouses, data cubes and OLAP analysis. Furthermore, in the theoretical part is introduced the stage of data preprocessing and its elementary parts performed before their insertion into data warehouses. Additionally, there are listed the basic types of OLAP operations and the possible ways of displaying data cubes. Acquired information is used to create application for displaying OLAP data cubes in suitable format.

Kľúčové slová

OLAP, ETL, dátový sklad, dátové kocky, multidimenzionálny model, PostgreSQL

Keywords

OLAP, ETL, data warehouse, data cube, multidimensional model, PostgreSQL

Citácia

KISS, Tomáš. *Aplikace pro zobrazování OLAP datových kostek*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Bartík Vladimír, Ph.D.

Aplikace pro zobrazování OLAP datových kostek

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Vladimíra Bartíka, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Tomáš Kiss
24. mája 2020

Poďakovanie

Chcel by som sa touto cestou poďakovať vedúcemu práce, Ing. Vladimírovi Bartíkovi, Ph.D., za vedenie mojej práce a stálu podporu a ochotu poradiť v dôležitých otázkach.

Obsah

Kapitola 1

Úvod

V dnešnej dobe vzniká každodenne obrovské množstvo dát uložených v elektronickej podobe. Firmy ukladajú rôzne údaje napríklad o svojich zákazníkoch, predanom tovare a o ďalších pre nich dôležitých entitách a udalostiach. Preto vzniká otázka, prečo je dobré uložiť údaje o niečom takom, čo sa udialo v minulosti? Cieľom tejto činnosti je neskôr tieto dáta analyzovať z rôznych hľadísk a získať z nich čo možno najrelevantnejšie informácie, ktoré sú použiteľné pre manažérov a vedenie firmy, na uľahčenie ich rozhodovania v strategických otázkach.

Cieľom tejto práce je predstaviť základnú problematiku dátových skladov, dátových kociek a OLAP analýzy. Poskytnúť vedomosti potrebné k pochopeniu súvislostí medzi týmito pojmami. Demonštrovať základné možnosti vytvárania a zobrazovania dátových kociek a vysvetlenie operácii OLAP analýzy, prevoditeľných nad nimi. Cieľom operácii je umožniť podrobné preskúmanie dát.

Dátové sklady a OLAP analýza tvoria spolu zásadný krok v získavaní informácií z databáz. Tie potom môžu byť použité v prospech organizácii v rôznej podobe od detegovania podvodu, cez akcie cielené na udržanie zákazníkov, až riadenia výrobných procedúr. Preto na trhu nájdeme veľa už existujúcich komerčných nástrojov a firmy obetujú nemalé finančné prostriedky a čas na vybudovanie vlastných celopodnikových skladov dát.

Tento dokument je rozčlenený na 5 hlavných častí. V kapitole 2 je popísaný základný princíp dátového skladu, proces jeho vytvárania, možné oblasti jeho využitia a odlišnosti v porovnaní s operačnými databázami.

V kapitole 3 je predstavená samotná OLAP analýza. Popísané sú jej rozličnosti oproti OLTP a oblasť jej použitia. Pre prácu s dátami v samotných kockách sú uvedené niektoré z operácii analýzy, ako je napríklad Roll-up, Drill-down.

Kapitola 4 sa zameriava na samotný návrh aplikácie, ktorá z pohľadu teoretickej stránky vychádza z informácii uvedených v predošlých častiach práce. Analýza vlastností aplikácie bola prevedená na základe požiadavky na samotnú aplikáciu a z hľadiska oblasti činnosti, zahrňujúcej prácu s veľkým množstvom dát.

Rozoberanie detailov implementácie sa nachádza v kapitole 5 s predstavením použitých technológii, ako je databázový systém PostgreSQL.

Testovanie vytvorenej aplikácie bolo prevedené na ukážkových dátových skladoch naplnených s fiktívnymi údajmi. Proces testovania je popísaný v kapitole 6.

Záverečné zhrnutie celej práce a prípadné možnosti na rozšírenie aplikácie v budúcnosti, sú popísané v kapitole 7.

Kapitola 2

Dátové sklady

Pre zjednodušenie získavania znalostí z databáz sa dáta uchovávajú v dátových skladoch, oproti bežným produkčným databázam. Táto časť práce je venovaná stručnému popisu teórie potrebnej k porozumeniu fungovania dátových skladov.

Sekcie 2.1 a 2.4 sa zaoberajú s vlastnosťami dátových skladov a krokmi, ktoré predchádzajú možnosti ich aktívneho používania.

Bežne používané typy databáz síce majú ako hlavný cieľ uchovávanie dát a predsa majú svoje znaky, v ktorých sú odlišné od seba. Tieto odlišnosti sú uvedené v sekcii 2.2.

V prípade nasadenia dátových skladov vo veľkých inštitúciách sú dáta často integrované z rôznych menších transakčných databáz. To prináša so sebou možnosť výskytu problémov, ako je nekonzistentnosť dát alebo chýbajúce údaje. Tieto problémy by mohli viesť až k nesprávnym rozhodnutiam, založených na získaných vedomostiach. Kroky vedúce k ich odstráneniu sú uvedené v sekcii 2.4.

2.1 Definícia dátového skladu

Dátové sklady (anglicky *data warehouse*) poskytujú architektúru pre ukladanie dôležitých dát a sú nad nimi vybudované nástroje pre manažérov a vedenie firiem pre uľahčenie rozhodovania v závažných otázkach, týkajúcich sa dosiahnutia strategických cieľov podniku. Môžu priniesť počas dlhých období zníženie nákladov, so sledovaním trendov, vzorov a výnimiek, napríklad v chovaní zákazníkov a to dôsledným a spoľahlivým spôsobom.

V predošlých rokoch pre dátové sklady boli vytvorené rôzne definície. V [2] sú predstavené, ako zbierka historických dát všeobecne používaná s OLAP technológiami. Integrujú dáta z od seba nezávislých heterogénnych dátových zdrojov. Vytvárajú jednotný pohľad nad celou organizáciou.

Podľa najznámejšej definície Williama H. Inmona, kto je považovaný za otca dátových skladov, sú to stále, integrované a časovo variantné kolekcie dát, ktoré sa orientujú na subjekty.

Na základe [7] a [13] sú štyri kľúčové vlastnosti, odlišujúce ich od ostatných dátových úložísk. Môžeme si ich vysvetliť nasledovne:

- **Orientovaný na subjekty:** znamená to, že sú organizované okolo hlavných subjektov, ako sú zákazníci, produkty, predaj a dodávatelia tovaru. Ich zameranie smeruje na modelovanie a analýzu dát oproti každodenným operáciám a spracovania transakcii organizácie. Poskytujú jednoduchý a stručný pohľad na subjekt dosiahnutý odstraňovaním dát, ktoré nie sú užitočné pre podporu rozhodovania.

- **Integrovaný:** Sú vytvorené integrovaním dát z niekoľko heterogénnych zdrojov, napr. databáz podnikov organizácie, ktoré sú lokalizované na rôznych miestach. Z toho dôvodu, že informácie pochádzajú z viacerých miest, je potrebné previesť opatrenia týkajúce sa zjednotenia konvencie pomenovania atribútov, merných jednotiek a terminológii. Pre tieto účely sa v etape naplňovania skladov je prevedené čistenie a integrácia dát (sekcia 2.4). Dáta sú na základe určitých pravidiel spojované tak, aby poskytli koncovému užívateľovi celopodnikový pohľad na oblasť jeho záujmu.

- **Časovo variantný:** Dáta sa ukladajú cieľom poskytovať informácie z historickej perspektívy, napr. o poslednom desaťročí. Preto je často pre nich používaný výraz historické dáta. Čas predstavuje kľúčovú veličinu v dátovom sklade a objavuje sa explicitne alebo implicitne v každej štruktúre dát. Ukladanie je uskutočnené v časových snímkach, ktoré vytvárajú históriu, časovo premennú radu snímkou.

- **Nemenný:** Dátový sklad je vždy fyzicky oddelený od dát z operačných databáz. Vďaka tomu že je oddelený nevyžaduje transakčné zapracovanie, zotavenie, riadenie paralelného zapracovania požiadaviek. Zvyčajne vyžaduje iba dve operácie okrem počiatočného nahrania dát do skladu: pridávanie nových dát a prístup k dátam.

Podľa Inmona podniky majú iba jeden dátový sklad, z ktorého dátové trhy získavajú potrebné informácie pre seba. Budovanie začína vytvorením centrálneho podnikového skladu dát, až potom sa vytvárajú dátové trhy pre analytické potreby jednotlivých oddelení firmy. Dáta sa ukladajú do relačných databáz v tretej normálnej forme (3NF). Inmon používa ER model pre centrálny sklad a dimenzionálny model iba pre trhy. Analytické systémy môžu pristupovať k údajom v hlavnom sklade iba cez trhy dát.

Dátový sklad podľa Ralpha Kimballa je budovaný podľa prístupu zdola nahor. Oproti Inmona vytvára sa spojením všetkých dátových trhov podniku do jedného centrálneho skladu. Informácie sa vždy ukladajú v dimenzionálnom modeli, ktorý môže mať podobu schémy hviezdy alebo snehovej vločky. Analytické systémy a nástroje pre podávanie správ môžu priamo pristupovať k údajom.

Obe architektúry majú jednu základnú spoločnú vlastnosť, obsahujú jedno integrované úložisko atomických údajov.

Pre porovnanie dvoch architektúr bol použitý zdroj [27].

Dátový trh

Dátový trh je podmnožina dátového skladu zvyčajne prínosná iba pre konkrétne určité oddelenie podniku [26]. Je zameraný na jeden vybraný subjekt. Dáta, ktoré sú získané buď z podnikových skladov alebo operačných databáz, sú v sumarizovanej podobe. Ich návrh a vývoj trvá oproti skladov iba niekoľko týždňov, prípadne mesiacov. Môžu existovať samostatne ako menší a flexibilnejší sklad dát alebo ako súčasť väčšieho systému.

2.2 Porovnanie relačnej databázy a dátového skladu

Relačná databáza je založená na relačnom modeli, v ktorom dáta sú štruktúrované do tabuliek, medzi sebou vzájomne prepojené [25]. Tabuľky majú riadky a stĺpce s jedinečným názvom, vyjadrujúce ich vlastnosti.

Klasické operačné databázy a dátové sklady sú relačné databázy, aj keď majú odlišné ciele a spôsob využitia. Na nasledujúce predstavenie odlišností bol použitý zdroj [9].

Klasické databázy

Operačné databázy majú stanovený ako primárnu úlohu on-line transakcie a spracovanie prichádzajúcich dotazov podľa [26]. Veľký dôraz sa kladenie na ich efektívnosť a rýchle spracovanie veľkého množstva prichádzajúcich dotazov.

Používajú sa na spravovanie dynamických dát v reálnom čase. Tieto systémy sa označujú ako OLTP (anglicky *On-Line Transaction Processing*) systémy podľa ich hlavnej úlohy. Uchovávajú aktuálne, modifikovateľné a aplikačne špecifické dáta. Obsahujú detailné informácie, potrebné ku každodenným obchodným operáciám – inventúra, predaj, mzda – a podporujú operácie nad ich obsahom, ako je vkladanie, aktualizácia, mazanie a dopytovanie. Dáta sa často menia vďaka aktualizáciám informácii o najnovších transakciách. Z hľadiska dátových skladov predstavujú zdroj nových informácii.

Dátové sklady

Symbolizujú databázy, navrhnuté pre zjednodušenie dopytovania, nad veľkým množstvom dát a proces ich samotnej analýzy. Na rozdiel od operačných databáz obsahujú mohutný objem dát z rôznych zdrojov (sekcia 2.1), ktoré sú určené iba pre čítanie. Transformujú ich do tvaru multidimenzionálneho dátového modelu (sekcia 3.2). Umožňujú flexibilným spôsobom ich vizualizáciu podľa požiadaviek v rôznych tvaroch. Tieto úložiská sú optimalizované na čítanie, prevedenie komplexnejších dotazov. Schopnosť prevedenia zložitých dotazov spočíva v tom, že tabuľky sú vytvorené ako jednoduché, obsahujú iba najpodstatnejšie atribúty, a nenormalizované. V porovnaní s databázami pre OLTP sú centralizované, je potrebné previesť spojenie menej počtu tabuliek pri dopytovaní.

Uspokojujú funkcionálne a výkonové požiadavky OLAP analýzy, ktoré sú výrazne odlišné od požiadaviek OLTP aplikácii – podporované operačnými databázami. Pre tento typ analýzy sú nezanedbateľné historické agregované dáta, ktorých dátové sklady obsahujú od stovky giga- až terabytov. Zhrnutie rozdielov (vychádza z [9]):

Operačná databáza	**Dátový sklad**
Navrhnuté pre podporu zapracovania veľkého množstva transakcii	Navrhnuté pre podporu veľkého množstva analytických procesov
Obsahujú aktuálne dáta	Obsahujú historické dáta
Pravidelne aktualizované podľa potrieb	Stály, dáta sa menia zriedkavo. Dáta môžu byť pridávané pravidelne.
Optimalizované pre jednoduchý súbor transakcií. Obecne pridávanie alebo vrátenie jedného riadku na tabuľky.	Optimalizované pre rozsiahly súbor komplexných, nepredvídateľných dotazov pristupujúci k mnohým riadkom jednotlivých tabuliek.
Optimalizovaný na validáciu informácii počas transakcii	Naplnený konštantnou, validnou informáciou nevyžadajúci overovanie v reálnom čase
Podpora pre tisícky konkurentných klientov	Podpora pre malý počet konkurentných klientov
Orientované na procesy	Orientované na subjekty
Optimalizované na prevedenie rýchle vkladanie a aktualizáciu malého objemu dát	Optimalizované pre vrátenie veľkého objemu dát

Tabuľka 2.1: Porovnanie vlastností operačnej databázy a dátového skladu.

Oba tieto systémy pracujú s podobnými informáciami ale napriek tomu majú odlišné využitie a charakteristiky. Detailnejší popis niekoľkých z najbežnejších rozdielov (zdroj [7] [6]) uvedený v tabuľke 2.1 :

- Užívatelia: OLAP aplikácie majú oproti OLTP menší počet stálych zákazníkov. Napríklad OLTP je používané v bankovníctve miliónmi zákazníkov denne, ale ich zvyky v investovaní analyzuje iba pár expertov.

- Orientácia systému: Z hľadiska orientácie sa OLAP približuje k obchodu, preto ich typických užívateľov predstavujú manažéri. Z druhej strany OLTP sa zameriava na zákazníka preto je často používaný úradníkmi.

- Dáta: OLTP systémy zvyčajne obsahujú aktuálne dáta s detailným popisom. Práve kvôli tejto detailnosti nie sú vhodné pre podporu rozhodovania, ich analýza by trvala príliš dlho. OLAP spravuje veľké množstvo historických dát, ktoré boli zaobstarávané počas dlhých rokov. Veľmi dôležitým atribútom u nich je čas.

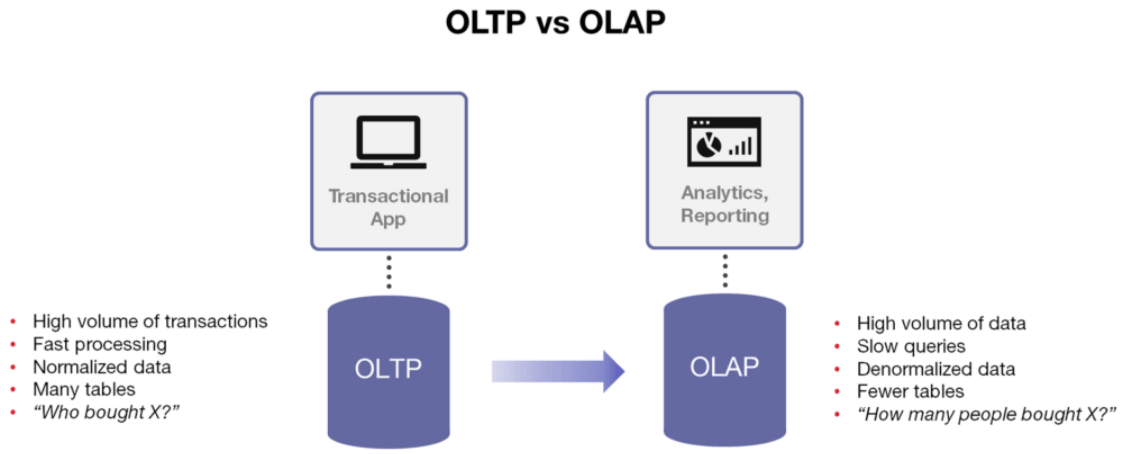
- Množstvo dát: OLTP sa zameriava na aktuálne dáta, ktorých zdroj môže byť buď celý podnik alebo jedno konkrétne oddelenie. Týchto dát je okolo stovky megabajt až gigabajt. Na rozdiel od toho OLAP spravuje stovky gigabajt až terabajt, lebo pracuje s informáciami nie len z jednej organizácie. Integruje obsah viacerých dátových úložísk. Kvôli ich veľkému objemu ich ukladanie je rozdelené na viacero pamäťových médií.

- Návrh databázy: Často sa používa entity-relationship (ER) dátový model u OLTP kombinovaný s aplikačne orientovaným návrhom databázy. Oproti tomu OLAP je subjektovo orientovaný a využíva schému hviezdy alebo snehovej vločky.

- Prístup k dátam: Vzhľadom na to, že OLAP pracuje s historickými dátami, preto pristupuje k ním operáciami pre čítanie a nemení ich. Čítanie je prevedené často so zložitými SQL (anglicky *Structured Query Language*) dotazmi. Prístup s krátkymi, atomickými operáciami je typické pre OLTP. Kvôli veľkému počtu paralelných dotazov je potrebné zaistiť mechanizmy pre prípadnú potrebu zotavenia.

- Metriky: U OLTP systémov prioritne sa zameriava na vysoký výkon, dostupnosť a priepustnosť transakcií. U druhého systému zase na flexibilitu, odozvu pri dopytovaní a počet obslúžených dotazov za jednotku času – priepustnosť.

Podľa uvedených rozdielov vo vlastnostiach týchto dvoch systémov je možné usúdiť, že je vhodnejšie pre analýzu použiť oddelený dátový sklad. Síce operačné databázy obsahujú mohutný objem údajov ale ich štruktúra nevyhovuje, chýba im historický charakter. Musia byť čistené a agregované pred možným použitím. Použitie OLAP operácii v operačných databázach by degradovalo ich výkon značným spôsobom.



Obr. 2.1: Kľúčové rozdiely medzi OLAP a OLTP systémy. Prevzaté z [6].

2.3 Možnosti návrhu dátového skladu

Rastom výkonu a poklesom nákladov na hardvér a softvér sa umožnilo vyhovieť náročným požiadavkám na budovanie a využívanie dátových skladov. Bolo potrebné najmä zabezpečiť spracovanie veľkého množstva dát, ich ukladanie a analýzu. Rozšírenie webovej technológie tiež sprístupnilo informácie širokému publiku adresátov.

Pri budovaní dátového skladu je odporúčané sústrediť sa na kvalitu zdrojových dát. Nie všetky dáta v prevádzkových databázach sú správne. Časté chyby, ako záporná hodnota počtu kusov, dátum zo začiatku storočia, nevyplnené PSČ a ďalšie, môžu z akéhokoľvek analyzovania urobiť presné zobrazenie nepresných čísel.

Zlé navrhnutý dátový sklad alebo nepresné hodnoty môžu vystaviť manažérov a vedenie k nebezpečenstvu prevedenia zlých strategických rozhodnutí a záverov.

Vybudovanie a používanie dátového skladu je komplexná úloha vyžadujúca niekoľko schopností [7]. Schopnosť pochopiť fungovanie týchto systémov, spravovanie dát. Je potrebné vymyslieť software pre extrakciu údajov a následné naplnenie z databázy a obnovovací, ktorý udržiava sklad dostatočne aktuálny. Pochopenie obchodných potrieb a ich pretavenie do dotazov uspokojiteľných dostupnými údajmi, patrí medzi základné činnosti.

Analytici dát musia byť schopný odhaliť vzory a trendy, odhadnúť chovanie zákazníkov na základe informácii o minulosti, hľadať zmeny paradigiem.

Pri návrhu dátového skladu je potrebné uvažovať o štyroch možných pohľadoch [7]:

Pohľad zhora dole

Je potrebné vybrať relevantné informácie, ktoré zodpovedajú súčasným a budúcim obchodným potrebám a môžu byť použité k budovaniu dátových skladov.

Pohľad na zdroj dát

Odhaľuje informácie, ktorý operačný systém zachytáva, ukladá a spravuje. Dáta môžu byť zdokumentované na rôznych úrovniach presnosti a podrobnosti. Zdroje dát sa modelujú pomocou tradičných modelovacích techník ako je relačný (ER) model.

