UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

ZAKLJUČNA NALOGA

OPTIMIZIRANJE ETL V PODJETJU X

EMA PERKOVIĆ

2023
ZAKLJUČNA NALOGA
PERKOVIĆ

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Optimiziranje ETL v podjetju X

(Optimising ETL in company X)

Ime in priimek: Ema Perković
Študijski program: Računalništvo in informatika
Mentor: doc. dr. Uroš Godnov

Koper, februar 2023

Ključna dokumentacijska informacija

Ime in PRIIMEK: Ema PERKOVIĆ

Naslov zaključne naloge: OPTIMIZIRANJE ETL V PODJETJU X

Kraj: Koper

Leto: 2023

Število listov: 38 Število slik: 20

Število referenc: 19

Mentor: doc. dr. Uroš Godnov

Ključne besede: podatki, obdelava, strežniki SQL, Power BI, integracija, poročilo, Excel, tabele

Izvleček:

V zaključni nalogi je obravnavan postopek obdelave in podana predstavitev velikih količin podatkov. Obravnavana je uporaba storitve SQL Server Integration Services (SSIS) in njenih orodij za združevanje podatkov iz različnih virov ter zmogljivosti orodja Power BI za manipulacijo in vizualizacijo podatkov. Naloga zajema tudi končno poročilo in strani, uporabljene za predstavitev podatkov.

Key document information

Name and SURNAME: Ema PERKOVIĆ

Title of the final project paper: OPTIMISING ETL IN COMPANY X

Place: Koper

Year: 2023

Number of pages: 38 Number of figures: 20

Number of references: 19

Mentor: Assist. Prof. Uroš Godnov, PhD

Keywords: data, processing, SQL servers, Power BI, integration, report, Excel, spreadsheets

Abstract:

This thesis will focus on the process of processing and presenting large amounts of data. The use of SQL Server Integration Services (SSIS) and its tools for combining data from different sources and the Power BI capabilities for data manipulation and visualisation will be discussed. The document will also cover the final report and the pages used to present the data.

ZAHVALA

Iskreno se zahvaljujem staršem in bratu za njihovo nenehno ljubezen in podporo na moji študijski poti. Na moj uspeh sta v veliki meri vplivala njihovo zaupanje v moje sposobnosti in pripravljenost žrtvovati se, da bi mi zagotovili vrhunsko izobrazbo.

Zahvaljujem se tudi svojemu partnerju za njegovo neomajno podporo in spodbudo. Njegovo razumevanje in spodbuda sta mi bila stalen vir motivacije in moči.

Hvaležna sem tudi svojemu mentorju doc. dr. Urošu Godnovu za usmerjanje in podporo med pisanjem te zaključne naloge. Njegovo strokovno znanje in vpogledi so bili zame neprecenljivi in zelo sem mu hvaležna za predani čas in prizadevanja, da bi mi pomagal uspeti. Še posebej sem hvaležna za idejo, ki mi jo je ponudil za mojo zaključno nalogo. Njegov predlog je bil zame prelomnica, zato mi je bilo v veselje raziskovati in proučevati to temo. Njegova spodbuda in vodstvo sta mi bila v pomoč pri uresničitvi te zamisli.

KAZALO VSEBINE

KAZALO SLIK IN GRAFIKONOV

SEZNAM KRATIC

ETL	Extraction, Transformation and Load
MSSQL	Microsoft SQL Server
SaaS	Programska oprema kot storitev – Software as a Service
SQL	Structured Query Language
SSIS	Integracijske storitve SQL-strežnika – SQL Server Integration Services
SSMS	SQL Server Management Studio
VBA	Visual Basics for Applications

1 UVOD

Vsak delovni dan mora podjetje izvesti izvoze iz različnih sistemov. Vsak sistem uporablja različne vrste formatov datotek. Vsak format pa predstavlja različne načine shranjevanja svojih podatkov. Trenutno ima podjetje X program, ki izvede izvoze iz točno določenih sistemov. Ta korak je treba avtomatizirati, da bi lahko program izvedel izvoze iz več različnih sistemov in jih izpisal v enem poročilu, ki bi bilo v enem datotečnem formatu. Težave se prav tako pojavijo pri razvrščanju prioritet pri končnem poročilu, saj program napačno opredeli pomembnost posameznega dela. Pri gradnji ali polnjenju podatkovnega skladišča se lahko pojavijo številna resnična vprašanja, ki zahtevajo natančne metode za odločanje o tem, kateri podatki bodo preneseni, kako bodo oblikovani in kako bodo povezani, če dobimo podatke iz več virov. Ta faza je ključnega pomena, zato v njej poteka postopek ETL ali zajem, preoblikovanje in nalaganje. S to tehniko lahko podatke pretvorimo na učinkovit in standardiziran način.

Namen zaključne naloge je skozi proces optimizacije postopkov ETL ugotoviti najboljše načine za optimizacijo teh postopkov v podjetju X. To bo prineslo uporabno vrednost tudi drugim podjetjem, ki se spuščajo v podobne projekte optimizacije procesov. Prav tako jim bo dalo vpogled v to, kako se na ta način izogniti procesnim napakam pri takšnih projektih. V sodobnih podjetjih je postopek ekstrakcije, transformacije in nalaganja (ETL) bistven za upravljanje podatkov. Da bi povečali produktivnost, zagotovili natančnost analize podatkov in sprejemanje odločitev, je treba optimizirati postopek ETL, saj se količina podatkov nenehno povečuje. Ker je orodij veliko in ponujajo različne funkcije, zaključna naloga ne bo podrobno obravnavala vsakega od njih; namesto tega se bo držala primera, ki se bo uporabljal z vsemi orodji, in se osredotočila na področja, ki so zanjo pomembna. Vrsta podatkovne zbirke MSSQL bo v zaključni nalogi uporabljena tako v izvorni kot ciljni podatkovni zbirki.

Pred kratkim sem imela priložnost razpravljati o zaključni nalogi s profesorjem in predstavniki podjetja. To je bila koristna izkušnja, ki je omogočila pridobiti mnenja in usmeritve ljudi iz akademskega in poslovnega sveta glede zaključne naloge.

Na srečanju smo razpravljali o temi zaključne naloge in ključnih točkah, ki sem jih nameravala predstaviti. Predavatelj in predstavniki podjetja so me pozorno poslušali in me spraševali, da bi dodatno pojasnila svoje točke.

Znanje in vpogled tako mojega profesorja kot predstavnikov podjetja sta me navdušila, njihova priporočila pa so se mi zdela praktična in spodbudna za razmišljanje. Srečanje sem zapustila z večjim zaupanjem v svojo zaključno nalogo.

Na splošno je bilo srečanje s profesorjem in predstavniki podjetja zame koristna priložnost, da sem dobila nasvete in pripombe o svoji zaključni nalogi z več vidikov. Hvaležna sem za priložnost, da sem lahko izkoristila izkušnje in znanje svojih mentorjev; to je bila odlična priložnost, da sem z njimi raziskala svoje zamisli.

Ker podatki izvirajo iz lastniškega vira, ki ga ne moremo razkriti, smo s podjetjem podpisali pogodbo o nerazkritju informacij, zato so podatki, ki se nahajajo v tej zaključni nalogi, spremenjeni. Spremenjena so imena tabel in stolpcev, ki so občutljive narave.

Glavni cilj zaključne naloge je pokazati ustrezne korake, kako iz precej nezanesljive rešitve pridemo do sodobne rešitve s strežniškim produktom.

Zaključna naloga je razdeljena na štiri poglavja. Uvod se začne s kratkim pregledom zaključne naloge in podaja problematiko te naloge. Opisuje tudi ozadje in povezano delo, zajema teoretične informacije ter ozadje, potrebno in povezano z zaključno nalogo. Poglavje 2 pojasnjuje metode izvajanja. Začne se s pregledom zaključne naloge, ki na kratko opisuje način izvajanja. Naslednje poglavje opisuje teorijo in končno poročilo. V zadnjem poglavju so povzeti najpomembnejši vidiki in podani obeti za naslednje možne korake, kateremu sledi poglavje o referencah.

1.1 Orodja in tehnologije

V zaključni nalogi smo za izvajanje raziskav in analiz uporabili različna Microsoftova orodja. Ta orodja vključujejo program Microsoft Excel za manipulacijo in vizualizacijo podatkov, Microsoft Word za pisanje in oblikovanje dokumenta naloge in Microsoft Power BI za ustvarjanje poročil.

Za uporabo teh Microsoftovih orodij smo se odločili iz več razlogov. Prvič, so široko dostopna in enostavna za uporabo, kar nam je omogočilo učinkovito organizacijo, analizo in predstavitev naših podatkov. Drugič, ponujajo širok nabor funkcij in lastnosti, ki so dobro prilagojene našim raziskovalnim potrebam. Navsezadnje pa so dobro znana in se pogosto uporabljajo na akademskem in drugih področjih, kar nam je olajšalo sodelovanje z drugimi raziskovalci in izmenjavo naših rezultatov.

Na splošno menimo, da je bila uporaba Microsoftovih orodij pomemben del našega raziskovalnega procesa in nam je pomagala pri uspešnem in učinkovitem ustvarjanju visokokakovostnih rezultatov.

Tehnologija ETL v podatkovnih skladiščih se nanaša na skupino tehnologij, ki se uporabljajo za pridobivanje, čiščenje, prilagajanje, preoblikovanje, povezovanje in

nalaganje podatkov iz različnih virov v podatkovno skladišče. Izraz »podatkovni tok« se uporablja za opis računalniških arhitektur, v katerih so številne računalniške enote logično razporejene tako, da opravljajo potrebne izračune v skladu z uporabniško določenimi merili. Strategije za optimizacijo podatkovnih tokov vključujejo načrtovanje podatkovnih tokov, predpomnjenje, vzporejanje, razporejanje delovnih mest in druge. Podatki v izvornih sistemih se lahko zaradi izboljšanega podatkovnega toka ETL hitro sinhronizirajo s podatkovnim skladiščem [4].

1.1.1 SQL

Za interakcijo s podatkovnimi zbirkami in komunikacijo z njimi se uporablja programski jezik, imenovan strukturirani poizvedovalni jezik (SQL). IBM-ovi raziskovalci so jezik SQL ustvarili v sedemdesetih letih prejšnjega stoletja. SQL je uporabniku izjemno prijazen in dostopen na različnih platformah [8]. Uporablja se za vzdrževanje in spreminjanje podatkov v relacijskih podatkovnih zbirkah. To je skupni jezik za uporabo sistemov za upravljanje podatkovnih zbirk, kot so MySQL, Oracle in Microsoft SQL Server, za dostop do podatkov in njihovo spreminjanje [13].

Podjetja in druge organizacije uporabljajo orodja SQL za ustvarjanje in spreminjanje novih tabel ter dostop do informacij in podatkov v svojih podatkovnih zbirkah ter njihovo urejanje [5].

Relacijska algebra in relacijski račun z nterkami sta bila začetna temelja SQL. Tehnično ga lahko uvrstimo med podjezike in vključuje različne vrste ukazov. Vključuje jezik poizvedovanja po podatkih (DQL), jezik za definiranje podatkov (DDL), jezik za nadzor podatkov (DCL) in jezik za manipulacijo s podatki (DML). Poizvedbe po podatkih, manipulacija s podatki (vstavljanje, posodabljanje in brisanje), opredelitev podatkov (izdelava in spreminjanje sheme) ter nadzor dostopa do podatkov so vključeni v področje uporabe jezika SQL. Kljub temu da se SQL pogosto imenuje deklarativni jezik (4GL), ima tudi postopkovne komponente [18].

1.1.2 Microsoft SQL Server Management Studio

SQL Server Management Studio (SSMS) je integrirano okolje za upravljanje katere koli infrastrukture SQL. Pakete lahko upravljate in spremljate aktivne pakete s storitvijo Integration Services programa SQL Server Management Studio (SSIS) [9]. Poleg tega lahko zaženete pakete, uvozite in izvozite pakete, preselite pakete Data Transformation Services (DTS) in nadgradite pakete Integration Services s programom Management Studio. Vsako infrastrukturo SQL lahko upravljate v integriranem okolju, ki se imenuje

SQL Server Management Studio (SSMS). Do vseh komponent strežnika SQL Server, podatkovne zbirke Azure SQL Database, upravljane instance Azure SQL Managed Instance, strežnika SQL Server on Azure VM in orodja Azure Synapse Analytics lahko dostopate, jih konfigurirate, upravljate, vodite in razvijate z uporabo SSMS. Za razvijalce in skrbnike podatkovnih zbirk na vseh ravneh SSMS ponuja enotno celovito rešitev, ki združuje široko paleto grafičnih orodij s številnimi zmogljivimi urejevalniki skript, ki omogočajo dostop do strežnika SQL Server [11].

1.1.3 Microsoft Visual Studio

Integrirano razvojno okolje (IDE) Microsoft Visual Studio se uporablja za ustvarjanje aplikacij z grafičnim uporabniškim vmesnikom (GUI), konzol, spletnih in mobilnih aplikacij, storitev v oblaku in spletnih storitev ter še več. Funkcije programa Visual Studio dopolnjujejo vsa področja razvoja programske opreme. S pomočjo tega IDE je mogoče izdelati tako upravljalno kot izvorno kodo. Uporablja se pri ustvarjanju računalniških programov, spletnih mest, spletnih aplikacij, spletnih storitev in mobilnih aplikacij [12].

1.2 Namestitev

Preden sem začela pisati zaključno nalogo, sem v računalnik namestila Microsoft SQL Server in Power BI. To je zahtevalo namestitev ustrezne programske opreme in prilagoditev parametrov mojim potrebam.

Najprej sem z Microsoftovega spletnega mesta prenesla namestitvene datoteke, nato pa sem konfigurirala strežnik Microsoft SQL Server. Programsko opremo sem nato namestila v svoj računalnik, pri čemer sem upoštevala navodila. Pri tem sem izbrala dele, ki sem jih želela namestiti, na primer orodja za upravljanje in mehanizem podatkovne zbirke. Določiti je bilo treba tudi mehanizem avtentikacije in nastavitve za določanje pravil razvrščanja, velikosti in lastnosti za shranjevanje podatkov.

Po končani namestitvi sem nastavila nastavitve strežnika Microsoft SQL Server. To je vključevalo konfiguracijo omrežnih protokolov, nastavitev zgornjih mej porabe pomnilnika RAM in procesorja ter nastavitev prijave. Z Microsoftovega spletnega mesta sem prenesla namestitvene datoteke Power BI in nastavila program v svojem računalniku, da sem lahko začela delati. Po tem sem program nastavila tako, da sem se prijavila s svojim Microsoftovim računom. To je zahtevalo nastavitve za osveževanje podatkov in sodelovanje ter povezavo programa Power BI z mojim strežnikom Microsoft SQL Server [11]. Na splošno sta bili namestitev programa in vzpostavitev ustreznih nastavitev vse, kar je bilo potrebno za nastavitev strežnika Microsoft SQL Server in programa Power BI v

mojem računalniku. Zaradi tega sem lahko ta orodja učinkovito uporabljala med delom na svoji zaključni nalogi [10].

Za zaključno nalogo sem poleg Microsoft SQL Serverja in Power BI uporabljala tudi Visual Studio 2019, in sicer zato, ker sem lahko sestavila in vzdrževala pakete ETL zaradi vrste projekta, imenovanega integracijske storitve v Visual Studiu.

Najprej sem morala na svoj računalnik namestiti Visual Studio 2019, preden sem ga lahko uporabila za svojo zaključno nalogo. Nato sem dodala nov paket integracijskih storitev kot razširitev [12]. To mi je omogočilo, da sem začela načrtovati in razvijati svoj postopek ETL.

Različna orodja in zmožnosti programa Visual Studio 2019 so mi olajšale delo s paketi ETL. Proces ETL sem na primer lahko razvila tako, da sem s pomočjo grafičnega oblikovalca povezala različne komponente, ki sem jih postavila na projektno površino. Nastavitve za vsako komponento sem lahko določila tudi v polju lastnosti, postopek ETL pa sem lahko preizkusila in odpravila težave z orodji za odpravljanje napak.

Na splošno je bil Visual Studio 2019 zaradi povezave z integracijskimi storitvami uporabno orodje za mojo zaključno nalogo. Omogočilo mi je učinkovito izdelavo in upravljanje paketov ETL ter njihovo vključitev v rutino upravljanja podatkov.

Za uvoz Excelovih datotek sem morala namestiti še 64-bitni Access Runtime 2016 in 32-bitni Microsoft Access Database Engine 2010.

1.3 ETL

V sistemu za skladiščenje podatkov proces ETL predstavlja podatkovni tok, ki pridobiva, preoblikuje, čisti in nalaga podatke v oblike, primerne za dodatno analizo in raziskovanje. Ta postopek, ki se pogosto uporablja pri pobudah za poslovno obveščanje, velja za enega najbolj zapletenih, dolgotrajnih in k napakam nagnjenih procesov. Težave pogosto povzroča pomanjkanje standardizacije med zmogljivostmi, ki jih zagotavljajo različne tehnologije na tem področju. Ta orodja uporabljajo več osnovnih jezikov, naborov funkcij in čarovnikov, zaradi katerih so konceptualno primerljivi modeli na fizični ravni videti zelo različni [19].

Temeljna zamisel postopka ETL je, da se iz sistema ETL povežemo z izvornimi podatkovnimi sistemi, pripravimo podatke za preoblikovanje in nato izvedemo preoblikovanje v pomnilniku ali sproti. Ciljni sistem, podatkovno skladišče, se nato naloži

s spremenjenimi podatki. Čeprav pri samopostrežni metodi BI ni ločenega sistema ETL, saj so funkcionalnosti vgrajene v samo orodje za ustvarjanje BI, podatki pa se v model BI vnesejo lokalno, še vedno veljajo isti koncepti [15].

Pri gradnji ali polnjenju podatkovnega skladišča se lahko pojavijo številna resnična vprašanja, ki zahtevajo natančne metode za odločanje o tem, kateri podatki bodo preneseni, kako bodo oblikovani in kako bodo povezani, če dobimo podatke iz več virov. Ta faza je ključnega pomena, zato v njej poteka postopek ETL ali zajem, preoblikovanje in nalaganje. S to tehniko lahko podatke pretvorimo na učinkovit in standardiziran način [15].

Med ekstrakcijo podatkov se neobdelani podatki kopirajo ali izvažajo z izvornih lokacij na pripravljalno območje. Ekipe za upravljanje podatkov lahko podatke pridobivajo iz različnih strukturiranih in nestrukturiranih virov podatkov [3].

Na območju za pripravo se neobdelani podatki obdelajo. Za predvideni primer analitične uporabe se podatki na tem mestu spremenijo in konsolidirajo. Filtriranje, čiščenje, odstranjevanje podvojenih zapisov in potrjevanje podatkov so nekatere od nalog, ki so lahko vključene v to fazo [3].

V tej končni fazi se pretvorjeni podatki pošljejo iz območja pripravljanja v ciljno podatkovno skladišče. To pogosto vključuje začetno nalaganje vseh podatkov, ponavljajoče se nalaganje postopnih sprememb podatkov, redkeje pa popolno osvežitev za popolno odstranitev in zamenjavo vseh podatkov v skladišču. Večina podjetij, ki uporablja ETL, ima avtomatizirane, dobro opredeljene, paketno vodene procese, ki potekajo neprekinjeno. ETL običajno poteka po delovnem času, ko je uporaba podatkovnega skladišča in izvornih sistemov najmanjša. Zato je sistem podatkovnega skladišča večino dneva na voljo vsem poslovnim uporabnikom [3].