Pohľad na dátový sklad

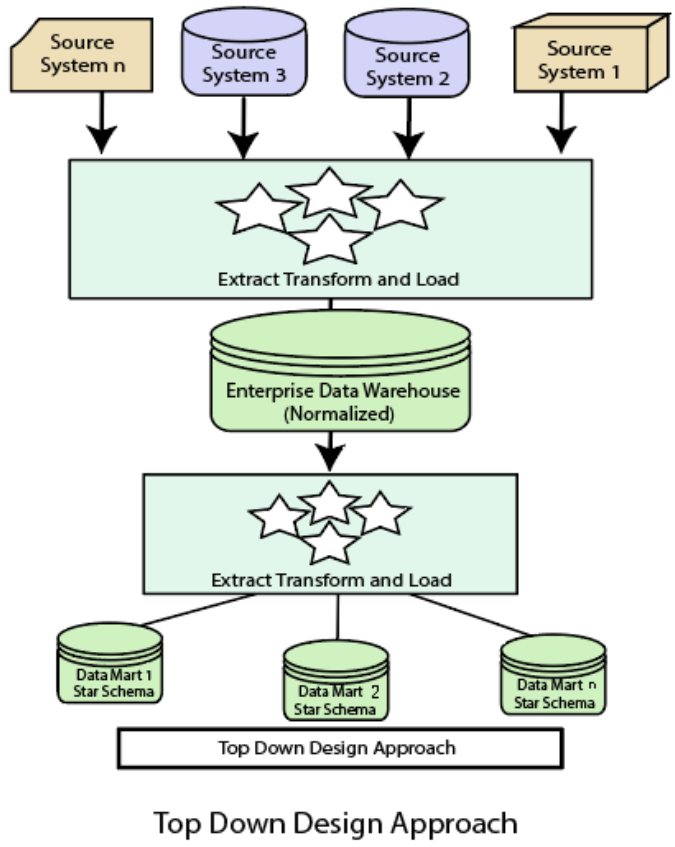
Reprezentuje údaje uložené v skladoch. Medzi tieto údaje patria aj vopred sumarizované počty. Ďalej pre zabezpečenie historického kontextu informácie o dátumu a času ohľadom zaznamenania.

Pohľad na obchodný dotaz

Údaje z perspektívy koncového užívateľa v dátovom sklade.

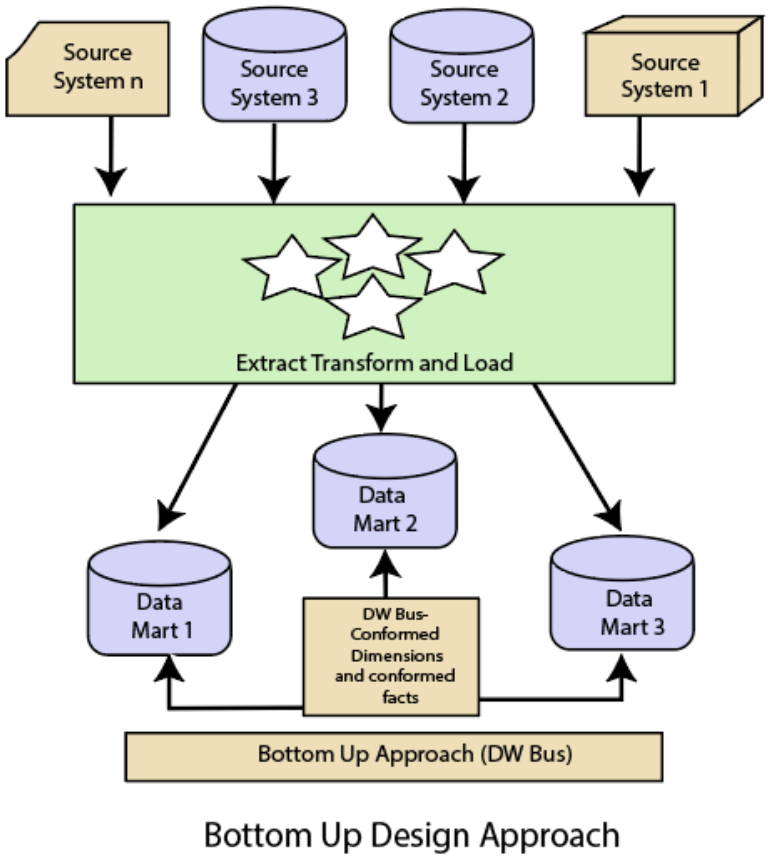
Dátový sklad môže byť navrhnutý podľa troch typov metód [7]: zhora dole, dole nahor a kombinácia predošlých dvoch.

Pri metóde zhora dole sa začína s celkovým návrhom a plánovaním systému. Pre použiteľnosť tejto metódy je potrebné, aby technológie, ktoré budú nasadené, boli známe a obchodné problémy boli zreteľné a dobre pochopiteľné. Je to prístup zameraný na dáta. Na základe toho, že sa najprv zhromažďujú informácie a až potom sa stanovia obchodné požiadavky na základe subjektov, na ktoré sa dátové trhy zamerajú. Výhodou metódy je aj podpora jedného centrálneho skladu, vďaka ktorého sa dátové sklady môžu prekrývať bezproblémovo. Nevýhodou je, že technika nie je flexibilná, v prípade neustále meniacich sa požiadaviek zo strany oddelení. Zdroj [10].



Obr. 2.2: Proces vytvárania dátového skladu s použitím metódy zhora dole. Prevzaté z [10].

Metóda zdola hore začína s experimentovaním a vytváraním prototypov. Tento princíp je veľmi užitočný v ranných/skorých fázach obchodného modelovania. Umožňuje organizáciám napredovať menšou záťažou na náklady a vyhodnotiť technologické výhody pred prevedením závažných zmien. Výhodou tejto metódy je, že má rýchlu návratnosť investícií. Návrh a vybudovanie jedného dátového trhu, v úlohe dátového skladu pre konkrétny subjekt, je rýchlejší a prináša menší risk zlyhania. Integrovanie nového trhu s už existujúcimi je jednoduchý. Umožňuje rýchle generovanie správ, dokumentov. Zdroj [10].



Obr. 2.3: Proces vytvárania dátového skladu s použitím metódy zdola hore. Prevzaté z [10].

Kroky návrhu a vybudovania dátového skladu z pohľadu softwarového inžinierstva zahrňujú: plánovanie, analýzu požiadaviek a problému, návrh, integráciu dát, testovanie [26]. Tieto kroky sú buď prevedené iba raz alebo niekoľkokrát inkrementálne ako aj u metódy zdola hore.

Všeobecne proces návrhu tvoria nasledujúce kroky [7]:

1. Výber obchodného procesu pre modelovanie a analýzu. Môže to byť napríklad predaj, objednávky, zásielky. Dátový sklad sa zvolí pre proces, ktorý zahrňuje kolekciu objektov a týka sa celej organizácie. Ak sa proces týka iba jedného konkrétneho oddelenia zvolí sa ako model dátový trh.

2. Výber atomickej jednotky dát, ktorá bude reprezentovať proces v tabuľke faktov. Ako príklad môže slúžiť transakcia, denný záznam.

3. Výber dimenzií, ktoré sa budú vzťahovať k záznamom v tabuľke faktov (sekcia 3.2). Čas, produkt, zákazník môžu byť typickým predstaviteľom dimenzie.

4. Výber merných jednotiek tabuľky faktov. Najčastejšie sú to cena produktu, počet predaných kusov.

Budovanie dátového skladu je dlhodobý proces, preto je potrebné stanoviť ciele, ktoré majú byť dosiahnuté a podľa ktorých sa dá skontrolovať už dosiahnuté štádium procesov. Ciele musia byť dostatočne špecifické, dosiahnuteľné a merateľné.

Pri návrhu je potrebné ďalej stanoviť rozpočet a čas vyhradený pre vytvorenie skladu. Počet a typ oddelení, ktoré bude obsluhovať a samotné zdroje údajov pre naplňovanie.

2.4 ETL

ETL je skratkou pre Extract/Transform/Load (po slovensky *extrakcia/transformácia/uloženie*) označovaný aj ako *dátová pumpa*. Podľa [20], práve funkcie zapadajúce pod tento proces sú tie, ktoré rozlišujú dáta v operačných databázach od informácií v dátových skladoch. Zapracovanie požiadaviek by nemohlo fungovať bez korektného vyťaženia, čistenia a transformácie dát. Predstavuje neoddeliteľnú časť získavania znalostí z databáz. ETL systém zabezpečuje nasledujúce činnosti [12]:

- Opravenie chýb a odstránenie chýbajúcich údajov.
- Poskytuje vierohodnosť dát.
- Usporiada dáta pre možnosť spoločného použitia z rôznorodých zdrojov.
- Prevedie štruktúrovanie dát.

Tieto činnosti sú časovo náročné, dáta je potrebné čím rýchlejšie poskytnúť pre analýzu, preto je tento systém jednou z kritických častí vytvárania a udržiavania dátového skladu. Problémy systému často vychádzajú z faktu, že zdroje dát medzi sebou majú odlišnosti v reprezentácií rovnakých údajov, bežia na iných platformách. Vývojári musia vyložením veľkého úsilia nájsť vhodné riešenie pre tieto rozličnosti. Často sa nepoužijú všetky stĺpce v pôvodnej podobe z extrahovaných tabuliek, ale sú nad nimi prevedené buď agregácie alebo sa rozdelia na menšie časti ak obsahovali zloženú informáciu.

Extrakcia dát v ETL

V tomto kroku sa často zameriava na efektívne, rýchle získavanie dát, s čo najmenším ovplyvnením a zdržaním zdrojových databáz. Dáta sa dočasne uložia do odkladacej oblasti, ktorá umožňuje aj ich validáciu, čistenie a prípadné medzi výpočty.

Je potrebné vytvoriť stratégiu pre bezproblémové prevedenie tohto kroku. Ako prvé sú vyznačené zdroje poskytujúce potrebné údaje pre každý jeden typ informácie, ktorý má sklad obsahovať. Zdroje majú ďalej stanovenú metódu, frekvenciu – čas po ktorom sa proces zopakuje – extrakcie.

Transformácia dát v ETL

Pred nahrávaním nových historických dát do dátového skladu je potrebné nad nimi previesť niekoľko typov transformácií. Cieľom týchto modifikácií je, aby boli jednotné a zapadali do špeciálnej schémy skladu.

Často sa nepoužijú všetky stĺpce v pôvodnej podobe z extrahovaných tabuliek, ale sú nad nimi prevedené buď agregácie alebo sa rozdelia na menšie časti ak obsahovali zloženú informáciu.

Hlavným prínosom tejto časti je zvýšenie kvality dát, nad ktorými dolovanie bude prevedené.

Nahrávanie dát v ETL

Ako posledný krok sa nahrajú nové kvalitné dáta do skladov. Je potrebný stanoviť vhodný čas nahrávania nových informácií do dátového skladu, aby jej používanie bolo čo najmenej narušené. Vhodným časom môžu byť neskoré večerné hodiny alebo skoré ranné pred začatím pracovnej doby manažérov a analytikov. Je vhodné mať pripravené obnovovacie mechanizmy pre prípad zlyhania procesu, aby sa nestratil pôvodný obsah.

Nahrávanie je často rozdelené na počiatočné, inkrementálne a znova načítanie. Rozdiel medzi nimi je v dobe ich prevedenia a spôsobu ukladania dát. Počiatočné načítanie je prevedené úplne na začiatku vytvárania skladu s kompletným naplnením.

Inkrementálne plnenie a znovu načítanie sú prevedené už priebežne počas fungovania skladu. Rozdiel medzi nimi je v reflektovaní zmien. Pri prvom sú nahrané iba zmeny oproti aktuálnemu obsahu, v prípade druhého spôsobu sa načítavajú celé dimenzie.

Proces vytvárania dátového skladu

Pri popise procesu budovania dátového skladu sa vychádza z článku [5] a knižky [7].

Z všeobecného hľadiska pri budovaní dátových skladov zahrňuje nasledujúce základné kroky:

1. Umiestnenie získaných transakčných dát zo zdrojov do odkladacej oblasti pre predspracovanie.
2. Transformácia zdrojových dát do vhodnej podoby na základe určitých kritérií.
3. Nahranie výsledkov transformácie do skladu dát.
4. Vytváranie agregácií dát s cieľom zrýchliť tvorbu správ.
5. Implementácia vlastného nástroja pre podávanie správ alebo zaobstaranie od tretej strany.

Vyťaženie transakčných údajov

Predstavuje proces získania dát z rôznych zdrojov dát a ich následné uloženie do odkladacej oblasti. Prísť na to, že z akým spôsobom je najlepšie extrahovať dáta z odlišných zdrojov, robí z tohto kroku najťažší krok. Ak je tento proces prevedený dostatočnou dôslednosťou výsledný dataset bude obsahovať významne menej chýb v podobe nekonzistencie, nadbytočnosť odvoditeľných informácií.

Predspracovanie zdrojových dát

Pre zabezpečenie kvality dát v dátovom sklade a pre dolovacie algoritmy[1] je treba dáta predspracovať [7]. Kvalita dát môže byť posúdená z viacerých aspektov – presnosť, úplnosť,

[1] Algoritmy pre získanie znalostí z dát uchovaných v dátových skladoch.

konzistencia. Táto činnosť je potrebná kvôli faktu, že databázy reálne používané ako zdroje často obsahujú údaje v nevhodnej podobe, v zlom stave. Označujú sa ako nečisté dáta. Dôvodom ich nečistoty je absencia hodnôt, nekonzistencia alebo že sú zašumené[2]. Všetky tieto faktory znižujú kvalitu dát, čo môže viesť k tomu, že záver vytvorený na základe výsledkov dolovania bude nesprávny, zavádzajúci.

Hlavným cieľom predspracovania dát je odstránenie ich nedostatkov. Pre dosiahnutie tohto cieľu sú zodpovedné nasledujúce úlohy [7]:

Čistenie a konzistencia dát

Je to časovo náročný, iteratívny proces. Po odstránení chýb sa môžu objaviť nové. Pozostáva z dvoch krokov:

1. **Detekcia nezrovnalostí v dátach:** Zdrojom nezrovnalostí sú často zle navrhnuté vstupy pre dáta a ľudská chyba pri ich vyplňovaní, integrácia z viacerých zdrojov. K detekcií tejto chyby vo veľkom pomáhajú metadáta. Obsahujú informácie o vlastnostiach dát – dátový typ, možné hodnoty, interval hodnôt, formát[3].

2. **Odstránenie nezrovnalostí :** Malý počet detegovaných chýb je opraviteľné aj ručne, ale častejšie vyžiadajú prevedenie transformácie. V tomto vedia byť nápomocné nástroje ako je ETL (anglicky *Extraction/Transformation/Loading*).

Transformácia dát

Zahrňuje techniky ako je vyhladzovanie šumu, vytváranie nových atribútov z už existujúcich, agregácia podrobnejších údajov do väčších celkov. Normalizácia je tiež často potrebnou technikou, pri ktorej sú numerické hodnoty upravené tak, aby zapadali do určitého intervalu. Dolovanie dát sa vďaka transformáciám prevedených na dátach stáva efektívnejším.

Dáta pochádzajú z rôznorodých zdrojov – prostý databázový súbor, operačná databáza – je treba zaviesť do súvislostí, vytvoriť medzi nimi vzťah. Keď je odkladací priestor hotový a naplnený dátami, je potrebné previesť dôležité operácie. Tieto operácie prevedú nevyhnutné modifikácie dát, aby sa dalo vytvoriť puto medzi tabuľkami a ich stĺpcami prichádzajúcimi z rozličných zdrojov.

Vytvorenie dimenzionálneho modelu

Relačná databáza (sekcia 2.2) je normalizovaná – sú eliminované duplicitné údaje – a fungujú dobre v spojení s OLTP. Ale pri vytváraní správ z historických dát nie sú najlepšou voľbou. Bolo by potrebné spojiť mnoho mohutných tabuliek. Oproti tomuto ponúka dimenzionálny model zvýšenie efektívnosti a rýchlosti dopytovania bez narušenia integrity dát. Na druhú stranu toto zlepšenie prináša väčšie nároky na ukladací priestor – dimenzionálny model môže obsahovať duplicity.

[2] Zašumené dáta obsahujú nesprávne, odľahlé hodnoty.
[3] Pri dátach typu dátum sú často používané odlišné formáty spôsobujúce nezrovnalosti.

Nahranie údajov do dátového skladu

Zdanlivo jednoduché naplnenie údajov z odkladacieho priestoru do skladu môže spôsobiť potrebu kombinovania stĺpcov, dohľadanie údajov. Takýto typ transformácie je potrebné previesť buď už pri získavania dát z ich zdrojov alebo práve v tejto fáze.

Vytvorenie súhrnných hodnôt

Po naplnení dimenzionálneho modelu s dátami sa vytvárajú takzvané agregáty – dopredu vypočítané súhrne hodnoty údajov. Vytváranie agregátov je časovo a pamäťovo náročné. Čím viacej, čím väčších dimenzií (sekcia 3.2) obsahuje model, tým dlhšie proces trvá. Proces sa dá zrýchliť poskytnutím väčšej operačnej pamäti.

Zaobstaranie nástroja pre generovanie správ

Reportovací nástroj pre podávanie správ generovaných z informácií ukladaných v dátovom sklade sa dá zabezpečiť kúpou už existujúcich riešení na trhu. Druhou možnosťou je navrhnúť a implementovať vlastný nástroj presne zodpovedajúci potrebám firmy. Vytvorenie nástrojov pre OLAP analýzu (sekcia 3.1) vyžaduje skúsených odborníkov a programátorov a môže byť časovo náročný.

2.5 Budúcnosť dátových skladov

V niektorých sa môže zrodiť otázka či má ešte zmysel zaoberať sa s takouto formou uchovávania dát, v dnešnom rýchlo meniacom sa svete. Podľa názorov expertov uvedených v [16] sklady dát majú naďalej veľkú budúcnosť pred sebou. Stávajú sa ešte kritickejšou súčasťou digitálneho sveta a stredovým bodom pre pochopenie zákazníkov a trhu.

Možnú budúcnosť vidia v tom, že organizácie premiestia tieto úložiská na cloud, ktorý zabezpečí im dostatočnú výpočtovú silu. Všestranne budú ponúkať prehľad o informáciách v reálnom čase i historických údajoch. Ako dátové prostredie, tak aj dátové sklady, sú čím ďalej tým viacej autonómne. To spôsobuje že, ich naplňovanie a prevádzkovanie bude vyžadovať menej zamestnancov a prinesie značné finančné úspory. Dátové sklady sa budú navzájom podporovať s umelou inteligenciou (anglicky *artificial intelligence*, skr. AI) a strojovým učením (anglicky *machine learning*). Úložiská dát budú pomáhať v prinášaní výsledkov s poskytovaním súborov dát a na druhej strane AI bude zlepšovať ich schopnosti a zrýchľovať operácie nad dátami.

Kapitola 3

OLAP analýza dát v dátovom sklade

Veľmi hodnotné poznatky o podnikoch môžu analytici a manažéri získať pomocou systémov zvaných OLAP (anglicky *On-line Analytical Processing*). Umožňujú užívateľom preskúmať údaje interaktívne, z viacerých perspektív 3.1.

Aby tieto informácie boli k dispozícii, je potrebné previesť veľký počet zložitých dotazov a operácií 3.4 nad uloženými informáciami. Aplikácie OLTP sa často postarajú o záchyt týchto dát [8].

Nástroje OLAP sú založené na multidimenzionálnom databázovom modely 3.2. Ich zobrazenie je uskutočnené pomocou dátových kociek 3.3. Umožňujú dáta modelovať vo forme viacerých dimenzii [7] a flexibilný pristúp k sumarizovaným dátam.

Medzi najznámejšie OLAP systémy patria MOLAP a ROLAP. Popis ďalších typov serverov z pohľadu uloženia dát a trojvrstvovej architektúry dátového skladu sa nachádza v sekcii 3.5

3.1 Definícia OLAP analýzy

Známa definícia od Edgara F. Codda predstavuje OLAP ako kategóriu softwarových nástrojov umožňujúcich analýzu dát v uložených v databázach.

OLAP analýza všeobecne označuje proces dopytovania textových a číselných dát z dátových skladov a trhov pre analytické účely. Vo veľkom podporuje prácu vedenia firiem v kritických situáciách, ako je rozhodovanie v strategických otázkach, vedúcich k neustálemu napredovaniu a rastu firmy. Pomocou týchto nástrojov je možné vytvárať rôzne scenáre pre skúmanie alternatívnych možností prevoditeľných v budúcnosti a vyhodnotiť ich vhodnosť a zmysel. Ďalej poskytujú možnosť previesť komplexné výpočty a hľadanie vzorov.

Často sa spája táto analýza s výrazom Business Intelligence (skratkou BI) – informačné systémy pre podporu plánovania v oblastiach podnikového manažmentu. Tvorí súčasť mnohých BI aplikácii.

Tieto nástroje pracujú s dátovými skladmi a trhmi 2.1 obsahujúce historické dáta, pri čom tabuľky v nich sú často nenormalizované. Musia byť optimalizované na prácu v jednom čase aj s miliónmi údajov pre zodpovedanie jediného dotazu a to za relatívne čím kratší čas.

Výraz v názvoch *On-line Analytical Processing* a *On-Line Transaction Processing* odkazuje na typ spracovania, pri ktorom počítač odpovie na daný dotaz ihneď, prípadne za krátky časový okamžik. Oproti možným mylným predstavám nemá nič spoločné s Internetom. Použitý zdroj [11].

Pravidlá pre OLAP

Edgar F. Codd spolu s kolegami stanovil nie len definíciu OLAP ale aj niekoľko pravidiel sťahujúcich sa na vlastnosti týkajúcich sa tohto systému [17]. Pár z týchto pravidiel :

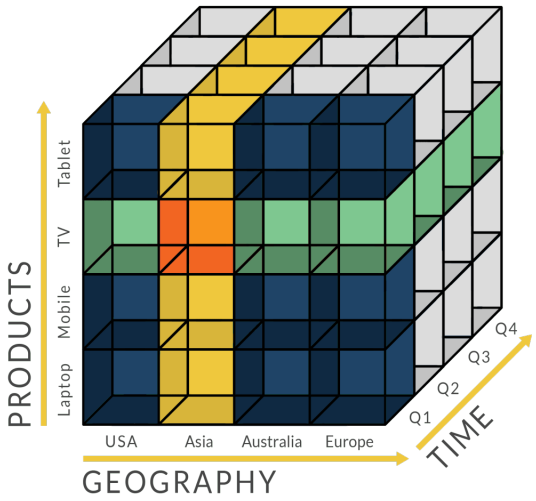
- Viacrozmerný konceptuálny pohľad – z pohľadu užívateľ, podnik má viacrozmerný charakter. Napríklad zisk môže byť zobrazený podľa regiónu, produktu, presného časového okamžiku. Viacrozmerný model umožňuje užívateľovi intuitívnejšie narábanie s dátami.
- Konzistentný výkon – nástroje OLAP musia zabezpečiť rovnaký výkon bez ohľadu na rastúci počet dimenzií v OLAP kockách.
- Ekvivalencia dimenzií – všetky dimenzie majú rovnakú štruktúru a operačné schopnosti.
- Neobmedzený počet dimenzií a úrovni agregácie – nekonečný počet podporovaných dimenzií a užívateľom definovaných agregácií nad dátami.