2 INTEGRACIJSKE STORITVE STREŽNIKA SQL

2.1 Predpriprava

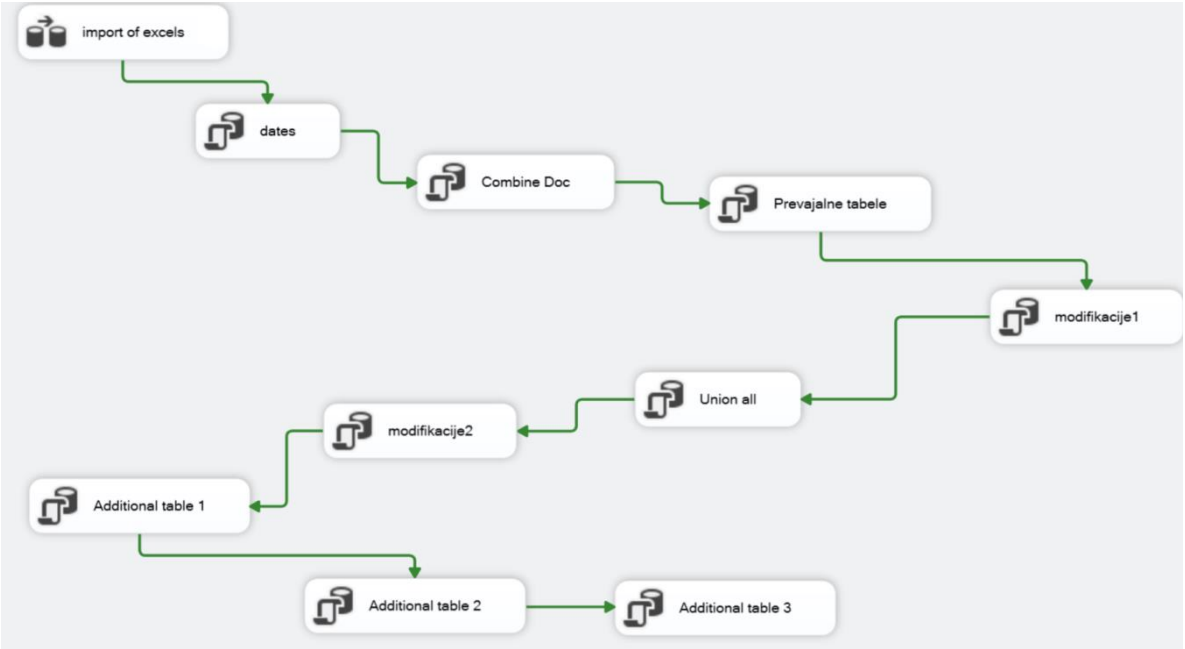
Pred začetkom uvoza Excelov in samim začetkom izdelave programa sem morala prebrati dokumentacijo o že narejenem programu. Med delom na tem projektu sem v njem našla nekaj napak, ki so nastale pri pisanju programa. Da bi te napake odpravila, sem morala opraviti tudi nekaj raziskav in spremeniti obstoječo kodo. Program je bil predhodno zgrajen v bazi Access v kodi VBA, zato sem v tem koraku prebrala dokumentacijo jezika VBA in se spoznala z jezikom za lažje razumevanje.

Prejela sem skupno osem Excelovih datotek in bazo Access. Postopek ETL je nato vključeval pridobivanje podatkov iz osmih Excelovih datotek in podatkovne zbirke Access, njihovo preoblikovanje v obliko, združljivo z obliko poročila, in nalaganje v pripravljalno območje, kjer sem lahko združila vse podatkovne sklope in ustvarila končno poročilo. Opazila sem, da je oblika končnega poročila drugačna od prejetih podatkov, zato sem se posvetovala s finančnim analitikom, ki je bil seznanjen z vsebino dokumentov. Po posvetovanju z njim sem prejela vsa potrebna navodila za spreminjanje podatkov. Ko sem imela vse potrebne informacije, sem lahko začela izvajati projekt.

V programu Microsoft Visual Studio sem uporabila storitev SQL Server Integration Services (SSIS). Excelove datoteke sem uvozila prek ustvarjene povezave OLE DB, ki se uporablja za povezavo z različnimi viri podatkov. SSIS se uporablja pri kontrolnem in podatkovnem toku. V SSIS je glavna razlika med kontrolnim in podatkovnim tokom v tem, da lahko kontrolni tok naenkrat linearno izvede le eno opravilo. Podatkovni tok pa lahko hkrati izvaja več preoblikovanj [7]. V SSIS se razdeli projekt na 11 sestavnih delov, od katerih ima vsak svoj poseben namen. V naslednjem razdelku dokumenta je podrobno opisana vsaka od teh komponent.

2.2 Izvedba

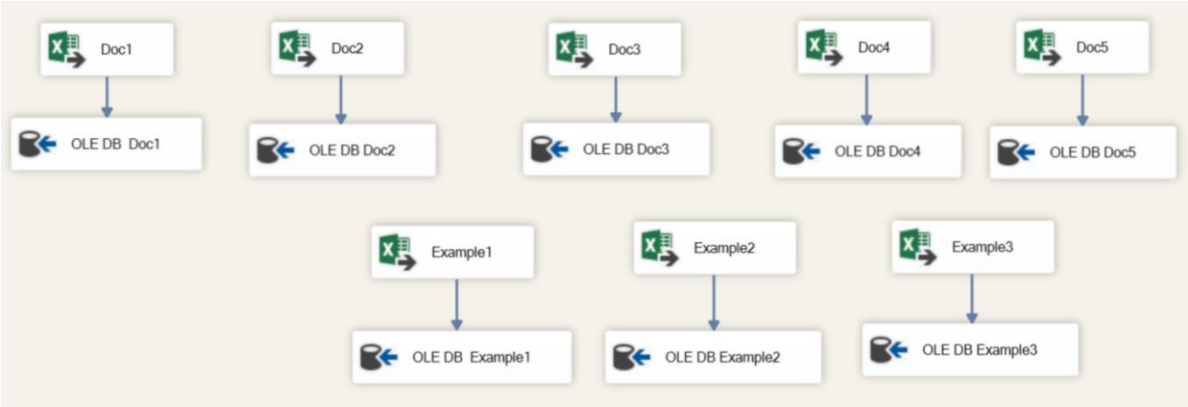
Kontrolni tok se začne z uvozom Excelov. Na sliki 1 je vidna sestava celotnega projekta. Večina korakov se izvede v kontrolnem toku (*Control Flow*), uvoz Excelov pa v podatkovnem toku (*Data Flow*).



Slika 1: Kontrolni tok

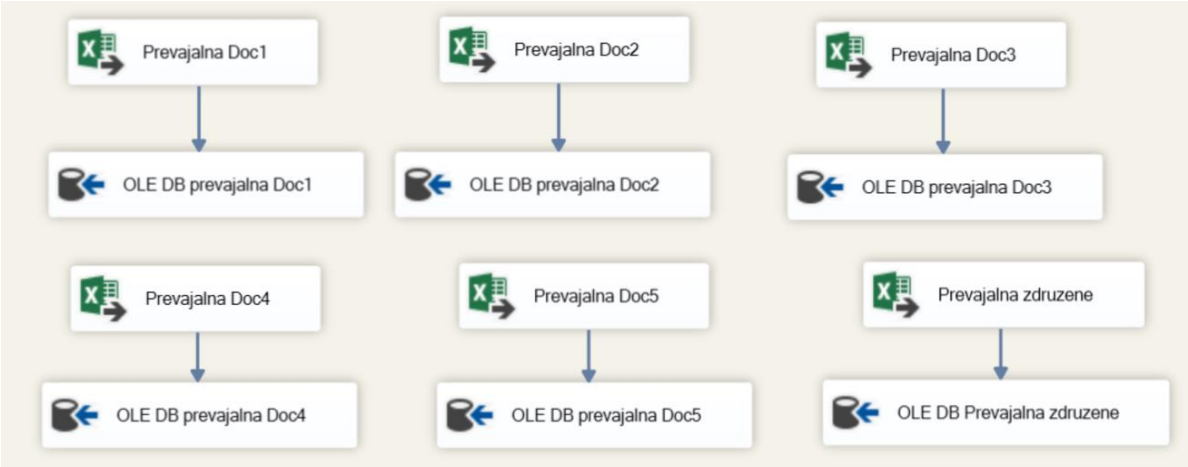
2.2.1 Uvoz Excelov

Na začetku izberem vire za pridobivanje podatkov. V mojem primeru sta vira podatkov datoteka Excel 2010 in strežnik SQL Server. Na tej stopnji iz virov izberem samo pomembne stolpce. Ta korak je bil opravljen v podatkovnem toku (*Control Flow*). V tem koraku se uvozi osem podatkovnih tabel in šest prevajalnih tabel, kot je prikazano na slikah 2 in 3, ki služijo kot opis posameznih podatkov, ki se pojavljajo v podatkovnih tabelah.



Slika 2: Uvoz podatkov

Vsaka posamezna prevajalna tabela predstavlja osebo, ki je zadolžena za realizacijo pravnih dokumentov, poleg nje pa so opisani statusi vsakega izmed teh dokumentov. Ker se statusi spreminjajo na mesečni ravni, sem podatke uvozila kot celotno tabelo, tako da je manj odvečnih podatkov in manj mest, ki jih je treba posodobiti. Tako lahko tabele še vedno kasneje združim in spreminjam samo prevajalne tabele.



Slika 3: Uvoz prevajalnih tabel

2.2.2 Brisanje tabel

Podatki, kopirani iz izvornih sistemov, so začasno shranjeni v območju za shranjevanje podatkov, imenovanem vmesne tabele, preden se preoblikujejo in naložijo v končni sistem. To območje vmesnih tabel služi kot varovalni mehanizem med izvornim sistemom in končnim ciljem ter omogoča spreminjanje, potrjevanje in preverjanje podatkov, preden se naložijo v ciljno podatkovno zbirko. Podatki so bili nato pred nalaganjem v končno poročilo dodatno obdelani in spremenjeni v tabelah za presledke.

Območje za shranjevanje podatkov je v arhitekturi podatkovnega skladišča najpogosteje potrebno zaradi časovnih omejitev. Z drugimi besedami, vsi potrebni podatki morajo biti enostavno dostopni, preden se lahko dodajo v podatkovno skladišče. Zaradi spreminjajočih se poslovnih ciklov, ciklov obdelave podatkov, omejitev strojne opreme in omrežnih virov ter geografskih spremenljivk ni mogoče pridobiti vseh podatkov iz vseh operativnih podatkovnih zbirk naenkrat [2]. Ker se podatki vsakodnevno spreminjajo v podjetju in jih je treba vsak dan znova uvoziti v program in izbrisati predhodne podatke, medtem ko osnovna oblika tabele ostane enaka, sem uporabila ukaz TRUNCATE TABLE, ki odstrani vse vrstice iz tabele, vendar struktura tabele, njeni stolpci in indeksi ostanejo enaki, ter ukaz CREATE TABLE »ime tabele«, ki ustvari novo tabelo, kot je prikazano na sliki 4 na primeru tabele Doc1. Prvi ukaz izbriše vse podatke v tabeli, ne izbriše pa dejanske tabele, drugi ukaz pa generira tabelo in v oklepaju opredeli seznam, ki opredeljuje vsak stolpec v tabeli in tip podatkov, ki jih vsebuje.

```
IF OBJECT_ID('dbo.StagginDoc1') IS NOT NULL
    TRUNCATE TABLE dbo.StagginDoc1

CREATE TABLE dbo.StagginDoc1 (
    Link    NVARCHAR (255),
    "Action Due Date"   date,
    "Completed Date"    NVARCHAR (255),
    [Action]    NVARCHAR (255),
    "Country / Region"  NVARCHAR (255),
    MarkName    NVARCHAR (255),
    "Current Application Number"    NVARCHAR (255),
    "Current Registration Number"   NVARCHAR (255),
    Notes   NVARCHAR (255),
    TRADEMARKMASTERID   NVARCHAR (255),
    [Status]    NVARCHAR (255)

);
```