3.2 Multidimenzionálny databázový model

Multidimenzionálny dátový model tvorí základ pre OLAP nástroje. Rozhodovanie pomocou interaktívnej analýzy, nad množinou údajov, je kľúčovým faktorom pre tento model. Na rozdiel od iných technológii vytvára špeciálny pohľad na dáta v podobe viacrozmerných dátových kociek (anglicky *data cubes, hypercubes*). Umožňujú modelovať údaje ako viac dimenzionálne (obrázok 3.1). Tieto kocky sú navrhnuté tak aby boli vhodné pre analýzu dát [19]. Vytváranie dátových kociek je tiež dôležité pri konštruovaní dátových skladov. Výpočet častí tvoriacich kocku prevedený dopredu zabezpečí výrazne vnímateľné zvýšenie výkonu a vďaka tomu aj zníženie času potrebného na odozvu OLAP nástrojov [7]. Na druhú stranu tieto prípravy majú nemalý nárok na výpočtový čas a ukladací priestor.

V multidimenzionálnom modeli sú dáta členené do dvoch skupín. Buď sa jedná o fakty (anglicky *facts*), ktorým sa priraďujú, nadobúdajú číselné hodnoty, alebo dimenzie (anglicky *dimensions*), ktoré popisujú charakter práve týchto faktov. Oba tieto typy tabuliek sú nenormalizované a definujú hyperkocky.

Vďaka tomu že do tohto typu databáz sa už ukladajú dopredu zapracované dáta, zhrnuté a agregované údaje sa z nich dajú získať aj pomocou menej náročných operácii.



Obr. 3.1: Príklad dátovej kocky. Uvedené dimenzie na osách sú lokácia, produkty a čas. Oranžová bunka vznikne ako priesek člena TV v dimenzii produkty, člena Ázie v dimenzii lokácia a celého prvého kvartálu v dimenzii čas. Prevzaté z [18].

Dimenzia

Popisné informácie vzhľadom, na ktorých chce organizácia zhromažďovať a ukladať záznamy sa nazývajú dimenzie. Tvoria súčasť definície dátových kociek. Bližšie určujú okolité informácie spojité so sledovanými faktami. Pomáhajú zaviesť dáta do kontextu a s tým ich interpretovať na informácie. Kombinácia hodnôt dimenzii definuje obsah buniek v kocke.

Ku každej dimenzie sa vytvára vlastná tabuľka zvaná tabuľka dimenzie. V ňom umiestnené atribúty v podobe stĺpcov a podrobnejšie popisujú danú dimenziu. Atribúty sa používajú na vyhľadanie, triedenie faktov. Spojenie je medzi nimi a tabuľkou faktov vytvorené pomocou cudzích kľúčov (anglicky *foreign key*). Tieto tabuľky sú menšie ako tabuľka faktov a ich obsah sa aktualizuje iba po dlhších časových intervaloch [13]. Jeden z najpodstatnejších typov dimenzií je časová dimenzia. Použitý zdroj [19].

Ako príklad môže slúžiť dimenzia produkt a k nemu príslušná tabuľka s atribútmi meno, kategória, cena.

Tabuľka Dimenzie

Produkt	
PK	**ProduktID**
	NazovProduktu
	Kategoria
	Cena

Obr. 3.2: Príklad tabuľky dimenzií. Dimenzia produkt s atribútmi meno, kategória a cena produktu.

Tabuľka faktov

Tabuľka faktov, obsahujúca veľký objem dát, je primárnou a najväčšou v celej databáze. Fakty sú numerické merné jednotky – merné jednotky obchodovania [13]. Tieto jednotky sa dajú vnímať ako množstvo, na základe ktorého sú analyzované vzťahy medzi vybranými dimenziami. Fakty sa v rámci tabuľky môžu kombinovať alebo vypočítať pomocou iných faktov.

Reprezentujú základnú tému okolo, ktorej je daná dátová kocka organizovaná. Pre nutnosť spojenia s tabuľkami dimenzií obsahujú okrem spomenutých informácii aj cudzie kľúče.

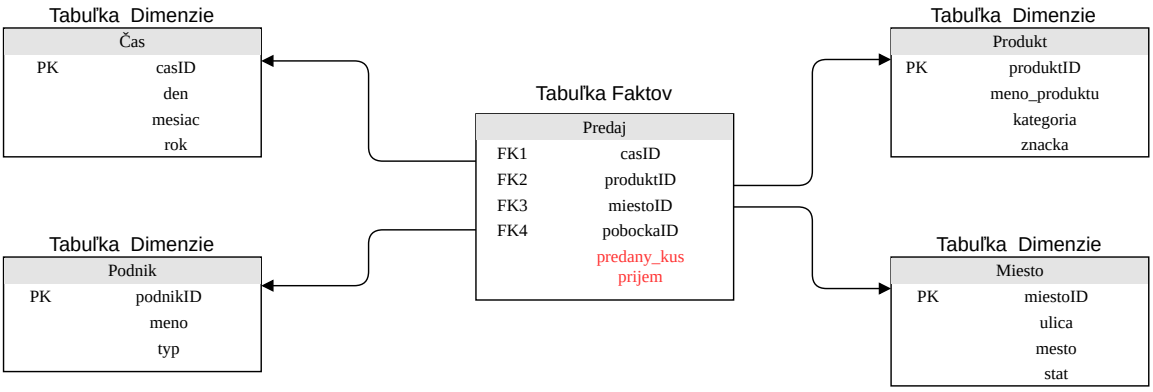
Dimenzie majú hierarchickú organizáciu skladajúcu sa z niekoľkých úrovní. Každá úroveň zodpovedá miere detailnosti. Inštancia alebo hodnota dimenzie patrí vždy do jednej z úrovní. Preto každý fakt má určitú granularitu – úroveň hĺbky dát [24] – danú úrovňou kombinovaných dimenzií, ktorý určujú práve tento fakt. Použitý zdroj [19].

Schémy multidimenzionálnych databáz

U dátových skladov, ako už bolo vysvetľované, najpoužívanejším dátovým modelom je viacrozmerný model existujúci často vo forme schém [7]. Tabuľka faktov a tabuľky dimenzii spolu vytvárajú práve tieto stručné a subjektovo orientované schémy. Existujú nasledujúce typy schém :

Schéma hviezdy

Najjednoduchšia schéma sa volá schéma hviezdy (anglicky *star schema*), skladajúca sa z tabuľky faktov a dimenzií [7]. Pomenovanie dostala na základe toho, že centrálna tabuľka faktov a s ním spojené dimenzie tvoria spolu štruktúru pripomínajúcu hviezdu (obrázok 3.3). Tabuľka faktov obsahuje veľké množstvo dát bez redundancie a cudzie kľúče od jed-



Obr. 3.3: Príklad schémy hviezdy. Schému tvory jedna tabuľka faktov a štyri tabuľky dimenzií. Ďalej sú vyznačené aj fakty.

notlivých tabuliek dimenzií. Jej primárny kľúč sa skladá zo všetkých jej cudzích kľúčov. Ak je potrebné spojiť tabuľky pri dotazoch tak spojenie je vytvorené iba cez tabuľku faktov.

Tabuľky dimenzií sú nenormalizované, neobsahujú relačné spojenie medzi sebou a vznikajú v nich redundancie. Majú vysoký dopytovací výkon – rýchlu odpoveď na otázky – vďaka tomu, že všetky údaje pre danú dimenziu sú, hoci redundantne, ale na jednom mieste [13]. Odpadá potreba ich skladať z viacerých relačných tabuliek, čo umožňuje menší počet spo-

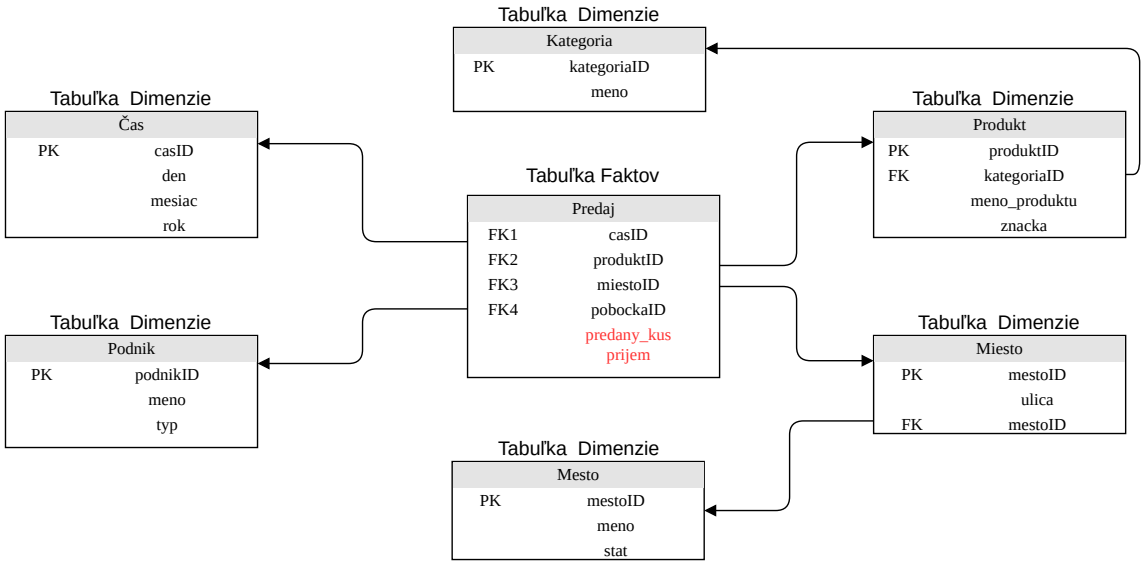
jení pri dotazoch. Na druhej strane s tým, že dáta sa opakujú, vytvorenie takéhoto modelu trvá dlhšie a spôsobuje väčšie nároky na pamäťový priestor.

Naplňovanie a aktualizácia modelu je jednoduchá. Obsah dimenzií je možné meniť osobitne bez zasiahnutia do ostatných dimenzií. Rozšírenie faktov je realizovateľné pridaním nových záznamov k tabuľke faktov.

Tento typ schémy je ľahko pochopiteľný pre užívateľov, vedia sa v ňom jednoducho zorientovať. Vďaka dobre navrhnutej schémy sú schopní po krátkom čase analyzovať mohutné súbory údajov.

Schéma snehovej vločky

Aj pri tejto schéme je názov snehová vločka (anglicky *snowflake schema*) odvodený od vzhľadu štruktúry výsledného modelu (obrázok 3.4). Jedná sa o modifikovanú verziu schémy hviezdy [7]. Hlavná zmena spočíva v tom, že niektoré tabuľky dimenzií sú už normalizované. Teda sú rozdelené do viacerých častí a vznikajú nové tabuľky. Čiastočným odstránením redundancie sa usporí vzácny pamäťový priestor a údržba sa stáva jednoduchšou. Napriek tomu táto úspora stráca význam v porovnaní s rozmerom tabuľky faktov. Pri vytváraní



Obr. 3.4: Príklad schémy snehová vločka. Schému tvory jedna tabuľka faktov.

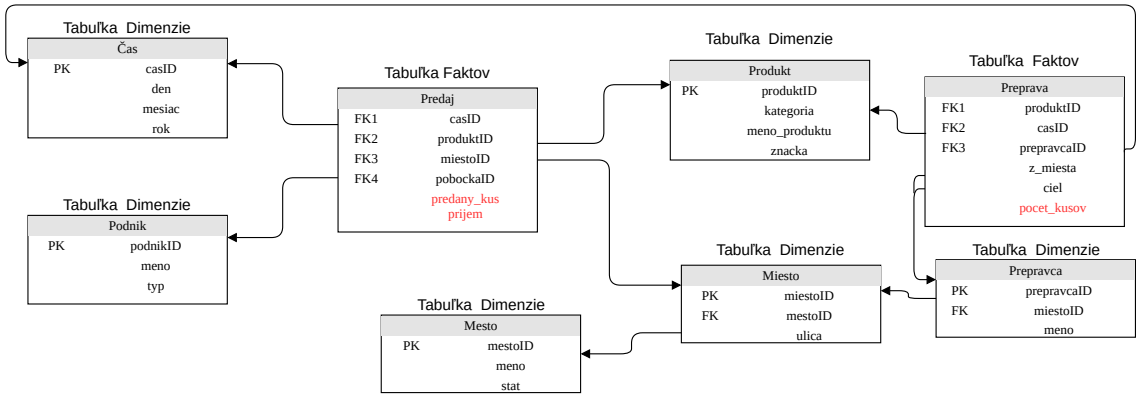
niektorých dimenzií sa použije veľa relačne zviazaných tabuliek čo značne degraduje dopytovací výkon [13]. Tabuľky sa spájajú centrálnou tabuľkou nepriamo, cez iné tabuľky. Táto degradácia spôsobuje to, že pri dotazoch je potrebné previesť oveľa viac spojení medzi tabuľkami.

Vďaka mnohým spojeniam sa dotazy stávajú menej prehľadnými a pochopiteľnými pre užívateľov. Vymenované záporné stránky schémy vedú k tomu, že jej používanie značne zaostáva za schémou hviezdy.

Schéma súhvezdie

Niektoré aplikácie môžu vyžadovať existenciu viacerých tabuliek faktov. Často sa označujú ako schéma súhvezdie (anglicky *galaxy schema* alebo *fact constellation schema*), lebo ich štruktúra predstavuje kolekciu schémy hviezd (obrázok 3.5). Tieto tabuľky medzi sebou

zdieľajú niekoľko tabuliek dimenzií [7]. Počet dimenzií by malo odpovedať úrovniam hierarchie detailnosti, každá úroveň má vlastnú tabuľku. Napríklad pôvodná dimenzia pre lokáciu s atribútmi mesto, krajina, kontinent bude rozdelená do troch dimenzií. Pri návrhu tohto typu modelu je potrebné brať ohľad na veľa možných kombinácií spojení pri dotazoch.



Obr. 3.5: Príklad schémy súhvezdie. Schému obsahuje dve tabuľka faktov a niekoľko tabuľky dimenzií.

3.3 Dátové kocky a spôsob ich zobrazovania

Multidimenzionálna databáza zobrazuje dáta v podobe dátových kociek. Tieto kocky generalizujú, pretvarujú tabuľky z databáz do dimenzií [19]. Sú navrhnuté tak aby podporovali zoradenie dimenzií do hierarchií a pritom ich definícia zostala bez duplicít. Na základe dátových skladov alebo multidimenzionálnych databáz je možné vytvárať kolekciu vzájomne súvisiacich kociek. Každá dimenzia je viazaná s prislúchajúcimi doménami z tabuliek umožňujúca jednoduché pridávanie nových hodnôt.

Bunky reprezentujú jedinečný logický priesek jedného, prípadne viacerých atribútov z dimenzií v kocke. Obsahujú hodnotu sledovaného faktu, prípadne viacerých faktov.

Slovo kocka implikuje v myšlienkach trojrozmerné teleso, 3D geometrickú štruktúru. V rámci dátových skladov kocka má trochu inú vlastnosť, môže byť celkovo n - dimenzionálna [7], neobmedzujú sa iba na tri dimenzie. V skutočnosti najviac v reálnom svete používaných kociek má štyri až niekedy dvanásť dimenzií. Takéto kocky sa už nazývajú hyperkocky. Zvyšujúcim sa počtom dimenzií sa často u OLAP nástrojov objavujú problémy s výkonom [19].

Všeobecne kocka umožňuje vidieť dva, maximálne tri dimenzie naraz. Sú prípady keď je štvrtá dimenzia vnorená do niektorého z troch na jednej osi [19].

Najjednoduchším prípadom sú dvoj dimenzionálne kocky, ktoré sú v skutočnosti jednoduché dátové tabuľky [7]. Príkladom môže byť tabuľka pre predaj produktov v závislosti na dimenziách čas a miesto (obrázok 3.6).

miesto = "Bratislava"

čas = "rok"	**produkt** PC	Tlačiareň	Fax	Monitor
1998	259	789	551	359
1999	562	301	198	421
2000	1247	50	65	256

Obr. 3.6: Príklad dvoj dimenzionálnej kocky. Dáta (počet predaných kusov) sú zobrazené pre konkrétne mesto „Bratislava“ podľa jednotlivých rokov.

Ak pridáme ďalšiu dimenziu v podobe produktov a zobrazíme podľa mesta predaja dostávame trojdimenzionálnu kocku (obrázok 3.7). Troj a viac dimenzionálne kocky je možné zobrazovať ako sériu (n - 1) dimenzionálnych kociek, kde n je počet dimenzií.

	miesto = "Bratislava"				**miesto** = "Žilina"				**miesto** = "Košice"			
	produkt				**produkt**				**produkt**			
čas = "rok"	PC	Tlačiareň	Fax	Monitor	PC	Tlačiareň	Fax	Monitor	PC	Tlačiareň	Fax	Monitor
1998	259	789	551	359	259	789	551	359	259	789	551	359
1999	562	301	198	421	562	301	198	421	562	301	198	421
2000	1247	50	65	256	1247	50	65	256	1247	50	65	256

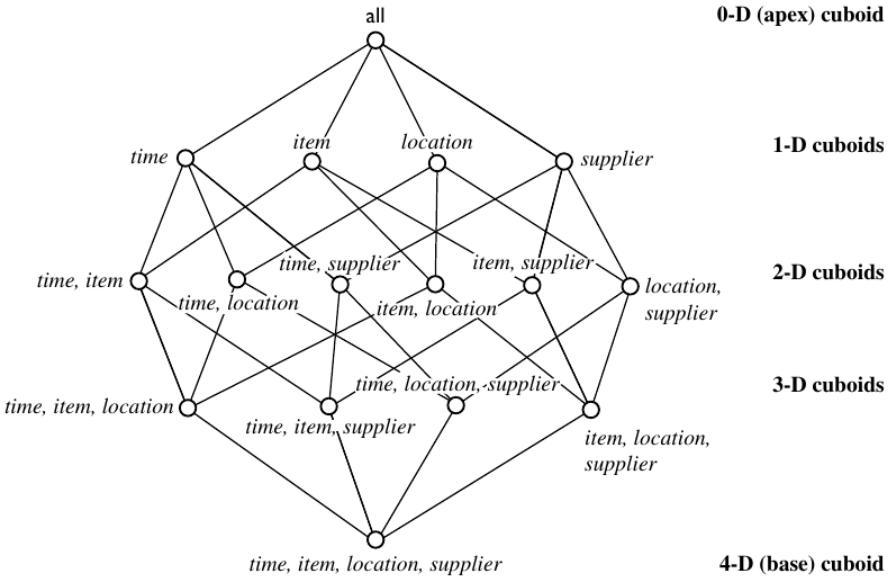
Obr. 3.7: Príklad trojdimenzionálnej kocky zobrazená ako séria 2D kociek. Dáta (počet predaných kusov) sú zobrazené pre všetky mestá v dimenzií miesto podľa jednotlivých rokov.

OLAP cuboid, jeden z možných spôsobov, ako zobraziť kocky, je podmnožina agregovaných metrík nad všetkými dimenziami. Je možné vygenerovať cuboid pre všetky možné kombinácie podmnožín dimenzií, ktoré zobrazujú dáta na rôznych úrovniach agregácie. Ako výsledok tvoria mriežku cuboidov (obrázok 3.8). Cuboid obsahujúci najnižší úroveň sumarizácie, teda je najdetailnejšou kockou, sa označuje ako základný cuboid. Ten, ktorý naopak obsahuje najvyššiu úroveň, má nula dimenzií a označuje sa vrchný (anglicky *apex*) cuboid. Zdroj [7].

V pokročilejších databázových aplikáciách zameraných na OLAP analýzu je možnosť zobraziť dátové kocky graficky v interaktívnej viac dimenzionálnej podobe.

3.4 Základné OLAP operácie

Kladnou stránkou multidimenzionálneho modelu je to, že je organizovaný tak, aby ponúkal dostatočnú flexibilitu užívateľom, hľadieť na dáta z rôznych perspektív [7]. V rámci modelu sú dáta usporiadané do dimenzií. Jednou z ich vlastností je to že obsahujú niekoľko úrovní abstrakcie, ktoré definuje koncepčná hierarchia.



Obr. 3.8: Príklad mriežky cuboidov. Prevzaté z [7].

OLAP vytvára užívateľsky prívetivé prostredie, v rámci ktorého je možné interaktívne analyzovať dáta. Aby bolo možné preskúmať spomenuté rôzne pohľady na dáta, OLAP ponúka niekoľko operácií špecializovaných na dátové kocky. Sú použiteľné na vytvorenie interaktívnych dotazov a hľadanie záznamov. Medzi najpoužívanejšie operácie patria Drill-down, Roll-up, Slice a Dice. Hlavný zdroj pre celú túto sekciu je [7].

Koncepčná hierarchia

Koncepčná hierarchia predstavuje mapovanie medzi množinou konceptov s nižšou úrovňou a množinou konceptov s vyššou úrovňou. Umožňujú manipulovať s informáciami na rôznych úrovniach abstrakcie.

Niektoré koncepčné hierarchie je možné stanoviť jednoducho z atribútov tabuľky danej dimenzie. Ako napríklad pre dimenziu miesto, obsahujúce atribúty zvané ulica, mesto, kraj, štát, vieme na prvý pohľad stanoviť ich poradie, ktoré bude na najvyššej, respektíve najnižšej úrovni. Hierarchia je zobrazená na obrázku 3.9.