Slika 4: Priprava tabel

2.2.3 Datumi

Naslednji korak je bila izdelava preglednice datumov, ki ima pomembno vlogo v končnem poročilu v programu Power BI, o katerem bom govorila v naslednjem poglavju. Za kreacijo preglednice datumov, kot je razvidno na sliki 5, sem uporabila ukaz DROP TABLE, ki v primeru, da tabela ni prazna, tabelo v celoti izbriše. V tej lastnosti se razlikuje od ukaza TRUNCATE TABLE, ki je bil uporabljen pri gradnji prevajalnih tabel. Pri preglednici datumov sem se osredotočila na začetni in končni datum, saj sem tako dobila razpon datumov, ki so vsebovali začetni, končni in vse vmesne datume. To lastnost je uporabnik kasneje uporabil v končnem poročilu za določitev datumskega razpona. Začetni datum je določen, končni datum pa je urejen po dnevih od začetnega datuma. Na primer, začetni datum je 1. 1. 2020, končni datum pa je oddaljen 2556 dni od začetnega datuma. To mi je uspelo narediti z ukazom DATEADD, ki vzame tri argumente. Prvi določa datumski del, pri katerem sem si izbrala dan. Naslednji določa številko, kjer minus pred številko predstavlja odštevanje, pozitivna številka pa predstavlja prištevanje zadnjemu argumentu. Zadnji argument pa predstavlja datum. To pomeni, da bo poročilo prikazalo zapise od januarja 2020 do 31. decembra 2026. Z drugimi besedami, če želim zapise od leta 2020 do 2026, potem moram začetnemu datumu januar 2020 dodati 2556 dni. Na podlagi tega znanja sem dodatno dodala merila, tako da bodo prikazani samo zapisi, pri katerih je končni datum večji od začetnega datuma. Za to sem v poizvedbi uporabila stavek while, ki je preveril, ali pogoj drži, in nato vnesel datum v novo tabelo, imenovano *Calendary*, ter poskrbel, da se začetni datum večkrat poveča, dokler ne doseže decembra 2026, nato pa se povečevanje ustavi.

```
if object_id('Calendary') is not null
    drop table Calendary


CREATE TABLE [Calendary]
(
    [CalendarDate] DATE
)

DECLARE @StartDate DATE
DECLARE @EndDate DATE
SET @StartDate = '01/01/2020'
SET @EndDate = DATEADD(d, 2556, @StartDate)

WHILE @StartDate <= @EndDate
      BEGIN
            INSERT INTO [Calendary]
            (
                  CalendarDate
            )
            SELECT
                  @StartDate

            SET @StartDate = DATEADD(dd, 1, @StartDate)
      END
```

Slika 5: Izdelava koledarja

2.2.4 Združevanje posamičnih datotek

SSIS (SQL Server Integration Services) je platforma za ustvarjanje aplikacij za integracijo podatkov in delovnih tokov, kot je bilo že opisano. Uporablja se za pridobivanje, obdelavo in nalaganje podatkov iz različnih virov, vključno z Excelovimi datotekami, v strežnik, kot je podatkovna zbirka SQL Server [16].

V naslednjem koraku je sledilo združevanje datotek, ki v imenu vsebujejo »Example«. Obstajajo tri takšne tabele. Ta korak je bil izvzet iz celotne zveze združevanja datotek, ker je treba spremeniti samo te tri tabele. Nekatere vrednosti je treba odstraniti ali izključiti iz končnega rezultata. Ločevanje postopkov lahko pripomore k boljši organizaciji in vzdrževanju podatkovne zbirke. Če je zapleten postopek razdeljen na manjše, ločene postopke, je kodo lažje razumeti in vzdrževati. To omogoča tudi posodobitve ali spremembe posameznih elementov postopka, ne da bi to vplivalo na sistem kot celoto.

Drugič, izolacija postopkov lahko pripomore k boljšemu delovanju podatkovne zbirke. Kompleksen postopek je mogoče razdeliti na manjše, posamezne operacije, vsak korak pa je mogoče optimizirati za učinkovitejše delovanje, kar pripomore k izboljšanju splošnega delovanja sistema.

Tretjič, ločevanje postopkov lahko poveča tudi varnost podatkovne zbirke. Z razdelitvijo zapletenega postopka na manjše postopke je mogoče enostavneje upravljati dostop do posameznih komponent postopka in zagotoviti, da imajo dostop do podatkov in operacij, potrebnih za posamezen postopek, samo pooblaščeni uporabniki.

Na splošno se lahko s pomočjo ločenih postopkov v strežniku SQL Server izboljšajo organizacija, zmogljivost in varnost zbirke podatkov, kar lahko olajša upravljanje in vzdrževanje sistema. Z drugimi besedami, ustvariti sem morala vmesni korak, ki vzame spodnje tri tabele, jih po potrebi spremeni in nato ustvari eno samo tabelo s končnim rezultatom.

Ukaz SELECT INTO »ime tabele« je bil uporabljen za preslikavo podatkov iz vseh treh tabel v končno tabelo z imenom *Examples*. Za združevanje treh tabel sem uporabila ukaz UNION ALL, ki se razlikuje od ukaza UNION v tem, da ohrani vse podvojene vrednosti. Osnovni pogoji za združevanje podatkov so enaki, ne glede na to, ali se uporabi UNION ali UNION ALL. Vsak ukaz SELECT mora vrniti enako število stolpcev in stolpce v enakem vrstnem redu, da se lahko združijo, vrnjeni stolpci pa morajo imeti podobne podatkovne tipe. Preverila sem, ali vsi pogoji držijo, začela sem s seštevanjem stolpcev. Ko sem izračunala število stolpcev v vsaki ločeni tabeli, sem opazila, da ima druga tabela, imenovana *Example2,* stolpec manj kot ostale tabele. To sem rešila tako, da sem ročno dodala vrstico, ko je prišlo do ustvarjanja vmesnih tabel (*Stagging tables*). Ročno dodajanje stolpca je lahko koristno, če je treba nove podatke, ki se nalagajo v vmesno tabelo, preoblikovati ali kako drugače spremeniti, preden so pripravljeni za vstavitev v končno ciljno tabelo, saj se vmesne tabele običajno uporabljajo kot začasna območja za shranjevanje podatkov, ki se nalagajo v zbirko podatkov. Zato se lahko podatki pred dokončno obdelavo in vstavitvijo v ciljno tabelo očistijo in pripravijo v preglednici za pripravo.

Slika 6 prikazuje, kako je videti SQL za ustvarjanje vmesne tabele.

```
if object_id('dbo.Examples') is not null
    drop table dbo.Examples

    SELECT [CASE NUMBER]
           ,CATCHWORD
           ,[CASE TYPE]
           ,CC
           ,[STATUS]
           ,REMARK
           ,[IS_EXTERNAL]
           ,[TERM ID]
           ,[TERM CATEGORY]
           ,[STATUS TEXT]
           ,[TASK DESCRIPTION]
           ,[ACTION / TASK NOTES]
           ,[TERM DATE]
           ,[EXTERNAL TERM DATE]
           ,[TERM RESPONSIBLE]
           ,[APPLICATION DATE]
           ,[APPLICATION NUMBER]
           ,[REGISTRATION DATE]
           ,[REGISTRATION NUMBER]
           ,[PROVE USE]
           ,[NEXT RENEWAL]
           ,APPLICANT
           ,[GROUP]
           ,CLIENT
           ,[PARTNER]
           ,TEAM

  INTO  [dbo].[Examples]
FROM
(
       SELECT     *
    FROM         [dbo].[Examples1]
    UNION ALL
    SELECT     *
    FROM        [dbo].[Examples2]
    UNION ALL
    SELECT     *
    FROM        [dbo].[Examples3])a
```

Slika 6: Združitev tabel Example v eno tabelo, imenovano Examples

2.2.5 Prevajalne tabele

Kot že omenjeno, prevajalne tabele predstavljajo osebe, ki so zadolžene za realizacijo pravnih dokumentov, poleg oseb pa so opisani statusi vsakega izmed teh dokumentov. V tem koraku se združita prevajalna tabela in posamezna tabela, v primeru, prikazanem na

sliki 7, sta se združili prevajalna tabela »Prevajalna_zdruzene« in tabela »Examples«, ki smo jo v predhodnem koraku združili iz treh tabel. Za združevanje dveh tabel sem uporabila ukaz LEFT JOIN, ki združi tabeli na podlagi pogoja. Tudi če v desni tabeli ni ujemanj, povezava SQL LEFT JOIN vrne vse vrstice iz leve tabele. To pomeni, da bo združitev še vedno vrnila vrstico v rezultatu, vendar z NULL v vsakem polju iz desne tabele, če klavzula ON ustreza nič vnosom v desni tabeli [17]. To pomeni, da levo združevanje vrne vse vrednosti iz leve tabele skupaj z vsemi ujemajočimi se vrednostmi iz desne tabele ali NULL, če ni nobenega predikata združevanja, ki bi se ujemal [17].

Slika 7 prikazuje, kako uporabiti polje STATUS TEXT za povezavo tabel Examples in Prevajalna_zdruzene. V tem primeru je STATUS TEXT združen stolpec; ker je vključen v stavek SELECT, je tudi prikazan v rezultatih poizvedbe in vse skupaj je shranjeno v novi tabeli, imenovani AllExamples. Izraz CASE ovrednoti vhodne podatke in jih razdeli na tri skupine: A, B in C. V primeru, da se vrstica v stolpcu TEAM začne z WIT, ji CASE pripiše vrednost A, če pa se vrstica konča z GLU, ji CASE pripiše vrednost B. Vse ostale vrednosti v stolpcu TEAM, ki ne izpolnjujejo prvega ali drugega pogoja, pripadajo vrednosti C. Rezultat, ki ga vrne, se shrani v nov stolpec, poimenovan SYSTEM.

```
if object_id('AllExamples') is not null
    drop table AllExamples

    SELECT [CASE NUMBER],[CATCHWORD],[CASE TYPE],[CC],[STATUS],[REMARK],[IS_EXTERNAL(1=TRUE)],
    [TERM ID],[TERM CATEGORY],C.[STATUS TEXT],[ACTION TEXT],[TERM DATE],[EXTERNAL TERM DATE],
    [TERM NOTES],[TERM RESPONSIBLE],[APPLICATION DATE],[APPLICATION NUMBER],
    [REGISTRATION DATE],[REGISTRATION NUMBER],[PROVE USE],[NEXT RENEWAL],
    [APPLICANT],[GROUP],[CLIENT],[PARTNER],[TEAM], P.[New_StatusText],
    CASE
        WHEN  TEAM LIKE '%WIT' THEN 'A'
        WHEN  TEAM LIKE 'GLU%' THEN 'B'
        ELSE  'C'
        END AS [SYSTEM],
        C.[STATUS TEXT] AS [TASK DESCRIPTION]

    INTO [dbo].[AllExamples]

    FROM  [dbo].[Examples] AS C
LEFT JOIN [dbo].[Prevajalna_zdruzene] AS P
    ON C.[STATUS TEXT] = P.[STATUS TEXT]
    ORDER BY [CASE NUMBER]
```

Slika 7: Združevanje prevajalnih tabel s tabelo Examples

2.2.6 Spremembe

Da bi bili podatki pripravljeni za integracijo, naslednji korak, imenovan modifikacija, najprej očisti podatke z iskanjem in popravljanjem (ali odstranjevanjem) morebitnih napak. Namen je ustaviti preoblikovanje »slabih podatkov« [14]. Da bi zagotovila kakovost podatkov glede na omejitve virov, jih po pridobitvi iz virov preverim. Predhodno je bila

koda napisana v jeziku VBA s funkcijo iif, ki vrne eno od dveh vrednosti, odvisno od tega, ali je izraz ovrednoten kot True ali False. V SQL sem uporabila funkcijo CASE. Ta korak se izvede na združevanju posamičnih datotek, ki v imenu vsebujejo besedo Example. V tem koraku je sledilo preoblikovanje podatkov, in sicer iz malih tiskanih črk v velike tiskane. To je bilo storjeno z uporabo ukaza CASE, kot je prikazano na sliki 8, ki iterira čez pogoje in vrne vrednost, ko je pogoj izpolnjen. Skupin je pet: GATES, JOBS, JANKOVIC, WALKER in MINAJ. Če dokumenti ne spadajo v nobeno od naštetih skupin, dobijo oznako OTHERS.

```
UPDATE [dbo].[PBI_Examples]

SET [TEAM] = CASE
                WHEN [TEAM] LIKE '%Gates%' THEN 'GATES'
                WHEN [TEAM] LIKE '%Jobs%' THEN 'JOBS'
                WHEN [TEAM] LIKE '%Jankovic%' THEN 'JANKOVIC'
                WHEN [TEAM] LIKE '%Walker%' THEN 'WALKER'
                WHEN [TEAM] LIKE '%WIT%' THEN 'MINAJ'
                WHEN [TEAM] LIKE '%GLU%' THEN 'MINAJ'
                ELSE 'OTHERS'
            END;
```

Slika 8: Primer uporabe ukaza CASE

2.2.7 Združevanje vseh datotek

Kot zadnjo tabelo sem ustvarila tabelo *Final*. Že iz imena je razvidno, da je to končna tabela, ki bo predstavljala končno poročilo za Power BI. Združila sem šest Excelovih tabel z uporabo ukaza UNION ALL. Dodatno sem dodala stolpec TimelineNo, ki na ukaz CASE razvrsti stolpec *Action Due Date* glede na število minulih dni od trenutnega datuma. Ker potrebujem podatke od trenutnega dneva, je zadnji argument definiran s funkcijo GETDATE, ki vrne trenutni datum in uro. Ker pa v končnem poročilu potrebujem samo datum, brez ure, uporabim ukaz CAST in ukaz as DATE, ki pretvori formatiran datumski izraz v datumski tip date.

Stolpec TimelineNo razdelim na pet skupin. V prvi skupini odštejem od današnjega datuma sedem dni in mu določim skupino –1. V drugo skupino spadajo trenutni datumi in jim določim skupino 0. Tretji skupini datumov prištejem osem dni in jim dodelim skupino 1, četrti pa 15 dni in jim dodelim skupino 2. Vrstice, ki ne spadajo v do sedaj določene skupine, pripadajo skupini 3, kot je vidno na sliki 9.

Na enak način sem dodala stolpec *Category*, ki sem se ga lotila na enak način kot *TimelineNo*, pogoji v ukazu CASE ostanejo enaki, spremeni se le izpis, ko je pogoj resničen. Kot vidimo na sliki 9, se skupine delijo na: in the past(more than 1 week), in the

past(less than 1 week), next 7 days, next 14 days in more than 14 days. Tako poimenovanje pride prav pri končnem poročilu, kjer uporabnik jasno vidi, kateri dokumenti se približujejo roku oddaje.

```
CASE
    WHEN  [Action Due Date] <  CAST(DATEADD(day, -7, GETDATE()) as DATE) THEN -1
    WHEN  [Action Due Date] < GETDATE() THEN 0
    WHEN  [Action Due Date] < CAST(DATEADD(day, 8, GETDATE()) as DATE) THEN 1
    WHEN  [Action Due Date] < CAST(DATEADD(day, 15, GETDATE()) as DATE) THEN 2
    ELSE  3
    END AS [TimelineNo],

CASE
    WHEN  [Action Due Date] < CAST(DATEADD(day, -7, GETDATE()) as DATE) THEN 'IN THE PAST(MORE THAN 1 WEEK)'
    WHEN  [Action Due Date] < GETDATE() THEN 'IN THE PAST(LESS THAN 1 WEEK)'
    WHEN  [Action Due Date] < CAST(DATEADD(day, 8, GETDATE()) as DATE) THEN 'NEXT 7 DAYS'
    WHEN  [Action Due Date] < CAST(DATEADD(day, 15, GETDATE())as DATE) THEN 'NEXT 14 DAYS'
    ELSE  'MORE THAN 14 DAYS'
    END AS [CATEGORY]
```

Slika 9: Razvrščanje datumov v skupine

2.2.8 Spremembe na končni tabeli

V tem koraku sem imela podatke, ki so ustrezali skoraj vsem kriterijem, potrebna je bila še kratka dodatna sprememba na končni tabeli *Final*. Predhodno sem izvedla ukaz UPDATE na tabeli *Examples*, ki je bila prvotno združena iz treh posameznih Excelov, sedaj pa sledijo spremembe, ki bodo vplivale na celotno končno tabelo. Najprej poskrbim za izbris vseh vrstic, kjer seže stolpec *External Term Date* preko leta 2000, to storim, ker podjetje ne potrebuje toliko starih podatkov, in omogočim manjšo končno datoteko, z drugimi besedami, znebim se odvečnih podatkov. Prav tako se znebim vrstic z datumi, ki predstavljajo eno leto v prihodnost. Razlog, da lahko izbrišem te datume, je, da podjetje vsakodnevno dobiva Excele, ki se spreminjajo na dnevni ravni. V zadnjem koraku poskrbim, da je tabela posodobljena v stolpcu *Term Responsible*, in zamenjam vrednosti NULL z vrednostjo »Unkown«. S tem omogočim boljšo in lažjo berljivost v končnem poročilu. Slika 10 prikazuje uporabo teh funkcij.

```
DELETE
FROM [dbo].[Final]
WHERE [EXTERNAL TERM DATE] < '2000-01-01'


DELETE
FROM [dbo].[Final]
WHERE [EXTERNAL TERM DATE] > DATEADD(YEAR,1,GETDATE())


UPDATE [dbo].[Final]
SET [TERM RESPONSIBLE] = 'UNKNOWN'
WHERE [TERM RESPONSIBLE] IS NULL
```

Slika 10: Koda urejanja podatkov tabele Final

2.2.9 Dodatne tabele

V zadnjem koraku sledijo tri posamezne procedure, ki se prav tako pojavijo v končnem poročilu Power BI. Vse potrebne podatke že imam, treba jih je še dodatno oblikovati za učinkovitejše poročilo. Prva procedura definira novo tabelo, imenovano *Categories*. V tej proceduri poskrbim za izpis potrebnih stolpcev. Ti stolpci so *Category*, *Team*, *Term Responsible*, *Number of deadlines* ter *Number of deadlines2*. *Number of deadlines2* se deli na dve skupini, in sicer v prvi skupini ukaz CASE vse vrstice, ki so null, označi z nič, vse preostale vrstice pa pusti take, kot so. Stolpce pridobim z uporabo ugnezdene poizvedbe SELECT, ki je v mojem primeru ugnezdena znotraj stavka FROM. Najprej se ovrednoti notranja poizvedba v stavku FROM, nato pa se rezultati vrednotenja shranijo v novo začasno relacijo. Nato se ovrednoti zunanja poizvedba, pri čemer se iz začasne relacije izberejo samo tiste urejene množice vrednosti, ki ustrezajo predikatu zunanje poizvedbe. Uporabila sem ugnezdeno poizvedbo, ker ponuja preprosto in učinkovito rešitev za upravljanje poizvedb, ki so odvisne od rezultatov druge poizvedbe.

Vsaka vrstica iz prve tabele je združena z vsako vrstico iz druge in tretje tabele z uporabo povezave CROSS JOIN. Kartezično združevanje je drugo ime za to obliko združevanja. Skratka, pri odločanju o uporabi CROSS JOIN v določeni poizvedbi je treba upoštevati število tabel, ki bodo združene. Časovno omejitev, ki se šteje kot opozorilo pri uporabi gnezdenih zank, sem rešila z uporabo ukaza DISTINCT, ki vrne manj vrstic (unikatne) in tako razbremeni CROSS JOIN. Celotna ugnezdena tabela je shranjena v tabelo z imenom CTE2, kot je razvidno s slike 11.

```
if object_id('Categories') is not null
    drop table Categories

    SELECT
         [CTE2].[CATEGORY]
        ,[CTE2].[TEAM]
        ,[CTE2].[TERM RESPONSIBLE],

    CASE
        WHEN [CTE3].[NUMBER OF DEADLINES2] IS NULL THEN 0
        ELSE [CTE3].[NUMBER OF DEADLINES2]
        END AS [NUMBER OF DEADLINES2],

    CAST(NULL AS INT) AS [NUMBER OF DEADLINES]

    INTO [Categories]
    FROM
        (SELECT DISTINCT
             [Unique].[CATEGORY]
            ,[Final].[TEAM]
            ,[Final].[TERM RESPONSIBLE]
                    FROM [dbo].[Final]
    CROSS JOIN

    (SELECT DISTINCT  [CATEGORY]
                    FROM [dbo].[Final])  AS [Unique] )  AS [CTE2]
    LEFT JOIN
    (SELECT
         [CATEGORY]
        ,[TEAM]
        ,[TERM RESPONSIBLE]
        ,COUNT([EXTERNAL TERM DATE]) AS [NUMBER OF DEADLINES2]

    FROM [dbo].[Final]
    GROUP BY  [CATEGORY]
        ,[TEAM]
        ,[TERM RESPONSIBLE])  AS [CTE3]

    ON [CTE2].[CATEGORY] = [CTE3].CATEGORY AND
       [CTE2].[TEAM] = [CTE3].[TEAM] AND
       [CTE2].[TERM RESPONSIBLE] = [CTE3].[TERM RESPONSIBLE]
```

Slika 11: Sestava dodatnih tabel

Nato sledi združevanje tabel CTE2 in CTE3. To sem storila z uporabo ukaza LEFT JOIN, ki poskrbi za izpis vseh zapisov iz prve tabele in usklajene zapise iz druge tabele. Nato ustvarim novo tabelo z imenom *CATEGORY_TEMPORARY*, v kateri ustvarim nov stolpec

z imenom NEXT 14 DAYS. V poizvedbi SELECT sem poskrbela, da so rezultati urejeni v grupiranem vrstnem redu, in na vsakega od rezultatov prikaže enaka kategorija za ta dan.