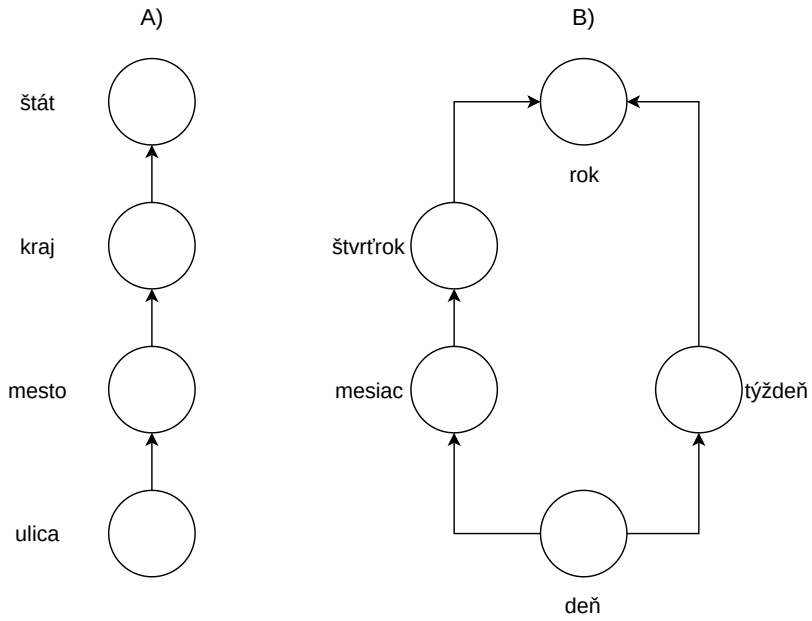
Špeciálnym typom koncepčnej hierarchie je mriežková hierarchia, v ktorej je možnosť dostať sa do jednej vyššej úrovne z dvoch alebo viac nižších úrovní alebo naopak. Atribúty sú usporiadané v čiastočnom poradí. Príklad na obrázku 3.9.

Pre jednu dimenziu môže existovať viacero hierarchií usporiadaných na základe toho ako užívateľ vníma význam danej dimenzie alebo jej atribúty. Zdroj [7].

Granularita dát

Granularita, iným slovom zrnitosť, vyjadruje úroveň hĺbky reprezentácie dát [24]. Definuje ako sú podrobné jednotlivé dáta.

Ak je zrnitosť údaju vysoká, znamená to, že je veľmi detailná, môže predstavovať atomickú hodnotu, zrno v tabuľke. Typickým príkladom môžu byť transakcie u OLTP. V opač-



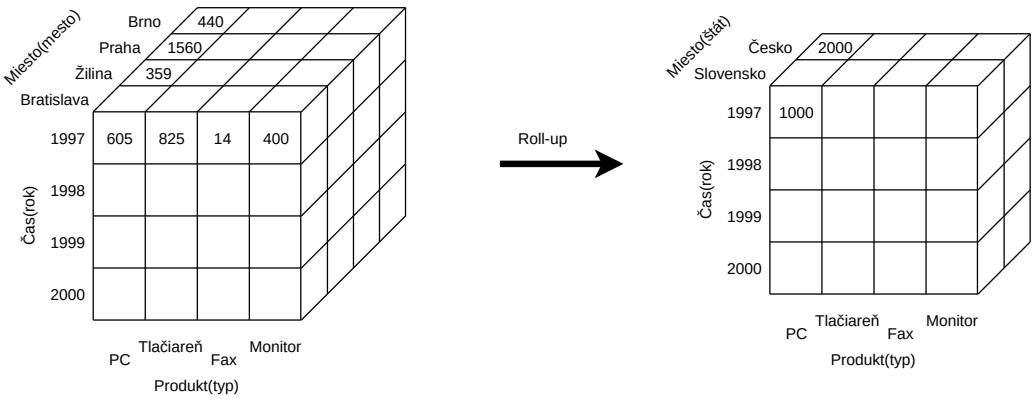
Obr. 3.9: Príklad variant koncepčnej hierarchie. Po A) je znázornená klasická hierarchia. Po B) je znázornená mriežková hierarchia.

nom prípade, ak granularita je nízka, dáta sú zoskupené do väčších celkov. Takýto celok reprezentuje zvyčajne agregovaná hodnota.

U dátových skladoch sa najčastejšie používajú agregované dáta urýchľujúce odpovede na dotazy. V momente hľadania odpovedi nie je potrebné ešte čakať na sumarizáciu dát s vysokou zrnitosťou.

Operácia Roll-up

Operácia je známa aj pod menom Drill-up. V rámci tejto operácie je prevedená agregácia na dátovej kocke. Táto agregácia je prevoditeľná buď s posunutím sa o jednu úroveň vyššie, v koncepčnej hierarchií, alebo znížením počtu zahrnutých dimenzií. Dáta po prevedení operácie budú mať v dátovej kocke nižšiu granularitu. Obrázok 3.10 znázorňuje prevedenie Roll-upu.



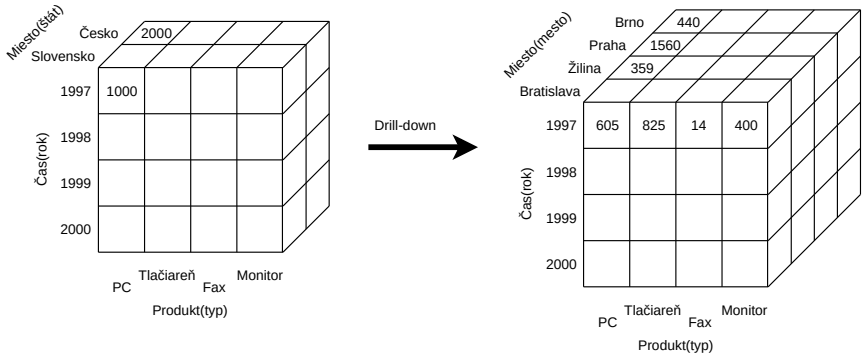
Obr. 3.10: Príklad operácie Roll-up.

Ak je prevedená varianta operácie, pri ktorej sa redukuje počet dimenzií, tak daná dimenzia je odstránená z hyperkocky. Príkladom môže byť kocka pre predaj tovaru, obsahujúca dimenzie čas a lokácia. Ak sa odstráni dimenzia lokácie, tak zisk z predaja bude sumarizovaný iba podľa času. Posunutie sa na najvyššiu úroveň v rámci hierarchie niektorej z dimenzii, tiež spôsobuje jej odstránenie [19].

Operácia Drill-down

Drill-down je inverzným párom operácie Roll-up. Prevedením operácie sa v rámci dátovej kocky dostávame k detailnejším úrovniam. Zvýši sa granularita dát. Operácia môže byť prevedená buď s posunutím sa o úroveň nižšie v koncepčnej hierarchií alebo zvýšením počtu zahrnutých dimenzií. Obrázok 3.11 znázorňuje prevedenie Drill-downu.

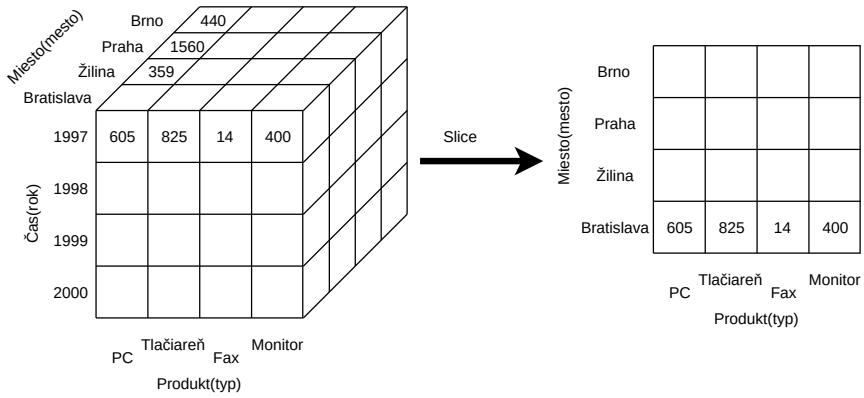
S tým, že zobrazenie dát sa dostane na detailnejšiu úroveň môže spôsobiť pridanie nových dimenzií do dátovej kocky.



Obr. 3.11: Príklad operácie Drill-down.

Operácia Slice

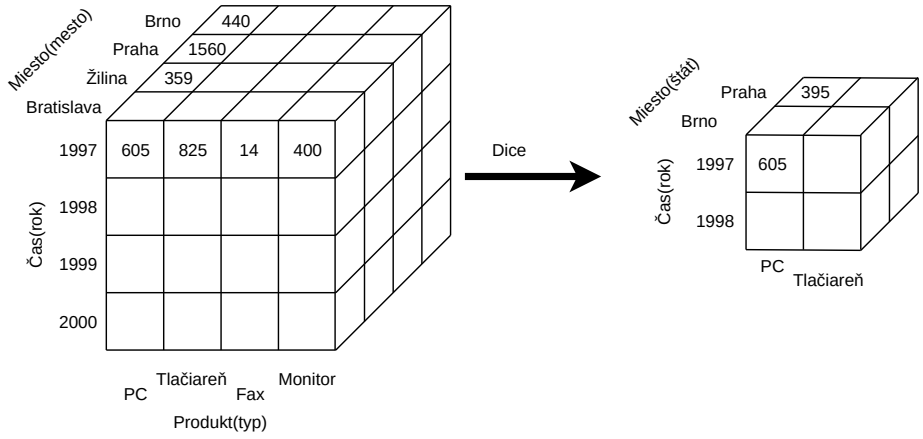
V rámci tejto operácie je prevedená selekcia nad jednou dimenziou, ktorá vedie k vytvoreniu podkocky. Výber jednej dimenzie kocky spôsobí zníženie počtu rozmerov dátovej kocky [19]. Napríklad použitie operácie nad kockou s troma dimenziami vytvorí dvojrozmernú tabuľku. Obrázok 3.12 znázorňuje prevedenie operácie Slice.



Obr. 3.12: Príklad operácie Slice.

Operácia Dice

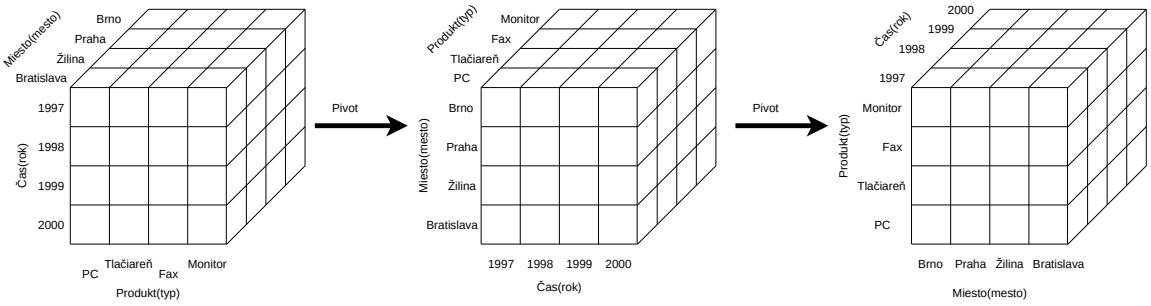
Operácia Dice prevedie vytvorenie menšej kocky, podkocky s výberom dvoch, prípadne viacerých dimenzií z pôvodnej kocky alebo určením intervalu hodnôt v rámci jednej dimenzie. Zvyčajne nedôjde k zníženiu počtu dimenzií. Obrázok 3.13 znázorňuje prevedenie operácie Dice.



Obr. 3.13: Príklad operácie Dice.

Operácia Pivot

Operácia tiež nazývaná ako Rotate, má vplyv na vizualizáciu kocky – je to vizualizačná operácia. Jedná sa o rotáciu dimenzií s cieľom vidieť dáta z inej perspektívy, v prehľadnejšej forme. Môže sa jednať aj o zobrazenie 3D kocky ako množina 2D kociek. Obrázok 3.14 znázorňuje prevedenie operácie Pivot.



Obr. 3.14: Príklad operácie Pivot.

Inverzné operácie Drill-down a Roll-up môžu byť prevedené spolu s Dice a Slice [19].

3.5 Typy OLAP systémov

Multidimenzionálne dáta sú prezentované užívateľom, analytikom bez toho, aby sa museli zaoberať s tým, odkiaľ dáta pochádzajú a ako sú zhromažďované. Práve OLAP servery zabezpečujú túto prezentáciu, starajú sa o získavanie dát z dátových skladov alebo trhov [7].

Relačný OLAP

Relačné online analytické spracovanie údajov (anglicky *Relational On-Line Analytical Processing*, skratkou ROLAP) pre analýzu získava informácie z relačných dátových skladov [13]. Užívateľom poskytuje multidimenzionálny pohľad tvorený zapracovaním dát. Okrem dát sú ukladané aj metadáta v podobe záznamov. Metadáta dynamicky sa používajú na generovanie SQL príkazov pri dopytovaní.

Relačný OLAP systémy používajú špeciálnu indexáciu, ako je napríklad bit-mapové indexovanie, indexové štruktúry. Cieľom toho je zvýšenie výkonu obsluhy dotazov [19]. Zvyčajne ako schéma pre multidimenzionálnu databázu sa používajú schémy hviezdy alebo snehovej vločky. Tabuľka faktov obsahuje stĺpce pre jednotlivé metriky a cudzie kľúče dimenzií.

ROLAP servery tvoria prostriedok medzi relačným databázovým serverom a klientskym rozhraním [7]. Dáta z dátových skladov sú udržiavané v relačných databázach, tabuľkách. OLAP nástroje doplňujú funkcie potrebné pre analýzu.

Dôležitou výhodou týchto systémov je veľká škálovateľnosť.

Multidimenzionálny OLAP

Multidimenzionálny online analytické spracovanie údajov (anglicky *Multidimensional On-Line Analytical Processing*, skratkou MOLAP) získava údaje z dátových skladov, ktoré ukladá do vlastných dátových štruktúr [13]. V multidimenzinálnych úložiskách založených na poliach majú tieto štruktúry podobu dátových kociek [7]. Počas získavania dát sú nad nimi prevedené výpočty, následne sa ukladajú už v agregovanej forme do kociek. Organizácia databázy sa zameriava na urýchlenie vyhľadávania dát z viacerých dimenzií.

MOLAP systémy obsahujú opatrenia pre prácu s riedkymi poľami dát. Používajú indexovanie a hashovanie pre jednoduchšie vyhľadanie dát pri odpovedaní na dotazy [19]. Tiež ponúkajú možnosť šetrenia siete od zaťaženia so zavedením údajov zo strany serveru ku klientovi [13].

Nevýhodou tohto typu systému je, že môže dôjsť značnej nadbytočnosti dát a potrebného úložného priestoru pri používaní mnohých dimenzií. Dôvodom je, že dáta sa ukladajú naraz na dvoch miestach, v relačnej databáze a multidimenzionálnej databáze.

Hybridný OLAP

Hybridný OLAP (anglicky *Hybrid OLAP*, skratkou HOLAP) kombinuje v sebe najlepšie vlastnosti MOLAP – rýchlosť výpočtov – a ROLAP – veľká škálovateľnosť [7]. Detailné údaje sa ukladajú do relačných databáz a agregované do multidimenzionálnych modelov.

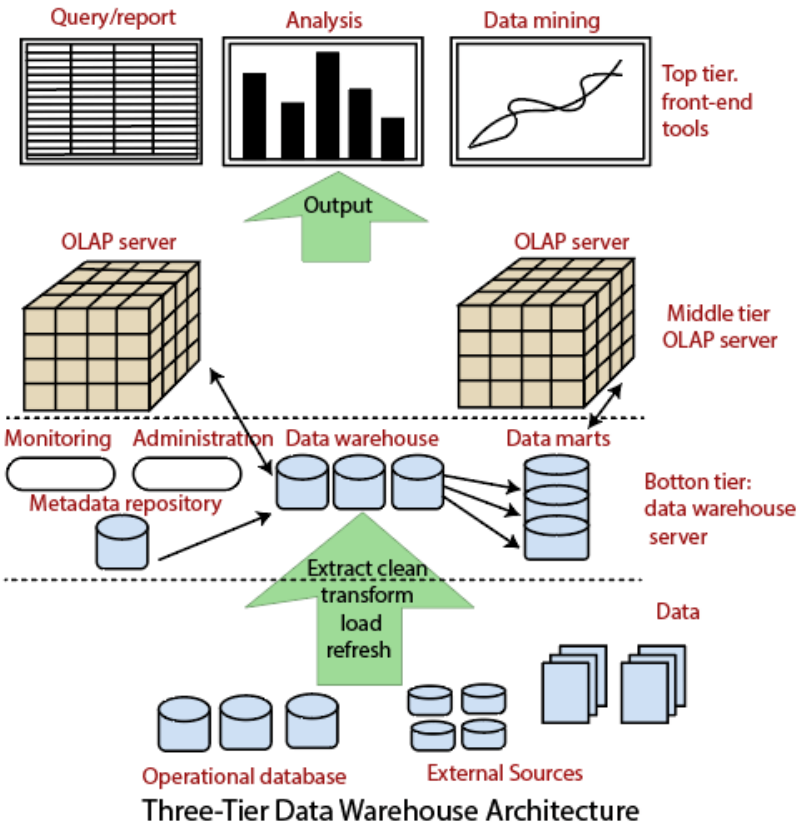
Trojvrstvová architektúra dátového skladu

Architektúra dátových skladov sa skladá z viacerých na seba nadväzujúcich vrstiev. Každá vrstva má iný účel v rámci skladu. Pri implementáciách sa najčastejšie používa trojvrstvová architektúra, ktorá sa skladá z nasledujúcich častí [7] :

- **Server dátového skladu:** Tvorí spodnú vrstvu architektúry. Je realizovaná vo väčšine prípadov ako relačný databázový server naplňovaný z operačných databáz a iných zdrojov dát. Nástroje určené pre naplňovanie, mimo iných, prevedú aj operácie ako je extrakcia, čistenie, transformácia, zabezpečujúca aktuálnosť obsahu skladu. Táto

vrstva do seba zahrňuje aj úložisko metadát obsahujúca popisné informácie o samotnom sklade a jeho obsahu. Aplikačné rozhranie, označované aj ako brána, poskytuje možnosť spúšťania databázových dotazov na danom serveryi.

- **OLAP server:** Stredná vrstva architektúry je tvorená OLAP serverom. Jeho implementácia je prevedená podľa jedného zo spomenutých typov OLAP systémov: buď ROLAP alebo MOLAP.
- **Klient:** Vrstva na najvyššej úrovni je tvorená klientskym rozhraním obsahujúce rôzne nástroje pre dopytovanie a podávanie správ, analýzu údajov obsadených na OLAP serveri.



Obr. 3.15: Trojvrstvová architektúra dátového skladu. Zdroj [7].

Kapitola 4

Návrh aplikácie

Na základe zadania bolo potrebné navrhnúť aplikáciu, ktorá užívateľovi umožní zobrazovať OLAP dátové kocky vo vhodnej podobe. V nasledujúcej kapitole sú uvedené požiadavky a vlastnosti týkajúce sa jednoduchej aplikácie navrhnutej a implementovanej v rámci tejto bakalárskej práce. Podkapitoly obsahujú popis plánovanej realizácie požadovaných funkcionalít.

Pred samotným návrhom bol prevedený prieskum už existujúcich riešení v rámci danej tematiky (sekcia 4.1), aby sa zistilo, aké základné operácie ponúkajú svojím zákazníkom.

Požiadavky týkajúce sa výsledného programu (sekcia 4.2) boli stanovené podľa zadania a z hľadiska očakávaní možného potencionálneho užívateľa.

V sekcií 4.3 sú uvedené najpodstatnejšie vlastnosti vyplývajúce z analýzy uvedených požiadaviek na aplikáciu.

4.1 Prieskum existujúcich riešení

Na trhu už existuje mnoho komerčných aplikácií zameriavajúcich sa na vytváranie a prácu s dátovými skladmi, kockami a prevedenie OLAP analýzy nad nimi. Príkladom môžu byť veľké firmy a ich nástroje ako napríklad *SQL Server Analysis service* (skratkou SSAS) od Microsoft Corporation alebo *Oracle Database OLAP Option* od spoločnosti Oracle.

Zaujímavou aplikáciou je tiež *CubesViewer*[1], ktorá je responzívna HTML5 aplikácia pre skúmanie a vizualizáciu datasetov rôznych typov. Zahrňuje základné nástroje pre vytváranie reportov, generovanie grafov z prístupných informácii a celopodnikovú analýzu dát. Vďaka tomu, že sa jedná o webovú aplikáciu ponúka jednoduchý prístup k dátam cez ktorýkoľvek webový prehliadač bez nutnosti iného vybavenia. Dostupný je ako program otvoreného zdrojového kódu (anglicky *open source*).

4.2 Požiadavky týkajúce sa aplikácie

Pri návrhu aplikácie sa zohľadnilo zadanie samotnej práce a výsledky hľadania možných potrieb potencionálnych užívateľov a už existujúcich riešení.

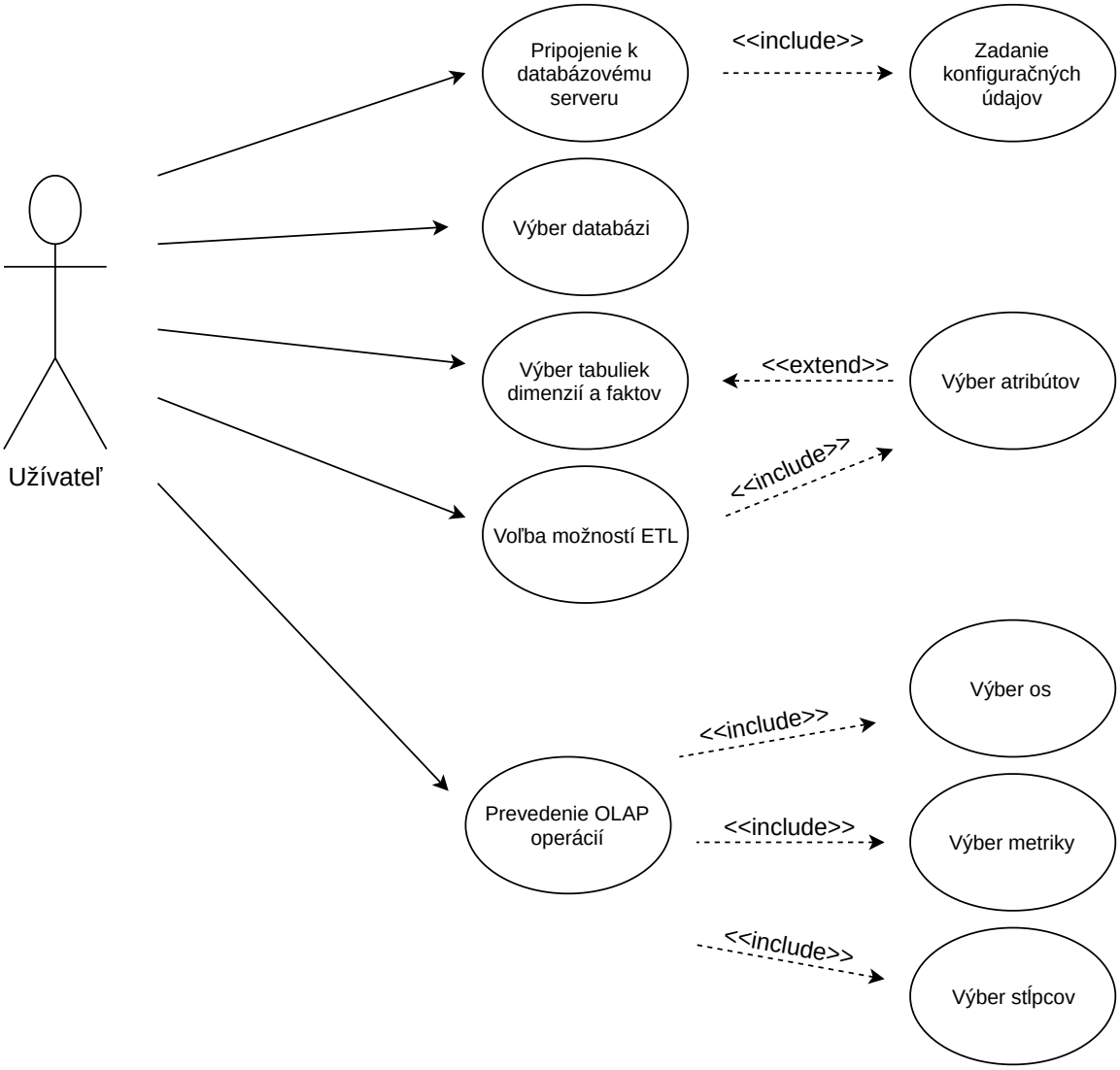
Zo zadania práce vyplýva potreba navrhnúť aplikáciu, ktorá má nasledujúce funkčné požiadavky :

- Práca s databázovým systémom typu PostgreSQL (sekcia 5.1).

[1] http://www.cubesviewer.com/

- Vykonávanie jednoduchej etapy ETL (sekcia 2.4) nad dátami.
- Zobrazovanie vytvorených dátových kociek vo vhodnej podobe.
- Podporovanie základných operácií OLAP analýzy nad dátami.

Ako potencionálny užívateľ sa predstavuje manažér pracujúci v malej, prípadne strednej organizácii. Organizácia sa zaoberá predajom určitého typu tovaru. Ako súčasť svojej práce má vedenie konkrétneho oddelenia alebo celého podniku. Viac krát mesačne musí odcestovať na krátke firemné cesty. K splneniu svojich povinností potrebuje mať možnosť rýchlo a efektívne previesť OLAP analýzu nad historickými dátami, napríklad o všetkých predajoch firmy za posledných päť rokov. K týmto dátam je potrebné, aby mal prístup aj počas svojich pracovných ciest. Vytvorený diagram prípadov použitia (anglicky *Use Case Diagram*)[2] na základe uvedených abstraktných požiadaviek je znázornený na obrázku 4.1 :



Obr. 4.1: Prípad užitia.

[2]Diagram prípadov použitia špecifikuje možnosti použitia systému. Pre jej tvorbu sa používa jazyk UML.

Pripojenie k databázovému serveru

Databázový server je fyzický počítač dedikovaný na hosťovanie databázy a systému riadenia bázy dát (anglicky *database management system*, skratka DBMS) [22]. Server je nezávislý na architektúre databázy preto ich môže naraz obsahovať viac, bez ohľadu na ich typ. Využíva sa hlavne na vyhľadanie záznamov uložených v databáze na základe požiadaviek od používateľov a vrátenie nájdených dát.

Server môže byť umiestnený buď priamo v rámci lokálnej sieti alebo mimo túto sieť, dostupný pomocou napríklad internetového pripojenia. Každý server má priradenú vlastnú adresu a unikátne číslo portu, na ktorom je dostupný.

V prípade ak sa uvažuje u programoch o základnom klient - server modeli databázový server je softvérový a tvorí súčasť back - endu aplikácie. V ojedinelých prípadoch sa s týmto názvom označuje kombinácia technického vybavenia aj programu spoločne.

Klient, aby mohol pristúpiť k údajom uložených v databáze, musí sa pripojiť k databázovému serveru. K prevedeniu tohto úkonu musí poznať spomenutú adresu a číslo portu pre naviazanie spojenia, prihlasovacie údaje, zvyčajne prihlasovacie meno a heslo, aby sa dali overiť jeho práva čítania, respektíve modifikovania uložených dát.

Z týchto faktov vyplýva, že je potrebné zaistiť užívateľovi začiatočný bod v aplikácií, kde si vie zadať údaje k naviazaniu spojenia s databázovým serverom. Prípadne zabezpečiť možnosť otestovať či daný server je v danom momente dostupný.

Výber databázy

Databáza je usporiadaná kolekcia dát ukladaná v počítači. Dáta sú často zoskupené podľa rôznych tematík. Tie, ktoré patria k tej istej téme sa ukladajú spolu zvyčajne do jednej databázy. Napríklad malá firma s jednou predajňou môže mať jednu databázu, v ktorej bude uchovávať všetky údaje o svojich zamestnancoch, obsahu skladu a predanom tovare. Ale organizácie, ktoré disponujú s viacerými predajňami, rozprestreté v celom svete, budú si ukladať pozbierané dáta do viacerých databáz. Napríklad podľa jednotlivých predajní. Takto môžu dosiahnuť lepšiu prehľadnosť a efektívnejšie využitie databáz. Ich databázový server teda bude obsahovať naraz viacero databáz.

Užívateľovi je potrebné umožniť výber z dostupných databáz na danom serveri. Cieľom je aby mohol prechádzať, analyzovať iba tie údaje, ktoré sú z hľadiska jeho záujmu a potrieb relevantné.

Výber tabuliek databázy

Relačné databázy ukladajú dáta vo forme tabuliek, ktoré majú medzi sebou vzťahy a tak sú navzájom prepojené [25]. Každá tabuľka sa skladá zo stĺpcov a riadkov, zo schémy relácie a n-tic. Príkladom môže byť tabuľka pre tovar s atribútmi meno, typ, kategória, cena.

Pre prevedenie analýzy užívateľ nemusí využiť všetky prítomné tabuľky v danej databáze, preto je mu potrebné umožniť výber z nich. OLAP analýza (sekcia 3.1) k svojmu správnemu fungovaniu potrebuje, aby jedna tabuľka bola označená ako tabuľka faktov a aspoň jedna ako tabuľka dimenzie.

ETL operácie

Organizácie, ako bolo spomenuté, vlastnia niekoľko odlišných databáz. Z týchto databáz sa vytvárajú dátové sklady uskladňujúce historické dáta prevedením integrácie (sekcia 2.2).

Pri vytváraní je potrebné brať ohľad na možnosť výskytu chýbajúcich hodnôt v stĺpcoch relačných tabuliek. Ďalším typom častých chýb môže byť zadanie hodnôt, ktoré sú mimo interval alebo množinu očakávaných odpovedí. Zdrojom spomenutých vád sú často zle navrhnuté formuláre, ktoré nekontrolujú spôsob vyplnenia jednotlivých polí pre odpovede, alebo sú také polia, ktorých vyplnenie je dobrovoľné. Užívateľ má možnosť rozhodnúť sa, či do nich chce uviesť vyžadované informácie alebo ich nechá nevyplnené. Preto, ako bolo uvedené aj v sekcii 2.4, pred samotnou integráciou sa používajú nástroje ako je napríklad dátová pumpa. Odstraňuje takéto a podobné nedostatky, zničujúce kvalitu dát a presnosť výsledkov analýz.

Užívateľovi je vhodné poskytnúť možnosť na čistenie dát. Táto činnosť je uskutočniteľná v praxi využitím odlišných algoritmov a postupov, ktoré nie sú v rámci tejto práci rozoberané. Základom je umožniť aspoň zistenie toho či tabuľka vybraná podľa jeho záujmu obsahuje hodnoty reprezentujúce neprítomnosť dát. Neprítomnosť údaju je typicky reprezentovaná buď hodnotou typu NULL alebo prázdnym reťazcom. Na základe tejto informácie užívateľ sa rozhodne o tom, či danú tabuľku zahrnie do analýzy alebo nie. Bude si vedomý toho, že získané výsledky sú veľkou pravdepodobnosťou do určitej miery skreslené, ovplyvnené.

Prevedenie OLAP operácií

Multidimenzionálny model tvorí základ dátového skladu (sekcia 3.2), v ktorom sa dáta organizujú do dimenzií [7]. Dimenzie obsahujú rôzne úrovne abstrakcie dát spoločne tvoriace konceptuálnu hierarchiu.

Pre umožnenie flexibilného a interaktívneho dopytovania nad užívateľom vybranými dátami je potrebné im poskytnúť OLAP operácie (sekcia 3.4). Použitím týchto operácii vedia jednoducho analyzovať svoje dáta z viacerých perspektív. Môžu sa posúvať medzi jednotlivými úrovňami konceptuálnej hierarchie a tak zobrazovať dáta inými úrovňami podrobnosti.

4.3 Vlastnosti navrhnutej aplikácie

Aplikácia je navrhnutá ako webová aplikácia s cieľom jednoduchého prístupu bez ohľadu na to, kde sa užívateľ práve nachádza. Pri návrhu sa postupovalo tak, aby výsledný program bol čo najuniverzálnejší týkajúci sa nasadenia a dátových skladov, s ktorými bude vedieť spolupracovať. Pre zjednodušenie implementácie niektorých operácií a po konzultácií s vedúcim sa rozhodlo podporovať iba tie dátové sklady, ktoré využívajú schému hviezdy (3.2). Podrobnejší rozbor niektorých zo uvedených vlastností a odôvodnenie ich výberu :

Webová aplikácia

V dnešnej dobe vytváranie webových aplikácii je veľmi rozšírené. Mnoho aplikácii sa realizuje touto formou. Možným dôvodom, vďaka ktorému sa tak rozšírilo, je jednoduchý prístupu k rýchlemu internetovému pripojeniu dostupného vo väčšine štátov, niekedy aj zadarmo.

Ako typ aplikácie bolo vybrané kvôli spomenutému jednoduchému prístupu, buď cez internet alebo lokálnu sieť. Závisí iba na tom, kde konkrétne je databázový server umiestnený. Ďalším dôvodom je to, že ako klient postačuje jednoduchý webový prehliadač (tenký klient), ktorý vie zobrazovať základné webové stránky a podporuje JavaScript. Nie je potrebné

nakupovať silné počítače pre všetkých zamestnancov, ktoré by zvládali beh komplexnejších aplikácii.

Webové prehliadače obsahujú dnes už všetky mobilné telefóny a tablety, preto aj počas cestovania na konferencie a mimo kancelárie môžu užívatelia jednoducho pristupovať k dátam podľa potreby a zistiť potrebné informácie. Toto riešenie ponúka veľkú flexibilitu týkajúcu sa používaného technického vybavenia.

Očakávaný typ schémy

Podľa návrhu aplikácia po pripojení sa na databázový server bude očakávať výber databázy, ktorá bude už pripravená pre OLAP analýzu. Teda nebude obsahovať nástroje pre vytvorenie dátového skladu, ale bude sa zameriavať na zobrazenie v ňom uložených dáta a možné operácie nad nimi. Vytvorenie skladu pre dát bude potrebné previesť v predstihu, pomocou iných databázových nástrojov alebo SQL skriptov.

Najpoužívanejšou modelovacou paradigmou pri multidimenzionálnom modelovaní je schéma hviezdy (3.2). Preto aplikácia u vybraného dátového skladu bude očakávať, že je vybudovaná práve na základe tejto schémy. Užívateľ si bude vyberať práve jednu tabuľku faktov a niekoľko tabuliek dimenzii z prítomných možností. Toto očakávanie znamená zjednodušenie implementácie niektorých operácií, ako je napríklad vytváranie SQL dotazu na základe užívateľom vybraných atribútov. Potrebné spojenie dvoch tabuliek je vykonávané iba cez tabuľku faktov a nie je potrebné zložito hľadať spôsob ich spojenia. Zníži sa počet prevedených spojení pri vykonávaní dotazu, s čím sa môže značne skrátiť čas spotrebovaný na poskytnutie odpovedi.

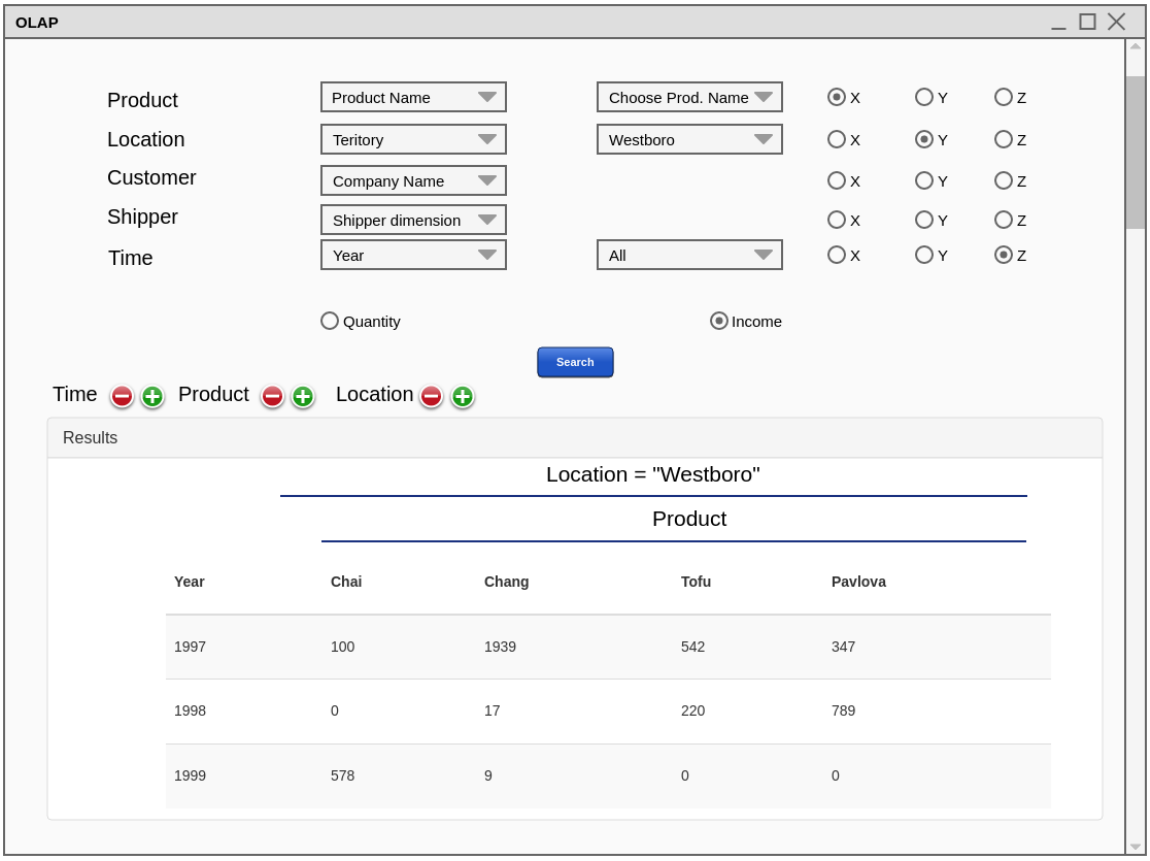
Užívateľské rozhranie

Veľmi dôležitou časťou aplikácie je užívateľské rozhranie, najmä jej intuitívnosť. Ak je správne navrhnuté vie znížiť čas vykonávania jednotlivých krokov požadovaných pre nastavenie informácií, potrebných pre fungovanie aplikácie. Ak sa užívateľ vie jednoducho zorientovať v rozhraní aplikácie, nestratí náladu a motiváciu pre vykonávanie pracovných povinností.

Pri návrhu jednotlivých webových stránok sa cielene zvolil minimalizmus. Obsahujú iba potrebné prvky pre vykonanie daných úloh, ako je formulár alebo tlačidlá. Vzhľadom na to, že stránky môžu byť zobrazované na zariadeniach s rozličnou veľkosťou zobrazovacej plochy je dobré ich vytvárať tak, aby boli responzívne. V rámci možností prispôsobovali svoj vzhľad k prítomným rozmerom, rozpoznávali na akom zariadení sa práve zobrazujú.

Dátové kocky, výsledky OLAP operácii budú zobrazované v podobe dvojdimenzionálnych tabuliek.

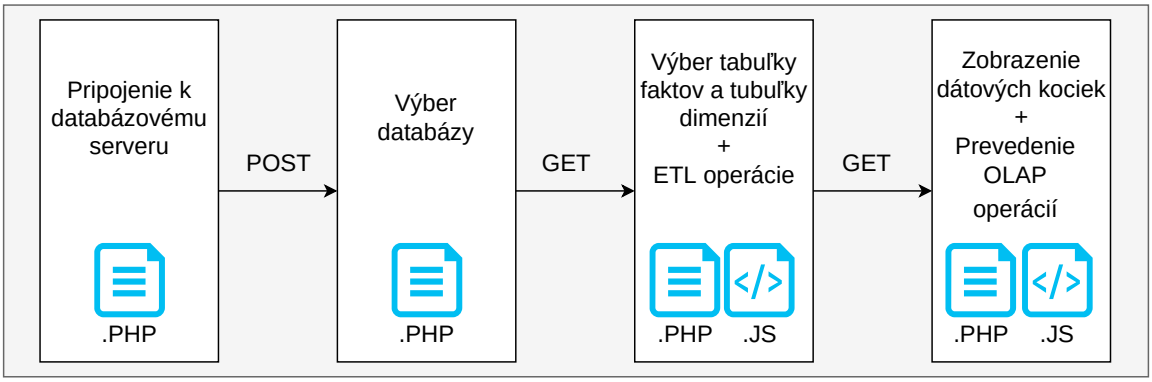
Na obrázku 4.2 je znázornený návrh rozloženia prvkov na stránke zodpovednej za výber atribútov dotazu a následného zobrazenia výsledkov.



Obr. 4.2: Mock-up plánovanej aplikácie.

4.4 Architektúra aplikácie

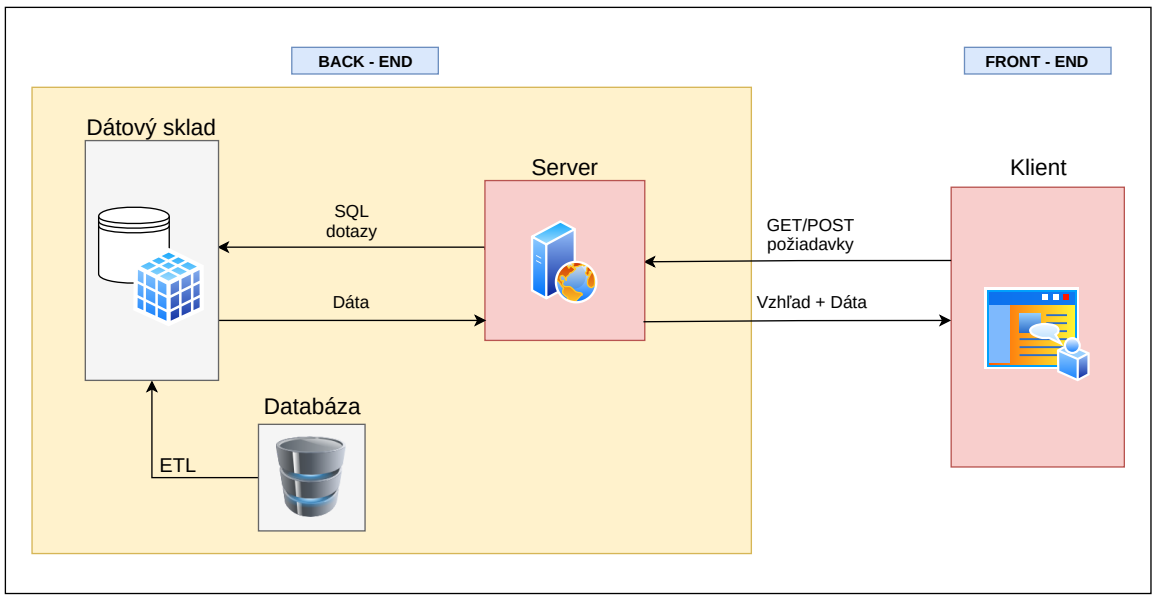
Jednotlivým funkciám aplikácie, spomenuté v sekcií 4.2, bude priradená osobitná webová stránka v rámci užívateľského rozhrania. Na obrázku 4.3 je možné vidieť postupnosť stránok a operácií. Typ požiadaviek medzi stránkami bude zvolený v závislosti prenášaných dát medzi nimi.