Ta tabela vključuje vse vrstice, ki so povezane s prihodnostjo. Potrebovala sem nov stolpec, ki bi predstavljal vsoto rokov, zato sem uporabila ukaz SUM. Želim, da ta tabela prikaže kategorijo, osebo, ki je zanjo odgovorna, in koliko rokov ima ta oseba v prihodnosti. Zato sem uporabila stolpca TEAM in TERM RESPONSIBLE. Za dodajanje novega stolpca, ki sešteva število rokov, sem uporabila ukaz SUM(), kot je razvidno na sliki 12.

```
if object_id('Category_temporary') is not null
drop table Category_temporary
---Next 14 days (Includes next 7 days)
    SELECT
        'NEXT 14 DAYS' AS [NewCategory]
        ,[TEAM]
        ,[TERM RESPONSIBLE]
        ,SUM([NUMBER OF DEADLINES2]) AS [NUMBER OF DEADLINES2]
    INTO [Category_temporary]
    FROM [dbo].[Categories]
    WHERE [CATEGORY] LIKE 'NEXT%'
    GROUP BY [TEAM], [TERM RESPONSIBLE]
```

Slika 12: Izdelava pomožne tabele kategorij za prihodnost

Z ukazom SQL je bila izdelana začasna tabela (prikazana na sliki 13). Nove informacije, ki jih je bilo treba dodati v tabelo *Categories*, so bile začasno shranjene v tej tabeli.

Trenutna tabela Categories je bila nato posodobljena z levo združitvijo z uporabo začasne tabele. Združitev SQL, imenovana levo združevanje, združi vrstice iz dveh tabel z ujemanjem vrstic na podlagi vnaprej določenih stolpcev. V tem primeru so se vrstice iz začasne tabele ujemale z vrsticami iz trenutne tabele *Categories* z uporabo stolpcev CATEGORY, TERM in TERM RESPONSIBLE.

Z levo združitvijo je nastala nova, posodobljena različica preglednice *Categories*, ki je vključevala vse novo dodane vrstice iz začasne preglednice in začetne podatke iz stare preglednice. Spremenjeno tabelo je bilo nato mogoče uporabiti za dodatne raziskave ali obdelavo.

Za posodobitev preglednice *Categories* je bilo treba najprej ustvariti začasno preglednico za nove podatke, ki so bili nato združeni s trenutnimi podatki v preglednici *Categories* z

uporabo leve združitve. Tako je bilo mogoče dodati vse nove podatke, ki so bili prisotni v začasni preglednici, hkrati pa ohraniti stare podatke.

```
UPDATE Z
SET Z.[NUMBER OF DEADLINES2] = H.[NUMBER OF DEADLINES2]
FROM [dbo].[Categories] AS Z
LEFT JOIN [dbo].[Category_temporary] AS H
ON (Z.[CATEGORY] = H.[NewCategory] AND
    Z.[TEAM] = H.[TEAM] AND
    Z.[TERM RESPONSIBLE] = H.[TERM RESPONSIBLE])

WHERE Z.[CATEGORY] LIKE 'NEXT 14 DAYS'
```

Slika 13: Posodobitev tabele Categories

Nato sem ta korak v celoti ponovila za naslednjo kategorijo, ki se imenuje *MORE THAN 14 DAYS*, in tako povzela vse kategorije, ki se navezujejo na prihodnost.

V drugi proceduri sem se osredotočila na dnevne roke. Najprej sem ustvarila novo tabelo z imenom Daily, ki ima osem stolpcev. Podatke za navedene stolpce sem dobila z uporabo ukaza CROSS JOIN, kjer sem združila tabelo *Final* samo s seboj, kot je vidno na sliki 14.

```
(SELECT DISTINCT
       [Final].[EXTERNAL TERM DATE],
           CASE
               WHEN  [Final].[EXTERNAL TERM DATE] <  CAST(DATEADD(day, 8, GETDATE()) as DATE) THEN 'NEXT 7 DAYS'
               ELSE  'NEXT 14 DAYS'
               END AS [CATEGORY]
               ,CTE.[TEAM]
               ,[CTE].[TERM RESPONSIBLE]
               FROM [dbo].[Final]
CROSS JOIN
       (SELECT DISTINCT
           [TEAM]
           ,[TERM RESPONSIBLE]
            FROM [dbo].[Final])AS [CTE]

               WHERE [Final].[EXTERNAL TERM DATE] >  CAST(DATEADD(day, -1, GETDATE()) as DATE)
               AND   [Final].[EXTERNAL TERM DATE] <  CAST(DATEADD(day, 15, GETDATE()) as DATE)) AS [CTE2]
```

Slika 14: Prikaz CROSS JOIN v tabeli Daily

Odločila sem se, da uporabim select distinct v ugnezdeni poizvedbi. Najprej sem s CASE določila, kako bodo datumi dodeljeni. Izbrala sem dve skupini, iz katerih se lahko dodelijo datumi, prva skupina se imenuje NEXT 7 DAYS, druga pa NEXT 14 DAYS. Ti dve skupini mi je uspelo ustvariti z uporabo funkcije DATEADD, kjer sem današnjemu datumu dodala osem dni. Ker DATEADD uporablja tip datetime, sem dodala tudi funkcijo CAST, ki pretvori vrednost (katere koli vrste) v določen podatkovni tip, v mojem primeru je to tip date. Če je na primer danes 1. december, bo funkcija datumu dodala osem dni in

datum bo postal 9. december. Če se je datum, ki je shranjen v EXTERNAL TERM DATE, zgodil pred 9. decembrom, bo dodeljen prvi skupini, sicer pa bo dodeljen drugi skupini. Nato sem v stavku WHERE poskrbela, da je bil izbran razpon datumov. Brez tega dela bi bili obe skupini iz prejšnjega dela napačni.

Z isto funkcijo kot prej sem izbrala razpon med dnevom pred današnjim dnem in 15 dnevi od današnjega dneva. Tako sem dobila razpon 16 dni. Če bi bil danes 1. december, bi se razpon začel pri 30. novembru in trajal do 16. decembra. Ker pa znak večje/manjše ne vključuje znaka za enakost, je dejanski razpon od 1. do 15. decembra, kar je točno 14 dni. Rezultat, ki ga dobim po izvedbi kode, je prikazan na sliki 15.

	EXTERNAL TERM DATE	CATEGORY	TEAM	TERM RESPONSIBLE
40	2022-12-08	NEXT 7 DAYS	MINAJ	MU
41	2022-12-08	NEXT 7 DAYS	JANKOVIC	YDS
42	2022-12-08	NEXT 7 DAYS	OTHERS	FAT
43	2022-12-08	NEXT 7 DAYS	OTHERS	BLU
44	2022-12-08	NEXT 7 DAYS	OTHERS	HAR
45	2022-12-08	NEXT 7 DAYS	OTHERS	SON
46	2022-12-08	NEXT 7 DAYS	JANKOVIC	MSU
47	2022-12-09	NEXT 14 DA...	WALKER	CL
48	2022-12-09	NEXT 14 DA...	GATES	YDSK
49	2022-12-09	NEXT 14 DA...	GATES	AB

Slika 15: Rezultat izvedbe ugnezdene zanke

Po uspešnem ustvarjanju tabele *CTE2* z uporabo CROSS JOIN sem se osredotočila na ustvarjanje zadnje tabele z imenom *CTE3*. Tabela *CTE3* je ustvarjena z levim združevanjem. Tako so končni stolpci v tabeli *CTE3* naslednji: EXTERNAL TERM DATE, CATEGORY, TEAM, TERM RESPONSIBLE, NUMBER OF DEADLINES in NUMBER OF DEADLINES2.

Nato sem v stolpcu EXTERNAL TERM DATE uporabila isto merilo za razpon datumov kot v predhodnem koraku. Za stolpca NUMBER OF DEADLINES in število NUMBER OF DEADLINES2 sem uporabila funkcijo COUNT(*), ki ob uporabi GROUP BY vrne število vrstic v vsaki skupini. Na sliki 16 je prikazano, kako sta tabeli *CTE2* in *Final* združeni z uporabo stolpcev EXTERNAL TERM DATE, TEAM, TERM RESPONSIBLE in CATEGORY.

```
LEFT JOIN
            (SELECT [EXTERNAL TERM DATE]
                ,[CATEGORY]
                ,TEAM
                ,[TERM RESPONSIBLE]
                ,COUNT(*) AS [NUMBER OF DEADLINES]
                ,COUNT(*) AS [NUMBER OF DEADLINES2]

            FROM [dbo].[Final]
            WHERE [Final].[EXTERNAL TERM DATE] >  CAST(DATEADD(day, -1, GETDATE()) as DATE)
            AND   [Final].[EXTERNAL TERM DATE] <  CAST(DATEADD(day, 15, GETDATE()) as DATE)
            GROUP BY [EXTERNAL TERM DATE],[CATEGORY],TEAM,[TERM RESPONSIBLE])
    AS [CTE3]
    ON CTE3.[EXTERNAL TERM DATE] = CTE2.[EXTERNAL TERM DATE] AND
       CTE3.CATEGORY = CTE2.CATEGORY AND
       CTE3.TEAM = CTE2.TEAM AND
       CTE3.[TERM RESPONSIBLE] = CTE2.[TERM RESPONSIBLE]
```

Slika 16: Left join tabele Daily

Stolpci, ki sem jih prevzela iz CTE2, so EXTERNAL TERM DATE, CATEGORY, TEAM in TERM RESPONSIBLE. Stolpca, ki sem ju vzela iz CTE3, sta NUMBER OF DEADLINES in NUMBER OF DEADLINES2. Dodati sem morala dva stolpca, ki sem ju oblikovala z uporabo stolpca EXTERNAL TERM DATE iz CTE2. Njihova uporaba je bila pomembna, ker sem potrebovala izpis datuma in številko dneva v tednu. S tem sem omogočila boljši pregled podatkov.