Obr. 4.3: Postupnosť webových stránok v rámci aplikácie. Vyznačený je typ funkcionality jednotlivých stránok a typ možných súborov, z ktorých sa budú tvoriť a tiež vyznačený je typ požiadaviek medzi stránkami.

Navrhnutá architektúra aplikácie (obrázok 4.4) nasleduje veľmi známy klient - server model (anglicky *Client - Server Architecture*). Klient posiela požiadavky na back - end aplikácie, ktorý pozostáva zo serveru, aplikácie a databázy [4]. Jeho častým predstaviteľom sú prehliadače, ktoré v odpovediach na požiadavky očakávajú kód stránky, ktorú budú zobrazovať užívateľovi alebo iné dáta. Server je počítač, ktorému sa posielajú dotazy. Na ňom beží aplikácia, ktorá prijíma prichádzajúce požiadavky, načíta podľa nich informácie z databázy a pošle ich späť ako odpoveď.

V úlohe aplikácie na serveri sa nachádza vyvíjaná aplikácia a ako klient vystupuje webový prehliadač. Aplikácia prijme a zapracuje požiadavky prichádzajúce od klienta a vhodným spôsobom ich prevedie na SQL dotazy. Tieto dotazy sa pošlú ďalej databázovému serveru a vďaka nim sa získajú požadované dáta. V závislosti na nich a iných pravidlách sa vygeneruje popis štruktúry a vzhľadu stránky vo vybranom programovacom jazyku. Vygenerovaný kód je predaný späť klientovi, ktorý sa o ňu požiadal.



Obr. 4.4: Architektúra aplikácie. V rámci projektu sa implementujú časti server a klient (vyznačené červene). Ostatné časti sú uvedené pre úplnosť, pre lepšie pochopenie fungovania aplikácie.

Kapitola 5

Implementácia aplikácie pre zobrazenie OLAP kociek

Koncová implementácia bola vytvorená na základe pripraveného návrhu (kapitola 4). Kapitola 5 obsahuje popis realizácie výsledného programu. Cieľom nie je úplnou detailnosťou popísať všetky funkcie a konštrukcie programu, iba vyzdvihnúť zaujímavé implementačné riešenia.

Sekcia 5.1 predstavuje vybrané technológie pre uskutočnenie návrhu a odôvodnenie pre ich používanie. Ďalej sa predstaví Model–View–Controller architektonický vzor a jeho rozhodujúci vplyv na konečnú štruktúru aplikácie, jej použitie.

S vytvorením funkcionalít očakávaných od výsledku práce sa zaoberá sekcia 5.2. Popisuje štruktúru jednotlivých stránok v rámci užívateľského rozhrania. Ich prepojenie medzi sebou a úlohu v zabezpečovaní potrebných interakcií s užívateľom.

Výsledky užívateľom prevedených OLAP akcií sa zobrazia na poslednej, takzvanej hlavnej stránke. Dátové kocky majú podobu klasických dvojdimenzionálnych tabuliek.

V závere kapitoly sú predstavované možné rozšírenia aplikácie do budúcna 5.3. Z mnohých vylepšení sú vybrané tie, ktoré by pri prevedení mohli obohatiť možnosti užívateľa.

5.1 Použité technológie

Nasledujúce technológie boli buď určené samotným zadaním, ako v prípade PostgreSQL, alebo vybrané ohľadom na zvolený typ aplikácie :

Laravel

Jeden z hlavných programovacích jazykov použitý pri implementácií je PHP. Bol zvolený pre tvorbu back-endu aplikácie. Aby vývoj a písanie kódu bolo rýchlejšie a jednoduchšie rozhodlo sa pre použitie aplikačného rámca (anglicky *framework*, ďalej sa bude používať anglický názov). Zo všetkých možností voľba padla na dnes rozsiahlo používaný *Laravel*. V rámci projektu sa používa verzia číslo šesť.

Laravel je open source PHP framework vyvinutý pre tvorbu webových aplikácií. Disponuje s elegantnou a výraznou syntaxou. Vo veľkom sa opiera o starší známy framework zvaný *Symfony*[1]. Filozofiou Laravelu, podľa jeho tvorcu, je zjednodušiť vývoj pomocou uľahčenia úloh, ktoré sa vyskytujú vo väčšine webových projektov [15] :

[1] https://symfony.com/

- Kontrola prístupu užívateľov
- Smerovanie a zapracovanie dotazov
- Komunikácia s databázou
- Správa relácií (anglicky *sessions*) a cookies

Kladie si za cieľ aby proces vývoja bavil vývojára a nedošlo pri tom k negatívnemu ovplyvneniu funkcionalít aplikácie.

Ďalšou veľmi kladnou vlastnosťou je precízne vypracovaná dokumentácia dostupná na oficiálnej stránke frameworku. Pre každú jednu implementovanú funkcionalitu je vytvorený prehľadný popis spolu s príkladom jeho použitia. Pre zabezpečenie spätnej kompatibility sú zachované texty pre predošlé verzie.

Ako pomôcka pre ešte rýchlejšie osvojenie základov frameworku slúži *Laracasts*[2], stránka obsahujúca sériu výukových videí.

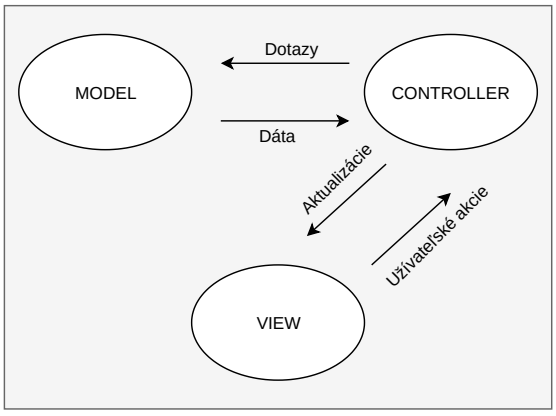
Počas uplynulých rokov sa vytvorila veľmi aktívna a ochotná komunita pomáhajúca v riešení implementačných problémov svojich členov.

MVC v laravel projekte

Laravel, nasleduje architektonický vzor MVC. Model-View-Controller, je softvérová architektúra najčastejšie známa pod skratkou *MVC*. MVC je označovaný aj ako návrhový vzor, ale v skutočnosti označenie ako architektonický vzor (anglicky *architectural pattern*) je oveľa vhodnejší, lebo má najväčší vplyv na samotnú architektúru aplikácie. Ovplyvňuje to, ako bude softvér rozdelený na jednotlivé moduly. Rozdeľuje zdrojový kód na väčšie logické celky.

Tradične bol používaný pre návrh grafického užívateľského rozhrania (skratkou GUI) u desktopových aplikácii, ale neskôr sa stal neoddeliteľnou súčasťou vytvárania webových aplikácií.

Programová logika je prevedená na tri vzájomne spolupracujúce prvky (obrázok 5.1). Vďaka tomu interná reprezentácia informácií je nezávislá na tom, akou formou sa dáta zobrazujú, respektíve získavajú od užívateľa.



Obr. 5.1: Architektonický vzor Model - View - Controller. Šípky vyznačujú smer a typ správ pri vzájomnej komunikácií vrstiev.

[2] https://laracasts.com/

Obecne MVC architektúra očakáva vytvorenie troch objektov v rámci programu: Model, View, Controller [3]. Tieto tri objekty sú od seba rozdelené iba abstraktnými hranicami. Každá aplikácia si môže v prevedenom zmysle nastaviť tieto hranice inde. Interná komunikácia medzi nimi tiež nie je bližšie stanovená v MVC. Predávanie správ medzi nimi závisí nie len na typu aplikácie, ale aj na použitom jazyku, prípadne aplikačnom rámci (anglicky *framework*).

Definícia troch objektov používaných v MVC architektúre [3] :

- Model - v sebe zapuzdruje dáta a definíciu funkcií, ktoré s nimi môžu manipulovať. Nie je závislí na vstupnom alebo výstupnom správaní sa systému. Objekty typu Model nie sú zobrazované užívateľovi. Príkladom môže byť modelový objekt *človek*, ktorý v sebe obsahuje dáta ako je *meno* a *priezvisko* a metódy pre ich získanie a nastavovanie.

- View (slovensky vzhľad) - časť používaná pre prevedenie dát, ktoré v sebe zapuzdrujú modely, do podoby vhodného pre interaktívnu reprezentáciu užívateľom. Vizualizácia môže byť prevedená napríklad grafom alebo tabuľkou. Pre dáta jedného objektu môže existovať viacero rôznych vzhľadov súčasne. Je to pravda aj pre celkovú aplikáciu, k jednej MVC architektúre môže existovať viac View prvkov. Ako reakcia na zmeny v modeli sa aktualizuje aj samotný vzhľad reprezentovaných dát.

- Controller (slovensky radič) - tento objekt tvorí prostredník medzi Model a View, sprostredkuje medzi nimi dáta oboma smermi. Je zodpovedný za aplikačne špecifické úlohy, ako je reakcia na udalosti a následné prevedenie zmien v modeli. Prijme vstupy od užívateľov, uskutoční nad nimi validáciu a pošle zodpovedajúce príkazy a informácie pre Model alebo View.

Veľkou výhodou MVC je, že umožňuje navrhnúť, implementovať a testovať každú jej časť individuálne a pri tom zdrojový kód zostáva prehľadný. Pridávanie nových funkcionalít je ľahko prevoditeľný. Vytvorené objekty sa dajú efektívne znovu použiť v rámci iných projektov.

Je vhodný tiež pre softvéry, na ktorých pracuje veľká skupina vývojárov. Obsahuje v sebe podporu pre rýchly a paralelný vývoj. Prácu je možné rozdeliť napríklad tak, že jedna skupina programátorov pracuje na vzhľade (front-end), druhá na modeli a radiči (back-end).

Výsledný program sa delí na základe MVC na tri hlavné časti (zdroj [14]):

- Model - táto vrstva v rámci frameworku je dedikovaná aplikačnej logike, práce s databázami a samotnými dátami. V rámci súboru */config/database.php* sa konfigurujú nastavenia jednotlivých databázových spojení. Je podporovaná väčšina databázových systémov ako napríklad MySQL, Postgres. Laravel v sebe obsahuje niekoľko zabudovaných nástrojov pre prácu s databázami[3] : Schema Builder, Query Builder, trieda DB a Eloquent ORM. Z uvedených možností sa pri implementácií použila trieda DB.

  Trieda DB v sebe obsahuje metódy ako je `select`, `insert`, `delete`, `update`. Tieto metódy umožňujú vykonanie príkazov napísaných priamo v jazyku SQL, anglicky takzvané *raw SQL queries*. Potrebné je mať k tomu dobre nastavenú konfiguráciu databázového spojenia. Metódy vytvorené SQL príkazy prevedú nad databázou, ktorá je stanovená vo vytvorenom spojení. Príklad pre `select` :

[3] https://laravel.com/docs/6.x/database

```
1 $results = DB::select('select * from users where id = :id', ['id' => 1]);
```

Dôvodom výberu tejto možnosti je, že užívateľ sa vie pripojiť pri spúšťaní aplikácie alebo aj dodatočne, vždy k iným databázovým serverom a vybrať si z prítomných databáz. Ostatné nástroje k fungovaniu vyžadujú prítomnosť modelov pre všetky používané tabuľky práve zvolenej databázy. Bolo by veľmi zložité vytvárať si dynamicky potrebné modely pri každej zmene databázového spojenia. Je oveľa jednoduchšie si potrebné informácie získať pomocou spomenutých metód triedy DB, uchovať si ich v premenných a potom podľa potrieb si z nich vygenerovať SQL dotazy a previesť ich nad databázou.

- Controller - vrstva v rámci frameworku, ktorá je zodpovedná pre zapracovanie prichádzajúcich požiadaviek (anglicky *requests*) napríklad typu GET a POST. Logiku pre prijímanie požiadaviek a následné prevedenie nutných krokov je vhodné organizovať do tried, rozširujúcich základnú triedu *Controller*. Kolektívne sa označujú ako radiči (anglicky *controllers*) a sú uchovávané v adresári *app/Http/Controllers*.

  Ku každému typu požiadavky na základe toho, pre ktorú adresu je vytvorená sa dá priradiť konkrétna metóda v radiči. Metóda sa zavolá vždy pri výskyte požiadavky.

  V rámci projektu je vytvorený radič *DBOperationsController.php*, ktorý obsluhuje všetky požiadavky posielané medzi jednotlivými stránkami aplikácie. Hlavnou úlohou metód v radiči je kontrola predávaných dát a komunikácia s databázou pomocou dotazov.

- View - vrstva, ktorá sa stará o zobrazenie výstupu užívateľovi. Oddeľuje radiče, aplikačnú logiku od prezentačnej logiky. Súbory definujúce vzhľad stránok, anglicky takzvané *Views*[4], obsahujú HTML a PHP kód. Ukladajú sa do adresáru *resources/views*.

  Webové frameworky často poskytujú nástroje pre prácu so šablónami. Cieľom je oddeliť HTML kód od programovacieho jazyka, ako je napríklad PHP. Pomocou šablón sa dá znížiť redundancia kódu a znovu použiť už vytvorené časti vzhľadu.

  Laravel ponúka šablónovací systém (anglicky *templating system*) zvaný *Blade*[5]. Oproti iným systémom umožňuje používať v šablónach aj čistý PHP kód vedľa vlastnej sady špeciálnych príkazov. V skutočnosti Blade pridáva nulovú réžiu k aplikácii vďaka tomu, že šablóny sa preložia do čistého PHP a sú ukladané do vyrovnávacej pamäti, kým nie sú modifikované. Súbory obsahujúce vzhľad majú v názve príponu *.blade.php* a sú ukladané do adresáru *resources/views*.

  Pri vytváraní stránok sa použili dve kladné vlastnosti Blade, ako je dedičnosť šablón a sekcie. Dedičnosť šablón znamená, že stačilo vytvoriť jednu hlavnú šablónu zvanú *app.blade.php*, ktorá slúži ako základ pre všetky stránky aplikácie. Ukladá sa typicky do adresáru *resources/views/layouts*. Obsahuje klasickú HTML kostru, importovanie CSS súborov a JavaScriptové funkcie.

  Ostatné vytvorené stránky dedia obsah od hlavnej stránky, ktorá sa určí pomocou príkazu *@extends*. Obsahujú v sebe sekcie označené príkazom *@section*, ktorých obsah sa pri spracovávaní súboru vloží do zdedenej šablóny. Miesto vloženia sa vyznačuje v hlavnom súbore príkazom *@yield*.

[4] https://laravel.com/docs/6.x/views#creating-views
[5] https://laravel.com/docs/6.x/blade

Vue.js

Neoddeliteľnou súčasťou dnešných webových aplikácií, ako aj tohto projektu je JavaScript. Bol použitý hlavne pri jednotlivých stránkach pre validáciu vstupov od užívateľov a pre prípadné zmeny obsahu stránok. Na vytvorenie zložitejších vecí, ako sú napríklad komponenty v tomto jazyku, je lepšie použiť sofistikovanejšie aplikačné rámce.

Laravel priamo obsahuje podporu pre *Vue.js*. Je to progresívny javascriptový framework pre tvorbu užívateľského rozhrania a jednostránkových aplikácií (anglicky *single-page applications*) [23]. Jednoducho sa dá adaptovať do projektov a integrovať s inými knižnicami.

Šablony

Vue.js pre tvorbu šablón vytvoril špeciálnu syntax založenú na HTML. Každá šablóna je platný HTML kód, ktorý vie analyzovať väčšina prehliadačov. Využíva takzvané v-direktívy. Sú vytvorené z HTML atribútov pridaním prefixu „v-“, napríklad *v-bind:class*. Po zmene hodnoty v-direktívy sa aktualizuje rozhranie stránky.

Obojsmerná väzba dát

Vue.js používa obojsmernú väzbu dát (anglicky *two-way data binding*), čo vo veľkom uľahčuje prácu programátora. Zjednodušuje synchronizáciu užívateľských vstupov s dátovým modelom aplikácie. Je dosiahnuteľná pomocou direktívy *v-bind*. Aktualizuje sa šablóna vždy, keď sa model zmení a aktualizuje sa dátový model vždy, keď sa šablóna zmení.

Komponenty

Štruktúra vytváraných stránok sa rozdeľuje na menšie jednotky, komponenty. Sú to opakovane použiteľné pomenované inštancie, podobné k widgetom. Najjednoduchšie je ich porovnať k objektom v objektovo orientovanom programovaní (skratkou OOP). Podobne ako objekty majú svoje vlastné atribúty, metódy a funkcie. Okrem toho ešte obsahujú HTML šablónu a štýlový predpis určujúci jeho vzhľad. Môžu byť do seba vnorené, vytvárajúce tak strom vnorených komponentov. Teoreticky sa dajú vytvoriť tak, že pridanie alebo odobranie jedného z nich neovplyvní ostatné komponenty.

V projekte sa používajú jedno súborové komponenty (anglicky *single file components*). Každá komponenta má vlastný súbor, ktorého meno obsahuje sufix *.vue*. Ukladajú sa do adresára *resources/js/components*. Výhodou je, že po zaregistrovaní sa dajú použiť aj v iných aplikáciách. Rozdelené sú tak, aby každá bola zodpovedná iba pre jednu konkrétnu funkcionalitu.

Axios

Vue.js sebe neobsahuje nástroje pre vytváranie HTTP požiadaviek, ale existuje mnoho knihovní, ktoré túto funkcionalitu vedia zabezpečiť. Axios[6] je jeden z nich, ktorý samotní vývojári odporúčajú používať. Je to malá jednoduchá knihovňa implementujúca REST (anglicky *Representational state transfer*) pre prístup k CRUD[7] operáciám. Podporuje súbežné vytváranie viacerých požiadaviek naraz [1].

[6] https://github.com/axios/axios

[7] Skratka pre Create, Read, Update, Delete (slovensky *Vytváranie, Čítanie, Aktualizácia, Mazanie*). Základné operácie nad záznamom.

Axios pri implementácii je využívaný na posielanie asynchrónnych požiadaviek s cieľom získať dáta z databázy. Požiadavky sa zapracujú radičom *DBOperationsController.php*, ktorý vhodným spôsobom získa dáta a pošle ich späť. Využíva sa napríklad pri získavaní výsledkov OLAP operácií. Výhodou asynchrónneho dotazu je, že sa neprevedie znovu načítanie celej stránky, stačí prekresliť iba určitú časť stránky, nezdržiava užívateľa.

PostgreSQL

PostgreSQL je objektovo-relačný systém správy databáz (anglicky *object-relational database management system*, skratkou ORDBMS) [21]. Systém je vyvíjaný skupinou PostgreSQL Global Development Group, zahrňujúca rôzne spoločnosti. Počas dlhoročného vývoja získala dobrú povesť spoľahlivosti, robustnosti a výkonu. Veľký dôraz kladie na rozšíriteľnosť a dodržiavanie SQL štandardov. V súčasnosti sa jedná o najvyspelejší a najsofistikovanejší voľne šíriteľný (open source) systém riadenia báz dát.

Ponúka mnoho moderných funkcií ako sú :

- Komplexné dotazy
- Triggery
- Aktualizovateľné pohľady
- Správa súbežnosti
- Referenčná integrita
- Uložené procedúry v mnohých programovacích jazykoch (Java, C, Python)
- Vstavaný procedurálny jazyk nazývaný PL/PgSQL

Užívatelia majú možnosť rozširovať a prispôsobovať si PostgreSQL pridávaním nových operátorov, agregovaných funkcií, dátových typov, zložením z už existujúcich typov.

PostgreSQL je multiplatformná DBMS bežiaca na všetkých hlavných operačných systémoch. Používa jazyk SQL pre prácu s údajmi ukladanými do tabuliek. Oproti jeho konkurencie sa vyznačuje jednoduchšou použiteľnosťou pri vytváraní aplikácií, ktoré požadujú dáta z databáz.

Funkcia CUBE

Zaujímavou funkciou z pohľadu OLAP dátových kociek je *CUBE*[8]. Je subklauzulou klauzuly *GROUP BY* a rozširuje ju o možnosť generovať viacero zoskupení. Poskytne všetky možné kombinácie prvkov danej množiny. Z pohľadu dátových kociek všetky dosiahnuteľné úrovne, výsledky operácie Roll-up. Napríklad výsledok klauzule CUBE(b1, b2, b3) rovná sa zoskupení skupín GROUPING SETS((b1, b2, b3), (b1, b2), (b1, b3),(b1), (b2, b3), (b2), (b3), ()).

Od užívateľom vybraných dátových skladov sa očakáva, že budú obsahovať databázy typu PostgreSQL. Preto všetky dotazy sa v rámci aplikácie generujú s použitím syntaxe používaným PostrgeSQL-om. V rámci aplikácie sa ďalej očakáva existencia databázových šablón s názvom *template0* a *template1*[9]. V praxi sa používajú na vytvorenie nových databáz. Pri

[8] https://www.postgresqltutorial.com/postgresql-cube/
[9] https://www.postgresql.org/docs/9.5/manage-ag-templatedbs.html

volaní funkcie *CREATE TABLE* sa vytvorí kópia jednej z nich. Pre získanie potrebných informácií o zvolenej databáze sa použije informačná schéma (*infotmation_schema*[10]). Pozostáva z pohľadov obsahujúcich relevantné informácie o objektoch prítomných v databáze.