Za stolpec, ki je potreboval izpis datuma, sem uporabila ukaz FORMAT, ki je vzel celoten stolpec EXTERNAL TERM DATE in ga oblikoval v datum v skladu z danim formatom, ki je v mojem primeru »dd/MM/yy«, kot je vidno na sliki 17. Za naslednji stolpec, kjer je bilo treba izpisati dan v tednu, sem uporabila funkcijo DATEPART, da sem iz datuma v stolpcu EXTERNAL TERM DATE izpisala dan v tednu. To vrednost sem nato oblikovala kot niz v želeni obliki, ki je v mojem primeru »dan v tednu tega dneva zapisa«. To mi je omogočilo prikaz dneva v tednu za vsak zapis v zbirki podatkov.

```
FORMAT(CTE2.[EXTERNAL TERM DATE], 'dd/MM/yyyy') AS [NewDate],
DATEPART(w, CTE2.[EXTERNAL TERM DATE]) AS [Day_Of_Week]
```

Slika 17: Oblikovanje stolpcev

V tretjem in zadnjem postopku projekta je sledilo dodajanje v tabelo Daily. Dodala sem še dve kategoriji, ki se nanašata na stolpec NewDate. To sem morala storiti tako, da sem skrbno pregledala podatke, ki so že bili v tabeli, izbrala prave kategorije za nove podatke in nato vstavila nove podatke v tabelo z uporabo poizvedb SQL. Potrebnega je bilo veliko truda in pozornosti do podrobnosti, vendar mi je na koncu uspelo tabelo pravilno posodobiti z novimi podatki. To mi je pomagalo bolje razumeti trende in vzorce v mojih

podatkih ter mi omogočilo izboljšati natančnost mojih raziskav in poročanja. Pomagala sem si z novo tabelo, imenovano *Daily Tmp.*

Da bi kategorije v preglednici Daily še bolj izpopolnila, sem morala dodati dve kategoriji: »more than 14 days« in »in the past«. To sem storila tako, da sem ponovila enake korake, kot sem jih uporabila za ustvarjanje kategorij »next 7 days« in »next 14 days«. Za kategorijo »more than 14 days« sem uporabila stavek WHERE, s katerim sem izbrala samo tiste vrstice, v katerih je bil EXTERNAL TERM DATE večji od 14 dni od trenutnega datuma. Za kategorijo »in the past« sem uporabila stavek WHERE, da sem izbrala samo tiste vrstice, v katerih je EXTERNAL TERM DATE manjši od trenutnega datuma. To mi je omogočilo natančno in zanesljivo kategoriziranje podatkov v tabeli glede na relativne datume EXTERNAL TERM DATE.

Poleg tega sem stolpec NewDate uporabila za pretvorbo EXTERNAL TERM DATE v obliko, ki je Evropejcem bolj poznana, kjer je najprej prikazan dan, sledita mu mesec in nato leto, kot je razvidno s slike 18. To je olajšalo branje in razumevanje datumov v tabeli ter pomagalo zagotoviti, da so podatki natančno predstavljeni in organizirani. Na splošno so mi ti koraki omogočili učinkovito posodobitev tabele *Daily* z novimi kategorijami in zagotovili, da so bili podatki natančno in zanesljivo organizirani in kategorizirani.

	EXTERNAL TERM DATE	NewDate	Day_Of_Week
36	2022-10-22	22/10/2022	7
37	2022-10-22	22/10/2022	7
38	2022-10-22	22/10/2022	7
39	2022-10-23	23/10/2022	1
40	2022-10-23	23/10/2022	1
41	2022-10-23	23/10/2022	1
42	2022-10-23	23/10/2022	1
43	2022-10-24	24/10/2022	2
44	2022-10-24	24/10/2022	2
45	2022-10-24	24/10/2022	2
46	2022-10-24	24/10/2022	2
47	2022-10-24	24/10/2022	2
48	2022-10-24	24/10/2022	2
49	2022-10-24	24/10/2022	2
50	2022-10-24	24/10/2022	2
51	2022-10-24	24/10/2022	2

Slika 18: Prikaz poizvedbe novih stolpcev

3 POWER BI

Power BI je aplikacija v oblaku za vizualizacijo podatkov in poslovno obveščanje, ki ustvarja interaktivne nadzorne plošče in poročila BI z uporabo podatkov iz številnih virov. Namizni program Power BI, ki temelji na SaaS (programska oprema kot storitev), in mobilne aplikacije Power BI, ki so na voljo za različne platforme, so le nekateri od

programov in storitev, ki jih ponuja paket Power BI. Te storitve pogosto uporabljajo poslovni uporabniki za uporabo podatkov in ustvarjanje poročil BI [10].

Sistemi za poslovno obveščanje običajno temeljijo na podatkovnih skladiščih. Vendar pa lahko poslovni uporabniki sami ustvarjajo poročila z orodji, kot je Power BI Desktop, brez večje pomoči strokovnjakov za BI, ki so dostopnejše kot kdaj koli prej. Zaradi tega se je pojavil samopostrežni pristop BI, ki je še posebej primeren za mala in srednje velika podjetja, ki še nimajo podatkovnega skladišča ali vzpostavljene strategije BI [1].

Ustvarjanje samopostrežne rešitve BI in dostop do podatkov v podatkovnem skladišču sta primerna načina uporabe orodja Microsoft Power BI. S čarovnikom za uvoz podatkov Power BI omogoča preprosto povezavo z najpogostejšimi viri podatkov, ki se danes uporabljajo pri razvoju rešitev BI, ne da bi za to potrebovali skriptno ali programersko znanje [15].

3.1 Implementacija

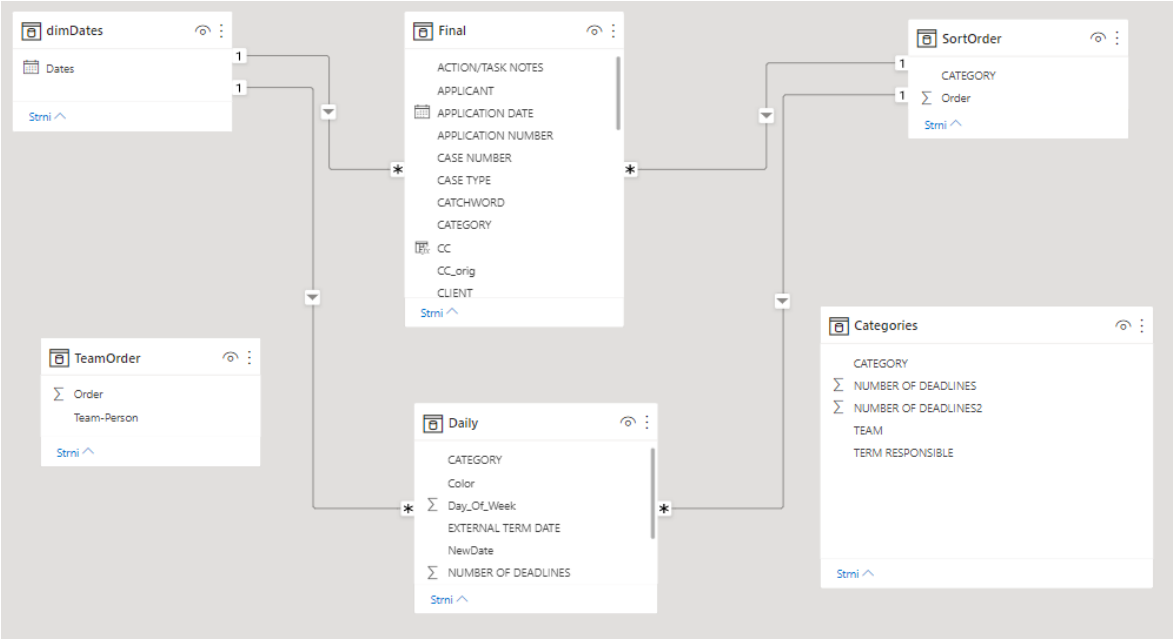
Ustvarila sem eno poročilo na treh straneh. Preglednica o rokih v končnem podatkovnem modelu, ki je prikazan na sliki 19, ima stanje normalizacije in denormalizacije. Relacije so bile ustvarjene po uvozu podatkovnih tabel s funkcijo »upravljanje relacij« v Power BI, kot je razvidno na sliki 19.

Aktiven	Od: tabele (stolpec)	Do: tabele (stolpec)
☑	Daily (CATEGORY)	SortOrder (CATEGORY)
☑	Daily (EXTERNAL TERM DATE)	dimDates (Dates)
☑	Final (CATEGORY)	SortOrder (CATEGORY)
☑	Final (EXTERNAL TERM DATE)	dimDates (Dates)

Slika 19: Relacije v podatkovnem modelu

V tem projektu so uporabljene tabele v podatkovni zbirki SQL. Njihova struktura in povezave so ročno narisane in prikazane na sliki 20. Ustvarjanje in posodabljanje tabel je dodeljeno samo osnovni razvojni ekipi, tako da ustvarjanje nove tabele v tem projektu ni dovoljeno. Tabela *DimDates* je povezana z dvema tabelama, in sicer s *Final* in *Daily* prek stolpca *EXTERNAL TERM DATE,* kot je razvidno na sliki 20. Tabela *SortOrder* pa je povezana prek stolpca *CATEGORY* s tabelo *Final* in *Daily*. To so tudi edine povezave, ki

jih podatkovni model potrebuje. Vrste razmerja glede na številnost so vse ena proti mnogo, saj je lahko en zapis v tabeli povezan z enim ali več zapisi v drugi tabeli.



Slika 20: Preglednica v podatkovnem modelu dokončane rešitve BI

Datumske tabele v Power BI so tabele, ki vključujejo samo podatke, povezane z datumi. To je tipična dimenzijska tabela, ki je lahko uporabljena za vrednotenje podatkov na podlagi datumov in služi kot referenca za datume v modelu. Še posebej so koristne pri pripravi poročil, ki potrebujejo natančne podatke o datumu, in pri izvajanju izračunov časovne inteligence [6]. Prav zaradi tega sem ustvarila tabelo z imenom *DimDates*, ki jo omenim v tretjem koraku poglavja Izvedba. Dimenzijska tabela *DimDates*, kjer vsaka vrstica označuje drugi datum, je uporabljena pri proučevanju obnašanja in vzorcev podatkov skozi čas, zato je tudi povezana z dvema glavnima tabelama.

Na novo narejena tabela *SortOrder* služi namenu razvrščanja podatkov, ki sem jih predhodno dobila iz stolpca *CATEGORY*. Tabela ločuje pet kategorij, in sicer: IN THE PAST(MORE THAN 1 WEEK), IN THE PAST(LESS THAN 1 WEEK), NEXT 14 DAYS, NEXT 7 DAYS, MORE THAN 14 DAYS, in jim pripiše nove vrednosti, katerih namen je dodeliti število stolpcu in ga nato razvrstiti. Ena tabela je verjetno boljša, kadar so informacije ena-za-eno, saj se tako zmanjša število povezav, ki jih mora zbirka podatkov opraviti za pridobitev rezultatov.

Podatke je treba razdeliti v več preglednic, če gre za podatke ena proti več, saj podvojeni podatki zapravljajo prostor za shranjevanje, predpomnilnik in otežujejo vzdrževanje podatkovne zbirke. To je razlog, da sem se odločila dodati novo tabelo *SortOrder*.

3.2 Poročila

Poročilo Power BI, imenovano »Poročilo o rokih«, prikazuje vse zunanje ukrepe za eno leto vnaprej in vse ukrepe iz preteklosti (omejitve do obdobja po letu 2000). Poročilo zajema vse zunanje ukrepe, ki so bili izvedeni v zadnjem letu, in tiste, ki so načrtovani. Z dostopom do teh podatkov poročilo uporabnikom omogoča, da proučijo in analizirajo prejšnje ukrepe ter tako opazijo trende in vzorce ter načrtujejo in se pripravijo na prihodnje ukrepe.

Za podjetja, ki morajo nadzorovati in spremljati zunanje ukrepe, je »Poročilo o rokih« uporabno orodje. Poročilo pomaga uporabnikom pri ohranjanju organizacije in obvladovanju njihovih nalog, saj ponuja jasen in jedrnat pregled vseh relevantnih operacij. Poročilo je preprosto za uporabo in navigacijo, uporabniki pa lahko spreminjajo filtre in nastavitve tako, da ustrezajo njihovim zahtevam.

»Poročilo o rokih« je na splošno močan in uporaben instrument za upravljanje zunanjih dejavnosti. Uporabnikom pomaga, da ostanejo organizirani, pripravljeni in obveščeni, saj podaja temeljit povzetek preteklih in prihodnjih dejanj.

Poročilo Power BI je razdeljeno na tri strani: pregled rokov, poročilo o rokih in nadzorna plošča rokov. Pregled rokov prikazuje osnovne informacije o izbranih rokih in statistične podatke o izbranih rokih, kot je na primer časovna kategorija. Za to stran so filtri omejeni in se delijo na tri dele, in sicer na odgovorno osebo, odgovornega odvetnika in časovno kategorijo. Pregled rokov je namenjen ekipam, ki želijo pregled nad številom rokov glede na posamezno ekipo. Naslednja stran je poročilo o rokih, ki je tudi glavna stran poročila. Prikazane so podrobne informacije za vsak izbrani rok. Stran ponuja veliko več možnosti izpolnjevanja kot prva stran, tako da lahko vsak uporabnik vidi podrobne informacije. Nato sledi nadzorna plošča rokov, ki prikazuje število rokov v preteklosti, v prihodnosti in število rokov za vsak dan od danes naprej. Na tem mestu je prikazana dodatna preglednica, ki je bila izdelana dodatno.

Nabor podatkov (poročilo) se samodejno osveži vsak delovni dan ob 9.30 in ponovno ob 10.30. To se izvede z uporabo funkcije On-premises data getaway.

Na nadzorni plošči so vsi atributi prikazani po abecednem vrstnem redu poleg gumbov za ustvarjanje vizualizacij. V osnovi obstajata dva načina za ustvarjanje vizualizacije: uporabnik lahko izbere atribute, ki ga zanimajo, in nato klikne gumb za vizualizacijo ali pa ustvari prazno vizualizacijo tako, da klikne gumb in nato povleče atribute v želene kanale.

Pri gradnji podatkovnega modela so bila med razvojem vključena vsa polja, za katera obstaja vsaj majhna verjetnost, da so pomembna.

4 ZAKLJUČEK

Za uspešno konkuriranje v industriji IT in drugje je vse bolj pomembno, da imamo na voljo ustrezne podatke in zbiramo potrebne informacije. Za izpolnitev teh zahtev je treba podatke pretvoriti v obliko, ki najbolj ustreza analitikom in na kateri je mogoče v čim krajšem času izvesti številne študije. Ker so na voljo številna orodja ETL, je treba izbrati najboljše za določeno podjetje. Glavni dosežek projekta, ki je opisan v zaključni nalogi, je uporaba tehnologije Power BI za izdelavo poročila, ki bo uporabnikom pomagalo pri učinkovitejšem iskanju in preverjanju informacij o izdelkih ob uporabi različnih orodij. Prvič sem se učila v skupini za poslovno upravljanje tako velike organizacije in za ponazoritev svojega dela uporabila resničen primer.

Program je bil do zaključka pisanja naloge uporabljen več kot stokrat. Pri tem projektu so težave vključevale ugotavljanje podatkovne strukture podatkovne zbirke SQL in uporabo programa Power BI. Poročilo v aplikaciji Power BI prikaže prave informacije, če je model ustvarjen pravilno pri delu, opravljenem v tej zaključni nalogi. Funkcionalno je delovalo pravilno in izpolnjevalo zahteve.

Zaključimo lahko, da morajo sodobna podjetja, ki so pri analizi in odločanju odvisna od podatkov, optimizirati postopek ETL. Podjetja lahko povečajo uspešnost in učinkovitost svojih operacij upravljanja podatkov z izvajanjem strategij, obravnavanih v tej zaključni nalogi, kot sta paralelizacija in preverjanje kakovosti podatkov, v praksi.

Poleg tega lahko podjetja z rednim spremljanjem in optimizacijo procesa ETL zagotovijo, da so njihovi podatki pravilni in posodobljeni, kar jim omogoča ohranjanje konkurenčnosti v današnjem hitro razvijajočem se poslovnem okolju. Na splošno ima optimizacija ETL veliko prednosti in podjetja, ki lahko uspešno izboljšajo svoj postopek ETL, bodo v sedanjem podatkovno usmerjenem gospodarstvu lahko uspešno poslovala.

Glavni dosežek projekta, ki je opisan v zaključni nalogi, je uporaba tehnologije SQL za izdelavo poročila, ki bo uporabnikom pomagalo pri učinkovitejšem iskanju in preverjanju informacij o dokumentih ob uporabi različnih orodij za poslovno obveščanje. Poslovna analiza ima pri programskem inženirstvu vedno ključno vlogo, čeprav je ta naloga tesneje povezana s poslovno analizo kot s programskim inženirstvom. Prvič sem se učila v skupini za poslovno upravljanje tako velike organizacije in za ponazoritev svojega dela uporabila resničen primer.
Orodje je narejeno tako, da ga je mogoče razširiti, tako da lahko skupina tudi v prihodnosti dodaja nove funkcije trenutni rešitvi.

5 LITERATURA IN VIRI

[1] A. Tolvanen, *Developing an efficient business intelligence solution for day-to-day supply chain management: Case pulp and paper industry workshop,* School of Engineering Science, Tietotekniikka, 2019.

[2] H. Varshney, *What is Data Staging Area?,* https://hevodata.com/learn/data-staging-area/. (Datum ogleda: 26. 10. 2022.)

[3] IBM, *What is ETL?,* https://www.ibm.com/cloud/learn/etl. (Datum ogleda: 26. 10. 2022.)

[4] J. A. Sharp, *Data Flow Computing: Theory and Practice,* Intellect Books, 1992.

[5] J. Melton in A. R. Simon, *Understanding The New SQL: A Complete Guide,* Morgan Kaufmann Publishers, San Francisco, 1993.

[6] K. Simlleoluea, *How to create data tables in Power BI Tutorial,* https://www.datacamp.com/tutorial/how-to-create-date-tables-in-power-bi-tutorial. (Datum ogleda: 26. 10. 2022.)

[7] Microsoft, *Control Flow,* https://learn.microsoft.com/en-us/sql/integration-services/control-flow/control-flow?view=sql-server-ver16. (Datum ogleda: 26. 10. 2022.)

[8] Microsoft, *Download SQL Server Management Studio (SSMS),* https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio. (Datum ogleda: 26. 10. 2022.)

[9] Microsoft, *Integration Services (SSIS) Projects and Solutions,* https://learn.microsoft.com/en-us/sql/integration-services/integration-services-ssis-projects-and-solutions?view=sql-server-ver16. (Datum ogleda: 26. 10. 2022.)

[10] Microsoft, *Power BI,* https://powerbi.microsoft.com/en-us/desktop/. (Datum ogleda: 26. 10. 2022.)

[11] Microsoft, *SSMS introduction,* https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studiossms?view=sql-server-2017. (Datum ogleda: 26. 10. 2022.)

[12] Microsoft, *Visual studio,* https://visualstudio.microsoft.com/. (Datum ogleda: 26. 10. 2022.)

[13] P. Loshin, *Structured Query Language (SQL),* https://www.techtarget.com/searchdatamanagement/definition/SQL. (Datum ogleda: 26. 10. 2022.)

[14] S. Chaudhuri in U. Dayal, An overview of data warehousing and OLAP technology, *ACM Sigmod record* 26 (1997), 65–74.

[15] T. Lachev, *Applied Microsoft Power BI : bring your data to life!,* Prologika Press, Atlanta, 2022.

[16] T. Mitchell, M. Masson, A. Leonard, J. Moss in M. Ufford, *SQL Server Integration Services Design Patterns,* Apress, 2014.

[17] Tutorials Point, *SQL Left Join,* https://www.tutorialspoint.com/sql/sql-left-joins.htm. (Datum ogleda: 26. 10. 2022.)

[18] W3Schools, *SQL introduction,* https://www.w3schools.com/sql/sql_intro.asp. (Datum ogleda: 26. 10. 2022.)

[19] Z. El Akkaoui, E. Zimányi, J. N. Mazón in J. Trujillo, A BPMN-based design and maintenance framework for ETL processes, *International Journal of Data Warehousing and Mining (IJDWM)* 9 (2013), 46–72.