Font Awesome a Bootstrap

Prvý dojem užívateľa z užívateľského rozhrania značne ovplyvní jeho názor a postoj k celej aplikácií. Preto je potrebné venovať čas a úsilie k jeho vytváraniu. Pre uľahčenie implementácie vzhľadu stránok boli použité nasledujúce nástroje :

Font Awesome

Font Awesome[11] je jeden z najpopulárnejších toolkitov ponúkajúci zbierku vektorových ikoniek. Ikony zobrazujú od najbežnejších každodenne používaných položiek až logá veľkých firiem a značiek. Používajú sa spolu z inline prvkami ako sú *<i>* a **. Príklad použitia ikony auta:

```
1 <i class="fa fa-car fa-lg" ></i>
```

Bootstrap

Laravel obsahuje natívnu podporu pre jednu z najpoužívanejších knižníc s front-end komponentmi, framework Bootstrap[12]. Časť frameworku je dedikovaná pre kaskádové štýly (anglicky *Cascading Style Sheets*, skratkou CSS). Ponúka vopred pripravené triedy použiteľné spolu s HTML prvkami. Tieto triedy sú navrhnuté tak, aby podporovali responzívnosť. Bootstrap sa vyznačuje s tým, že umožňuje vytvárať *mobile-first* stránky. Znamená to, že stránky sú najprv navrhnuté pre mobilné telefóny alebo tablety a až potom pre klasické zobrazenie na počítači. V minulosti to bolo práve naopak. Dôvodom je, že v dnešnej dobe veľa užívateľov pristupuje k stránkam firiem cez menšie zariadenia.

5.2 Implementácia funkčných požiadaviek

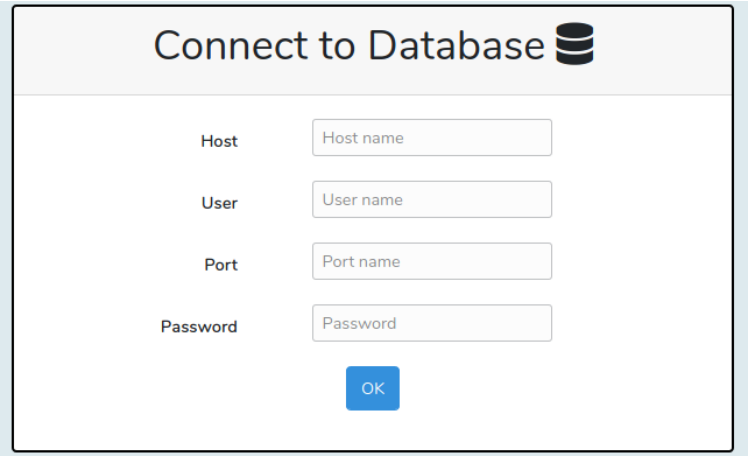
V tejto sekcií budú znázornené jednotlivé stránky aplikácie spoločne s implementáciou jednotlivých funkcionalít. Užívateľské rozhranie je rozdelené tak, aby každá stránka mala na starosti iba určitý počet funkčných požiadaviek. Popisy obsahujú aj ukážku relevantných častí vzhľadu stránok.

Správne vyplnenie vstupných polí, povinný výber možností alebo označenie zaškrtávacích políčok je kontrolované na strane klienta v rámci blade.php súborov. Pre kontrolu sa používajú jednoduché JavaScriptové funkcie. Validácia je prevedená i v radiči DBOperationsController.php pre prípad, keby užívateľ modifikoval kód stránok a tak odstránil povinný charakter položiek. V prípade vyskytnutých chýb užívateľ je informovaný pomocou alertu.

[10] https://www.postgresql.org/docs/9.1/information-schema.html
[11] https://fontawesome.com/
[12] https://getbootstrap.com/

Pripojenie k databázovému serveru

Pre pripojenie k databázovému serveru musí užívateľ zadať potrebné informácie, ako je napríklad adresa serveru, číslo portu, prihlasovacie údaje. Poskytnuté informácie sa skontrolujú podľa spomenutého postupu a uložia sa v podobe premenných do relácie (anglicky *session*).



Obr. 5.2: Formulár pre vyplnenie údajov potrebných pri vytváraní spojenia s databázovým serverom. Vstupné pole: *Host* - adresa serveru, *User* - prihlasovacie meno, *Port* - číslo portu, *Password* - heslo

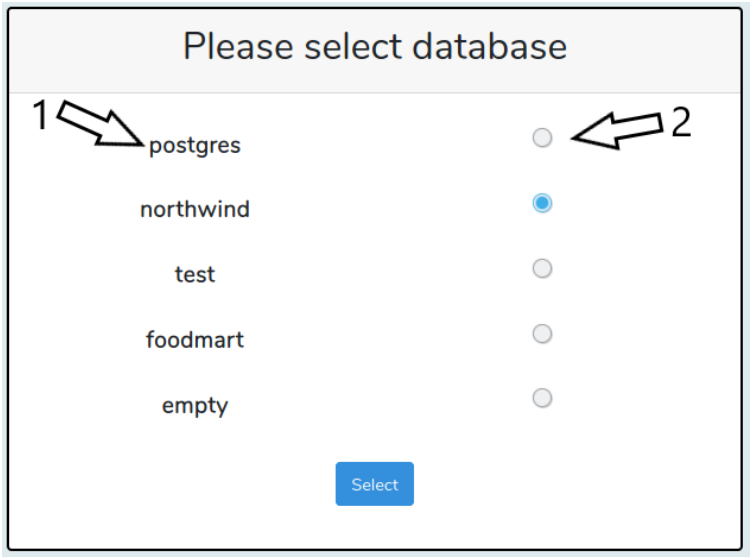
Dôvodom ukladania týchto údajov takouto formou spočíva v tom, že Laravel aplikácie predpokladajú ich prítomnosť v konfiguračnom súbore *.env* už pri spúšťaní aplikácie. Dáta z tohto súboru by boli použité pri spojovaní vždy s tým istým serverom a nedajú sa meniť počas behu. Oproti tomu aplikácia sa spojuje vždy dynamicky s práve vybraným serverom. Pre vytvorenie spojenia pred každou databázovou operáciou sa zavolá konštruktor radiča a použije k tomu dáta uchovávané v relácií. Pre zistenie dostupnosti serveru a správnosti prihlasovacích údajov je prevedené pokusné vytvorenie spojenia a získanie názvu serveru. Ak sa spojenie nevytvorí užívateľ je informovaný o tom a požiadaný o kontrolu poskytnutých údajov.

Výber databázy pre analýzu

Po úspešnom vytvorení spojenia z databázovým serverom sa užívateľovi zobrazia všetky dostupné databázy v rámci servera (obrázok 5.3). Jeho úlohou je vybrať jednu databázu označením jej prepínacieho tlačidla (anglicky *radio button*, ďalej bude používaný tento názov). Všetky nasledujúce databázové operácie budú prevedené nad touto databázou a bude považovaná za dátový sklad pri OLAP operáciách. Označenie databázy sa tiež uloží do relácie.

Meno všetkých prítomných databáz sa získa pomocou čistého SQL dotazu 1 zo základného katalógu PostgreSQL zvaný *pg_database*. Obsahuje všetky systémové informácie o nich.

```
1 $data_bases = DB::select("SELECT datname FROM pg_database WHERE
    datname NOT LIKE 'template%'");
```



Obr. 5.3: Formulár pre výber jednej databázy, ktorý bude ďalej používaný počas analýzy. Legenda: 1 - mená databázy prítomných na databázovom servery, 2 - políčka pre označenie jednotlivých databáz

Časť dotazu *„NOT LIKE 'template%'“* je použitý, aby neboli zahrnuté v databáze *tamplate0* a *tamplate1* spomenuté v popise PostgreSQL 5.1.

Výber tabuliek databázy

Stránka pre výber tabuľky faktov a tabuľky dimenzií pozostáva z viacerých častí. Základ tvorí súbor *ChooseTables.blade.php*, do ktorého je umiestnená komponenta *ChooseTables.vue* s ďalšou vnorenou komponentov *SetColumns.vue*.

Jedinou funkcionalitou základného súboru je zabezpečenie dedenia schémy od hlavného súboru *app.blade.php*.

Komponenta *ChooseTables.blade.php* je tvorená formulárom zabezpečujúcim možnosť stanovenia, ktoré tabuľky budú zahrnuté do analýzy a akú budú mať úlohu (obrázok 5.4). V rámci formuláru sú umiestnené postupne pod sebou v riadkoch mená všetkých relevantných tabuliek, dostupných v danej databáze. Každý riadok okrem mena obsahuje v poradí radio button, jedno zaškrtávacie políčko a jedno základné tlačidlo.

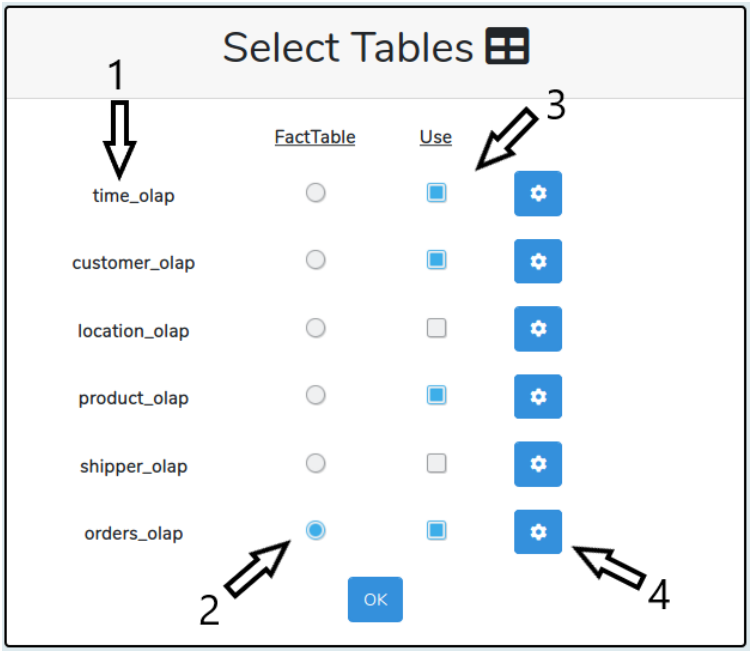
Z nadpisu *FactTable* nad stĺpcom radio buttonov, vyplýva, že tlačidlá slúžia na označenie tabuľky, ktorá bude považovaná za tabuľku faktov v schéme hviezdy. Radio button sa použil pre účel obmedzenia výberu iba jednej tabuľky. Po prekliknutí na tlačidlo inej tabuľky, tá sa automaticky označí ako vybratá a pôvodná sa vráti do neoznačeného stavu.

Stĺpec zaškrtávacích políčok s označením *Use* slúži na výber tabuliek, ktoré budú zahrnuté do analýzy v úlohe tabuľky dimenzie. Naraz môže byť z nich zaškrtnutý neobmedzený počet políčok. Minimálne je potrebné, aby boli označené tri, pre zabezpečenie správneho chodu OLAP operácií a zobrazovanie výsledných tabuliek reprezentujúcich kocky.

ETL operácie

Pre ukážku fungovania ETL operácií sa ponúkajú užívateľovi nasledujúce možnosti :

- Výber stĺpcov tabuľky, ktoré budú použité pri analýze dát



Obr. 5.4: Formulár pre výber jednej tabuľky faktov a aspoň troch tabuliek dimenzií. Legenda: 1 - mená tabuliek prítomných v databáze, 2 - políčka pre označenie tabuľky faktov, 3 - zaškrtávacie pole pre označenie tabuliek dimenzií, 4 - tlačidlo pre zobrazenie komponenty s ETL operáciami

- Nahradenie chýbajúcich číselných hodnôt v stĺpcoch faktov
- Odstránenie riadkov z tabuľky faktov s chýbajúcimi hodnotami

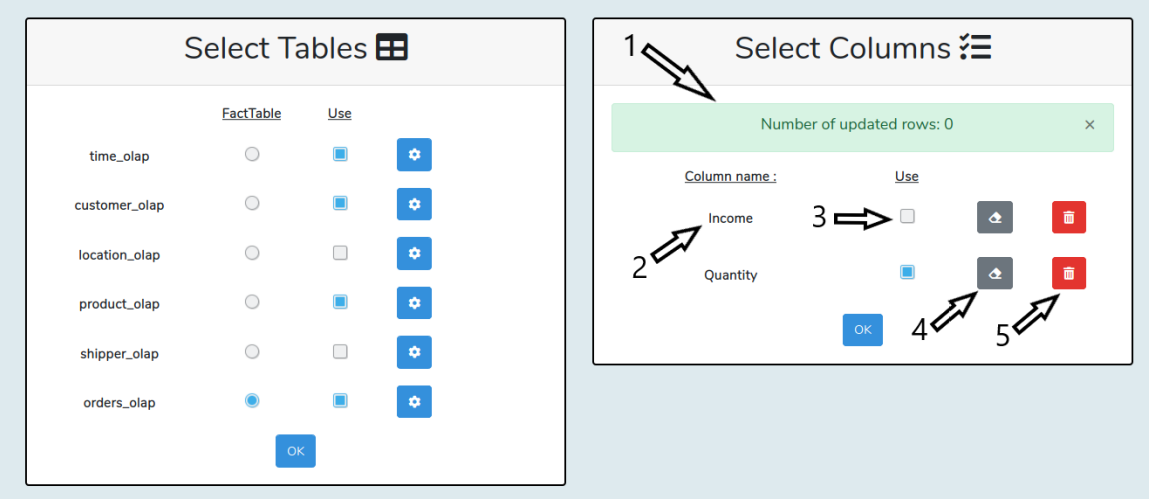
Reálne aplikácie ponúkajú oveľa zložitejšie a sofistikovanejšie operácie a postupy. Tieto operácie slúžia iba ako príklad pre splnenie zadania.

Prevedenie ETL operácií je dostupné cez vnorenú komponentu *SetColumns.vue*. Zobrazenie komponenty je možné dosiahnuť použitím tlačidla na konci riadkov formuláru. V závislosti na tom, na ktorom riadku bola stlačená sa mení obsah formulára, ktorý tvorí komponentu (obrázok 5.5).

Formulár, podobne ako v prípade výberu tabuliek, tvoria riadky obsahujúce mená stĺpcov, zaškrtávacie políčko. Uvedené sú iba mená tých stĺpcov, ktorých hodnoty v danej tabuľke neslúžia ako primárny alebo cudzí kľúč. Označenie políčka znamená že daný stĺpec tabuľky má byť zahrnutý do analýzy. Tie, ktoré ostanú neoznačené sa nepoužijú. Implicitne sa počíta s tým, že všetky stĺpce tabuľky dimenzie budú zahrnuté bez nutnosti ich označenia.

Prípadne, ak tlačidlo bolo použité pre tabuľku faktov, tak v riadku sa zobrazia ešte dve tlačidlá. Prvá z nich so sivou farbou slúži pre prevedenie operácie, ktorá nahradí všetky chýbajúce hodnoty v stĺpci daného faktu novou hodnotou. Ako doplnková hodnota bol zvolený priemer spočítaný zo všetkých zadaných hodnôt v danom stĺpci.

Druhé tlačidlo s červenou farbou slúži na prevedenie operácie, ktorá vymaže všetky riadky z tabuľky faktov, v ktorom chýbajú hodnoty v stĺpci daného faktu. Užívateľ je upozornený pomocou modálneho okna, že po prevedení operácie sa nedajú získať späť vymazané hodnoty.



Obr. 5.5: Formulár na pravej strane obrázku slúži pre explicitný výber stĺpcov tabuľky, ktoré budú dostupné pre výber pri analýze. Legenda: 1 - informácia o počte dotknutých riadkov po prevedení zvolenej operácie, 2 - mená stĺpcov, 3 - zaškrtávacie pole pre explicitné označenie stĺpca, 4 - tlačidlo pre nahradenie chýbajúcich hodnôt priemerom, 5 - tlačidlo pre odstránenie riadkov s chýbajúcimi hodnotami

Po prevedení nahradenia chýbajúcich hodnôt alebo vymazania riadkov je užívateľ informovaný o počte dotknutých riadkov. Stlačením tlačidla *OK* sa uchovajú voľby a komponenta sa skryje.
Pred odoslaním formuláru v *ChooseTables.blade.php* sa skontroluje, či bola stanovená tabuľka faktov a minimálne tri tabuľky dimenzií. Ak sú splnené všetky požiadavky je možné prejsť na poslednú a zároveň hlavnú stránku aplikácie.

Zobrazenie dátových kociek a OLAP operácie

Hlavná stránka aplikácie slúži pre zobrazenie najdôležitejších funkcionalít celého programu, uskutočnenia OLAP operácií a zobrazenie výsledných dátových kociek.

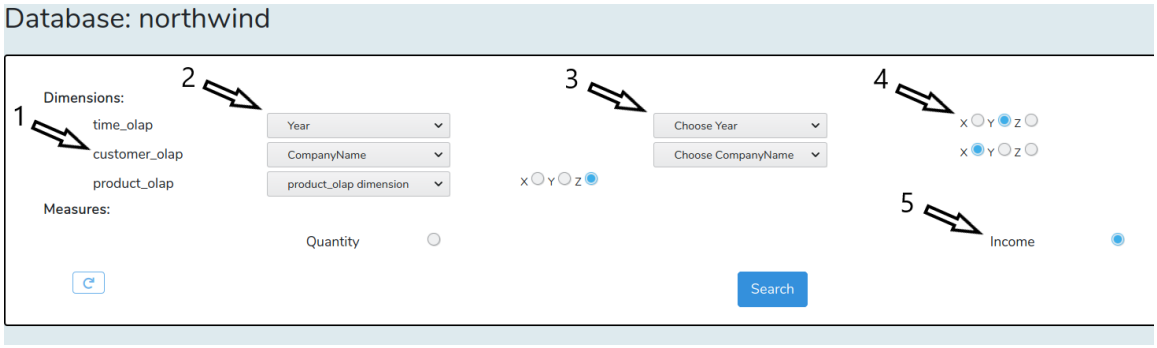
Stránka pozostáva zo súboru *SearchFormPage.php* a komponenty *SearchForm.vue*. Táto hlavná komponenta ďalej obsahuje dve vnorené komponenty *Buttons.vue* a *Tables.vue*.

Základný súbor zabezpečuje dedenie obsahu *app.blade.php* a predávanie parametrov vo formáte *JSON* hlavnej komponente.

Komponenta *SearchForm.vue* obsahuje v sebe formulár umožňujúci výber potrebných atribútov pre prevedenie OLAP operácií (obrázok 5.6). Každá tabuľka dimenzie sa reprezentuje v rámci formuláru jedným riadkom.

Riadky obsahujú meno dimenzie, dve výberové polia (anglicky *select*) a tri radio button. Výberové možnosti polí sú získavané z databázy zvlášť, pre všetky tabuľky ešte pred vykreslením samotnej stránky.

Prvé pole slúži pre výber stĺpca tabuľky dimenzie. V uvedenom zmysle, možnosti predstavujú jednotlivé úrovne v konceptuálnej hierarchii dimenzie. Buď sú vymenované všetky úrovne alebo iba tie, ktoré užívateľ explicitne vybral na predchádzajúcej stránke pri výbere tabuliek.



Obr. 5.6: Formulár pre výber atribútov pre prevedenie OLAP operácií. Legenda: 1 - meno dimenzie, 2 - meno úrovne v konceptuálnej hierarchií dimenzie, 3 - konkrétna hodnota v úrovni, 4 - výber osí, 5 - výber faktov

Po výbere stĺpca sa zobrazí druhý výber, ktorý obsahuje ako možnosti všetky hodnoty vyskytujúce v danom stĺpci a špeciálnu hodnotu *ALL*, ktorá znamená že celý stĺpec sa použije bez ohľadu na hodnoty.

Stanovením iba úrovne v rámci konceptuálnej hierarchie dimenzie, prevedie v symbolickom zmysle OLAP operácia *Slice*. Ak sa vyberie aj konkrétna hodnota v druhom výber tak sa už jedná o operáciu *Dice*.

Posledné tri radio button, označené ako *X*, *Y*, *Z* slúžia pre výber osi, na ktorej bude vybraná dimenzia zobrazená vo výslednej dátovej kocke. Zmenou osi vybraných dimenzií sa dá previesť OLAP operácia *Pivot*. Tak je možné dáta zobrazovať z iných perspektív a dosiahnuť prehľadnejšie zobrazenie výsledných tabuliek. Je to užitočné najmä v prípade širokých tabuliek.

Na konci formuláru sú uvedené mená faktov obsadené v tabuľke faktov. Vybrať z nich je možné označením jedného z radio button.

V prípade ak by sa užívateľ pomýlil pri výbere atribútov, tak má možnosť formulár uviesť do pôvodného stavu. Túto funkcionalitu prezentuje tlačidlo *Reset* v ľavom dolnom rohu formulára.

Prevedenie SQL dotazov

Po potvrdení vybraných atribútov je prevedená kontrola splnenia požiadaviek. Medzi ne patria výber jedného faktu a troch tabuliek dimenzií, výber stĺpcov tabuliek a rozličných osí pre dimenzie. Ak kontrola neprebehne úspešne, tak prípadné chyby sú oznámené užívateľovi, ktorý ma možnosť ich opraviť.

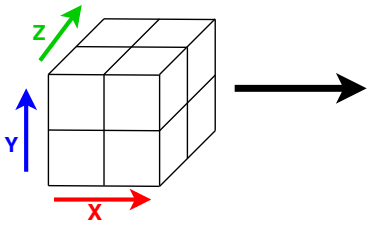
Výsledok SQL dotazu, vytvoreného na základe vybraných atribútov, sa získa asynchrónne od radiča. Najprv sú asynchrónnou požiadavkou predané atribúty v radiči. Ten na základe nich vytvorí dotaz obsahujúci klauzuly *select*, *from*, *where*, *group by* a *order by*. Dotaz sa prevedie nad databázou a jeho výsledok pošle späť radič ako odpoveď na pôvodnú požiadavku.

V závislosti od výsledku sa modifikuje obsah stránky. Ak dotaz bol neúspešný, z dôvodu že pre danú kombináciu dimenzií a atribútov sa nenašli dáta, tak užívateľ dostane oznámenie o neúspechu. V druhom prípade, ak sa našli požadované dáta, zobrazia sa vnorené komponenty *Buttons.vue* a *Tables.vue*.

Komponenta *Buttons.vue* je zodpovedná pre prevedenie ďalších OLAP operácií (obrázok 5.7). Obsahuje pre každú dimenziu dve tlačidlá. Tlačidlo označené znakom „+“ slúži pre operáciu *Drill-down* a tlačidlo označená znakom „-“ slúži pre *Roll-up*. Po stlačení niektorého z nich sa skontroluje či operácia je prevediteľná. Prevediteľnosť znamená, že sa dá posunúť o úroveň vyššie, respektívne nižšie v konceptuálnej hierarchií dimenzie. Pri kladnom vyhodnotení podmienky sa prevedie znovu dotaz, ale už s pozmenenými atribútmi.

Obr. 5.7: Tlačidlá pre prevedenie operácií Roll-up a Drill-down. Legenda: 1 - tlačidlo pre Roll-up, 2 - tlačidlo pre Drill-down

O premapovaní výsledných dátových kociek na dvojdimenzionálne tabuľky sa stará komponenta *Tables.vue*. Princíp premapovania osí kociek na jednotlivé časti zobrazenej tabuľky je znázornené na obrázku 5.8.



CompanyName = Alfreds Futterkiste			
	Year		
Category	1996	1997	1998
Beverages	0	21	0
Condiments	0	20	16
Confections	0	0	0
Dairy Products	0	0	20
Grains/Cereals	0	0	0
Meat/Poultry	0	0	0
Produce	0	15	2
Seafood	0	2	40

Obr. 5.8: Premapovanie dátových kociek na dvojdimenzionálne tabuľky. Farebnými čiarami sú vyznačené osi kocky a umiestnenie ich hodnôt v rámci tabuľky.

Zobrazenie všetkých dát v jednej tabuľke by bolo veľmi neprehľadné. Z toho hľadiska, že nie všetky zariadenia majú dostatočne veľkú zobrazovaciu plochu. Preto sa rozhodlo dáta rozdeliť do viacerých tabuliek, podľa dimenzie na osi *X*. Počet tabuliek zodpovedá počtu hodnôt tejto dimenzie.

5.3 Možné rozšírenia aplikácie

Aplikácia splňuje požiadavky jednoduchého užívateľa, ktorý by chcel skúmať menšie množstvo dát vo vopred pripravených dátových skladoch. Napriek tomu obsahuje stránky, ktoré by bolo potrebné zlepšiť alebo rozšíriť aby sa jednalo o plnohodnotnú aplikáciu použiteľnú v profesionálnom prostredí.

Príkladom môže slúžiť implementácia nástrojov pre vytváranie nových dátových skladov z užívateľom vybraných databáz. Treba podporovať aj schémy snehová vločka alebo súhvezdie, okrem základnej vločky, kde by sa zmenil aj spôsob vytvárania databázových dotazov. Jednoduchá a časovo nie príliš náročná etapa ETL, s použitím zaujímavých algoritmov, by vedela zaujať nie jedného odborníka.

Tieto úpravy by posunuli aplikáciu o úroveň vyššie a rozšírili by skupinu potencionálnych užívateľov.

Kapitola 6

Testovanie

Nasledujúca sekcia popisuje spôsob odhaľovania chýb a kontrolu realizácie funkčných požiadaviek počas implementácie výslednej aplikácie.

Testovanie tvorí dôležitú súčasť vývoja každej jednej aplikácie. Je dobrým zvykom testovanie prevádzať už od samotného začiatku práce na výslednom produkte. Celý cyklus vývoja pozostáva z viacerých častí, z ktorých sú niektoré venované práve tejto činnosti ako napríklad návrh, plánovanie, vykonávanie testov. Ich kladný vplyv sa odráža v úspore nákladov, šetrenia času spotrebovanej na opravu chýb a zvýšenia kvality výsledku implementácie.

Po dobu trvania vývoja bolo testovanie vykonávané manuálne. Využíval sa pri tom webový prehliadač, server a univerzálny nástroj pre správu databáz. Boli postupne vyskúšané jednotlivé kroky, ktoré by potreboval previesť bežný užívateľ pri používaní aplikácie. Počas testovania sa skúšali aj scenáre, ako napríklad pripojovanie k neexistujúcemu databázovému serveru, nesplnenie požadovaných podmienok u jednotlivých formulároch. Tieto kroky by mohli vyvolať neočakávané chovanie alebo chyby. Bolo potrebné skontrolovať správnosť reakcie systému na testovaných scenároch.

Databázové skripty

Správnosť prevedenia implementácie je najlepšie testovať v podmienkach pripomínajúcich reálne nasadenie v podnikoch. Preto bolo rozhodnuté využiť voľne dostupné databázy napodobňujúce reálne skladište dát firiem:

Databáza Northwind

Northwind Traders je fiktívna spoločnosť, ktorá sa zaoberá importom a exportom špeciálnych potravín. Informácie týkajúce sa predajov firmy sú verejne dostupné v databáze Northwind[1]. Obsahuje klasické tabuľky využívané u OLTP.

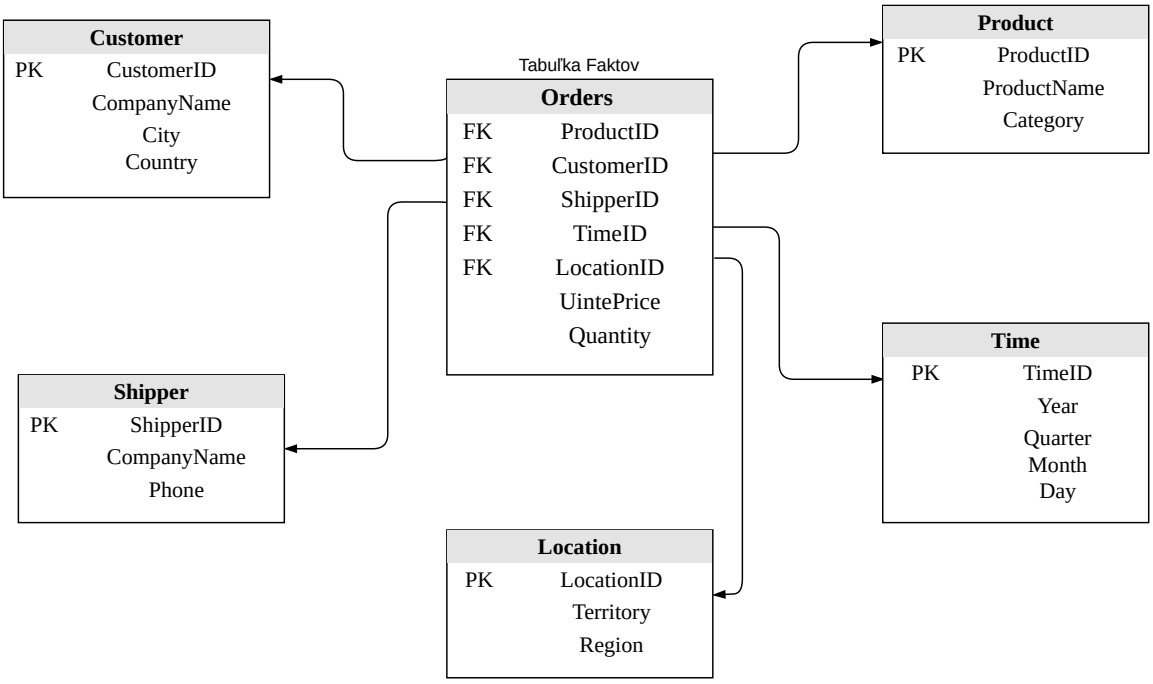
Databáza Foodmart

Databáza FoodMart[2] obsahuje veľké množstvo dát o predajoch fiktívnej spoločnosti obchodujúcej s potravinami. Okrem klasických tabuliek napríklad pre potraviny alebo zamestnancov obsahuje predom vytvorené agregácie údajov. Agregácie sú vytvorené z dát týkajúcich

[1] https://relational.fit.cvut.cz/dataset/Northwind
[2] https://github.com/OSBI/foodmart-data

sa predaja v jednotlivých rokoch, ale neboli použité pri testovaní.

Pre účely jednoduchšieho testovania aplikácie, využitím dát z týchto dvoch databáz, boli vytvorené vlastné databázové skripty. Prvý typ skriptov predpokladá spúšťanie v rámci databázy, ktorá bude prevedená na dátový sklad. Vytvára nové tabuľky a naplňuje ich postupne s dátami, získanými pomocou výberov z pôvodných tabuliek. Oproti tomu druhý typ už obsahuje v sebe aj dáta a s využitím funkcie insert ich vkladá do novo vytvorených tabuliek. Zjednodušuje znovu vytváranie skladu po testovaní ETL operácií. Po preskúmaní schém databáz a obsahu jednotlivých tabuliek boli vybrané tie, ktoré sú z hľadiska vytvorenia dátových skladov najviac využiteľné. Na obrázku 6.1 je zobrazený ako príklad schéma skladu, vytvorená z databázy Northwind. Oba tieto dátové sklady sú založené na schéme hviezdy.



Obr. 6.1: Schéma v tvare hviezdy vytvorená z pôvodnej databázy Northwind. K vytvoreniu databázového skladu sa použili nové tabuľky s atribútmi z originálnych tabuliek. Schéma obsahuje jednu tabuľku faktov a päť tabuliek dimenzií.

Spravovanie a nasadenie databáz vyžadovalo použitie nástroja vytvoreného pre tento účel. Dbeaver[3] sa ukázal ako správna voľba. Podporuje všetky známe typy databáz a medzi nimi práve aj PostgreSQL, ktorá je z hľadiska projektu nevyhnutná. Možnosť kontroly a prevedenia SQL skriptov uľahčila ich vytváranie. Aplikáciou konštruované dotazy a výsledky OLAP operácií sa dali jednoducho skontrolovať prevedením tých istých dotazov v nástroji a vzájomným porovnaním získaných výsledkov.

Každý webový prehliadač zobrazí obsah stránok odlišným spôsobom. Rozdiely môžu byť nie len v vzhľade jednotlivých prvkov ale aj v podpore zabudovaných nástrojov, javascriptových

[3] https://dbeaver.io/

funkcionalít. Preto v úlohe klienta bolo vyskúšaných niekoľko prehliadačov ako napríklad Opera[4], Google Chrome[5] a Mozilla Firefox[6].

Všetky tieto prehliadače v sebe obsahujú zabudované vývojárske nástroje (anglicky *browser developer tools*), ktoré umožňujú preskúmať kód, z ktorého pozostávajú webové stránky a zobraziť hlásenia o prípadne vyskytnutých chybách. Využívalo sa najmä pre ladenie kaskádových štýlov aplikovaných na jednotlivé prvky. Ďalšou často používanou funkcionalitou bolo zobrazenie dočasných výstupov javascriptových funkcií a kontrola posielaných paketov medzi klientom a serverom.

Servery zabezpečujúce prostredie pre beh webových aplikácií sú vytvárané na lokálnom počítači pomocou aplikácií ako je *Apache* alebo *Nginx*. V rámci projektu bol využitý pre tento účel vstavaný vývojový server PHP. Laravel má v sebe zabudované nástroje pre spúšťanie a správu tohto HTTP serveru.

Testovanie chybových stavov

Počas testovania chybových stavov, do ktorých by sa aplikácia mohla dostať počas jej nasadenia v reálnom prevoze, sa vyskúšalo niekoľko možných scenárov. Vďaka nich sa podarilo odhaliť a neskôr odstrániť nedokonalosti z programu.

Formulár prítomný na úvodnej stránke aplikácie ponúka hneď niekoľko možností, v ktorých by užívateľ mohol urobiť chybu. Najprv bola preverená správna kontrola vyplnenia všetkých vstupných polí jednak na strane klienta, ale aj na strane serveru. V prípade formulárov na tomto i na všetkých ostatných stránkach sa vstupy kontrolujú aj na servery s cieľom, aby sa zamedzilo pokusom obísť povinnosť ich vyplnenia jednoduchou modifikáciou HTML kódu stránok a odstránením ich atribútu *required* (slovensky *povinný*).

Práve pri tejto kontrole po niekoľkých zlých vyplnení hesla z nepozornosti sa vytvorila myšlienka pridania tlačidla, ktorého použitie spôsobí prevedenie hesla na podobu čitateľného textu. Jeho opätovné použitie znova skryje heslo.

Ako ďalšia možnosť je skúsiť pripojiť sa buď k neexistujúcej databáze alebo k takej, ktorá nie je typu PostgreSQL. Pre účely pokusného pripojenia bola vytvorená lokálna MySQL databáza.

Pri výbere databázy pre samotnú analýzu z uvedených možností sa testovalo to, ako program zareaguje na výber možnosti, u ktorej sa po kontrole zistí, že okrem systémových tabuliek žiadna iná neexistuje. V prípade nulového počtu použiteľných tabuliek je vyzvaný užívateľ ku kontrole použitých nastavení alebo výberu inej vhodnejšej databázy.

Častou situáciou pri skúšobnom používaní bolo, že sa zabudlo na označenie niektorých z potrebných výberových polí. Preto chybové hlásenia boli doplnené o presnejšie informácie odôvodňujúce výskyt chyby a pokyny pre správne vyplnenie formuláru.

[4] https://www.opera.com/
[5] https://www.google.com/chrome/
[6] https://www.mozilla.org/sk/firefox/new/

Kapitola 7

Záver

V posledných rokoch sa výrazne zvýšila miera využívania historických dát. Spoločnosti cielene ich uskladňujú v nadmerných množstvách a zámerne prevádzajú nad nimi analýzu. Skúmaním týchto dát získajú údaje zľahčujúce rozhodovanie v strategických otázkach a umožňujú zvýšiť kvalitu ponúkaných služieb.

V rámci teoretickej časti bakalárskej práce boli prezentované základné prvky patriace k vykonaniu analýzy. Z hľadiska problematiky uskladnenia dát sú popisované dátové sklady. Pre jednoduchšie predstavenie ich fungovania boli porovnávané s obecne známejšími relačnými databázami. Pre naplňovanie skladov s dátami sú nezanedbateľné nástroje zabezpečujúce prípravu dát. Najdôležitejšie procesy predspracovania boli popísané prostredníctvom ETL, označovanej aj ako dátová pumpa. Hlavné časti ETL tvoria extrakcia, transformácia a nahrávanie dát.

OLAP analýza slúži pre získavanie informácií z uložených dát. Multidimenzionálny model vytvára špeciálny pohľad na dáta v podobe viacrozmerných dátových kociek. Z prevoditeľných OLAP operácií nad kockami boli uvedené Roll-up, Drill-down, Slice a Dice. Vykonaním týchto operácií sa získajú dáta z kombinácie rôznych úrovní konceptuálnych hierarchií zvolených dimenzií.

Nadobudnuté teoretické vedomosti boli využité k návrhu aplikácie zobrazujúce dátové kocky v podobe dvojdimenzionálnych tabuliek. Funkčné požiadavky boli vytvorené na základe zadania a potrieb budúcich potencionálnych užívateľov. Technológie použité pri implementácií boli zvolené zámerne pre uľahčenie procesu vývoja webovej aplikácie.

Výsledná aplikácia funguje podľa očakávania vyplývajúceho zo zadania. Testovanie prebiehalo pomocou datasetov strednej veľkosti, napodobňujúce nasadenie v menšej organizácii. Napriek spoľahlivému fungovaniu niektoré stránky by bolo potrebné rozšíriť, aby sa jednalo o plnohodnotnú aplikáciu. Príkladom by mohla byť možnosť tvorby dátových skladov s užívateľom stanovených databáz. Tieto úpravy by dostali aplikáciu na profesionálnu úroveň.

Literatúra

[1] Bemenderfer, J. *Vue.js REST API Consumption with Axios* [online]. 2017 [cit. 2020-03-21]. Dostupné z: https://alligator.io/vuejs/rest-api-axios/.

[2] Bouaziz, S., Nabli, A. a Gargouri, F. *From Traditional Data Warehouse To Real Time Data Warehouse*. Február 2017, s. 467–477. ISBN 978-3-319-53479-4.

[3] (BPSE), B. P. S. E. *Model View Controller Pattern* [online]. 2013 [cit. 2020-03-20]. Dostupné z: http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns/mvc.html.

[4] codecademy. *Back-End Web Architecture* [online]. 2019 [cit. 2020-03-18]. Dostupné z: https://www.codecademy.com/articles/back-end-architecture.

[5] Dewald, B. *Steps Involved in Building a Data Warehouse* [online]. 2002 [cit. 2020-02-08]. Dostupné z: http://www.informit.com/articles/article.aspx?p=24902.

[6] G, H. *OLTP vs. OLAP* [online]. Brno: diffzi.com, júl 2019 [cit. 2020-02-23]. Dostupné z: https://diffzi.com/oltp-vs-olap/.

[7] Han, J. a Micheline, K. *Data mining : concepts and techniques*. 3rd ed. Waltham: Morgan Kaufmann Publishers, 2012. ISBN 978-0-12-381479-1.

[8] IAN. *What is OLAP?* [online]. 2017 [cit. 2020-02-21]. Dostupné z: https://database.guide/what-is-olap/.

[9] Jalaja, T. a Shailaja, M. A Comparative Study on Operational Database, Data Warehouse and Hadoop File System. *International Journal of Engineering and Techniques*. 2015, roč. 1, č. 4, s. 37–41. ISSN 2395-1303.

[10] JavaTpoint. *Data Warehouse Design* [online]. 2020 [cit. 2020-02-08]. Dostupné z: https://www.javatpoint.com/data-warehouse-design.

[11] Jukic, N., Jukic, B. a Malliaris, M. Online Analytical Processing (OLAP) for Decision Support. In: *Handbook on Decision Support Systems 1*. 1. vyd. Springer, Berlin, Heidelberg, 2008, s. 259–276. ISBN 978-3-540-48712-8.

[12] Kimball, R. a Caserta, J. *The data warehouse ETL toolkit: practical techniques for extracting, cleaning, conforming, anddelivering data*. 1. vyd. Wiley Publishing, 2004. ISBN 07–645–7923–1.

[13] Lacko, L. *Databáze: datové sklady, OLAP a dolování dat s příklady v Microsoft SQL Serveru a Oracle*. 1. vyd. Brno: Computer Press, 2003. ISBN 8072269690.

[14] Laravel. [online]. [cit. 2020-03-20]. Dostupné z: https://laravel.com/docs/6.x.

[15] Laravel. *The PHP Framework for Web Artisans* [online]. [cit. 2020-03-20]. Dostupné z: https://laravel.com/.

[16] McKendrick, J. *Rethinking-the-Future-of-Data-Warehousing* [online]. actian.com, apríl 2019 [cit. 2020-02-06]. Dostupné z: https://www.actian.com/wp-content/uploads/2019/04/Rethinking-the-Future-of-Data-Warehousing.pdf.

[17] OLAP.com. *Codd's Paper* [online]. 2013 [cit. 2020-02-22]. Dostupné z: https://olap.com/learn-bi-olap/codds-paper/.

[18] OLAP.com. *What is the definition of OLAP?* [online]. 2016 [cit. 2020-02-22]. Dostupné z: https://olap.com/olap-definition/.

[19] Pedersen, T. B. a Jensen, C. S. Multidimensional database technology. *Computer*. 2001, roč. 34, č. 12, s. 40–46. ISSN 0018-9162.

[20] Ponniah, P. a Caserta, J. *Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals*. 1. vyd. John Wiley & Sons, 2001. ISBN 0-471-41254-6.

[21] PostgreSQL. *What Is PostgreSQL?* [online]. 2020 [cit. 2020-03-21]. Dostupné z: https://www.postgresql.org/docs/12/intro-whatis.html.

[22] techopedia. *Database Server* [online]. 2016 [cit. 2020-03-16]. Dostupné z: https://www.techopedia.com/definition/441/database-server.

[23] Vue.js. *Introduction* [online]. [cit. 2020-03-21]. Dostupné z: https://vuejs.org/v2/guide/.

[24] Zedníček, J. *Granularita* [online]. Brno: [b.n.], apríl 2017 [cit. 2020-02-25]. Dostupné z: https://biportal.cz/BI-slovnik/granularita/.

[25] Zendulka, J. a Rudolfová, I. *Databázové systémi*. Studijní opora. Brno: FIT VUT v Brně, 2006. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIDS-IT%2Ftexts%2FDatabazove_systemy.pdf&cid=12737.

[26] Zendulka, J. a. k. *Získávanie znalostí z databáz*. Studijní opora. Brno: FIT VUT v Brně, 2009.

[27] ZenTut. *Kimball vs. Inmon Data Warehouse Architectures* [online]. 2020 [cit. 2020-02-08]. Dostupné z: https://www.zentut.com/data-warehouse/kimball-and-inmon-data-warehouse-architectures/.

Príloha A

Obsah DVD

Popis obsahu jednotlivých adresárov priloženého DVD nosiča:

A.1 Bakalarska praca - Text

V tomto adresári sa nachádza text bakalárskej práce vo formáte PDF a zdrojových súborov použitých k jeho vytvoreniu:

- **LATEX** - adresár obsahujúci zdrojové súbory písomnej práce vytvorenej v LaTeXe
- **PDF** - adresár obsahujúci písomnú prácu vo formáte *PDF*

A.2 Bakalarska praca - Zdrojove subory

Adresár uchováva všetky potrebné zdrojové súbory pre fungovanie aplikácie a skripty pre jeho testovanie:

- **Aplikacia** - adresár, obsahujúci zdrojové súbory potrebné pre beh aplikácie
- **Databazove skripty** - adresár obsahujúci skripty vytvárajúce dátové sklady pre testovanie
- **READEME.md** - súbor vytvorený vo formáte *Markdown*. Obsahuje vymenované závislostí aplikácie a postup pre lokálne nasadenie programu.