5. melléklet

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR

SZAKDOLGOZAT

OE-NIK	Hallgató neve:	**Lakó Boldizsár**
2022	Hallgató törzskönyvi száma:	**T/006280/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai Rendszerek Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Lakó Boldizsár**
Törzskönyvi száma: T/006280/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

Nemek szerinti bér és juttatás analízis céges környezetben, webes nyomonkövetéssel
Gender based wage and benefit analysis in corporate environment with online tracking

Intézményi konzulens: Dr. Nagy Enikő
Külső konzulens:

Beadási határidő: 2021. december 15.

A záróvizsga tárgyai: Számítógép architektúrák
Big Data és üzleti intelligencia specializáció

A feladat

Tervezzen és valósítson meg egy olyan online felületet, amely lehetővé teszi a cég adatbázisának bővítését és módosítását. Továbbá legyen lehetőség dinamikusan frissülő riportok megtekintésére. Kutassa fel és vizsgálja meg a riportokhoz szükséges adattárház elkészítéséhez az ETL eszközöket. Ismertesse az online felülethez használandó modellt. Valósítsa meg a szükséges funkciókat és tervezze meg az ehhez szükséges adattárházsémát.

A dolgozatnak tartalmaznia kell:

- online felület kialakításához szükséges modell kiválasztásának indoklását,
- online felület megtervezését és megalkotását,
- az adatforrás réteg meghatározását, illetve a rendszerek és a kívánt adatok/táblák kiválasztását,
- ETL réteg részletes megtervezését és megalkotását,
- adattárház tervezését és megvalósítását,
- az adatpiac számára a megfelelő információk kiválasztását és megfelelő formára alakítását.

Ph.

.....
Dr. Fleiner Rita
intézetigazgató

A szakdolgozat elévülésének határideje: **2023. december 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2022.05.02.

.....
hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
9. sz. melléklet

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Neptun kód: Tagozat:

Lakó Boldizsár..... [redacted] ..... nappali.....

Telefon: Levelezési cím (pl: lakcím):

[redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

Nemek szerinti bér és juttatás analízis céges környezetben, webes nyomon követéssel.

Szakdolgozat / Diplomamunka[2] címe angolul:

Gender based wage and benefit analysis in corporate environment with online tracking.

Intézményi konzulens: Külső konzulens:

Dr. Nagy Enikő..... .....

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2021.02.15	Szakdolgozat címének és tartalmának megbeszélése	
2.	2021.03.19	Szakdolgozat felépítésének megbeszélése	
3.	2021.05.06	Szakdolgozat 1 tartalmának áttekintése	
4.	2021.05.11	Szakdolgozat formai követelményei	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2021. 05. 11.

intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
9. sz. melléklet

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Neptun kód: Tagozat:

Lakó Boldizsár..... [redacted] ..... nappali.....

Telefon: Levelezési cím (pl: lakcím):

[redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

Nemek szerinti bér és juttatás analízis céges környezetben, webes nyomon követéssel. ....

Szakdolgozat / Diplomamunka[2] címe angolul:

Gender based wage and benefit analysis in corporate environment with online tracking.

Intézményi konzulens: Külső konzulens:

Dr. Nagy Enikő..... .....

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2022.03.09	Szakdolgozat adattárházának átbeszélése	
2.	2022.04.07	Webes felület tervezése	
3.	2022.04.20	Vizualizációk megbeszélése	
4.	2022.05.02	Szakdolgozat áttekintése	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2022.05.02.....

.....
intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!

TARTALOMJEGYZÉK

1. BEVEZETÉS

A nemek közötti fizetés és juttatási szakadék az elmúlt fél évszázad egyik kiemelkedő kérdése. Elemzők mégse találnak konkrét bizonyítékot arra, hogy ez a szakadék csökkenne bármilyen formában. Ennek az indokai összetettek viszont az adatok azt mutatják, hogy családi háttértől és tanulmányi háttértől függetlenül a nők valamilyen módon kevesebbet keresnek.

A szakdolgozatom egy képzeletbeli cég számára egy interneten keresztül elérhető platform létrehozása, amelyen a cég dolgozói a cég adatbázisába tudnak új adatokat felvinni, törölni és módosítani. Ebből az adatbázisból ETL[1] folyamaton keresztül adattárház kiépítése melyben az adatokat visszamenőleg is tároljuk, nem csak a pillanatnyi állapotokat. Az adattárházban lévő adatokból végül dinamikusan frissülő riportokat fogok készíteni, ami az internetes platformon megtekinthető.

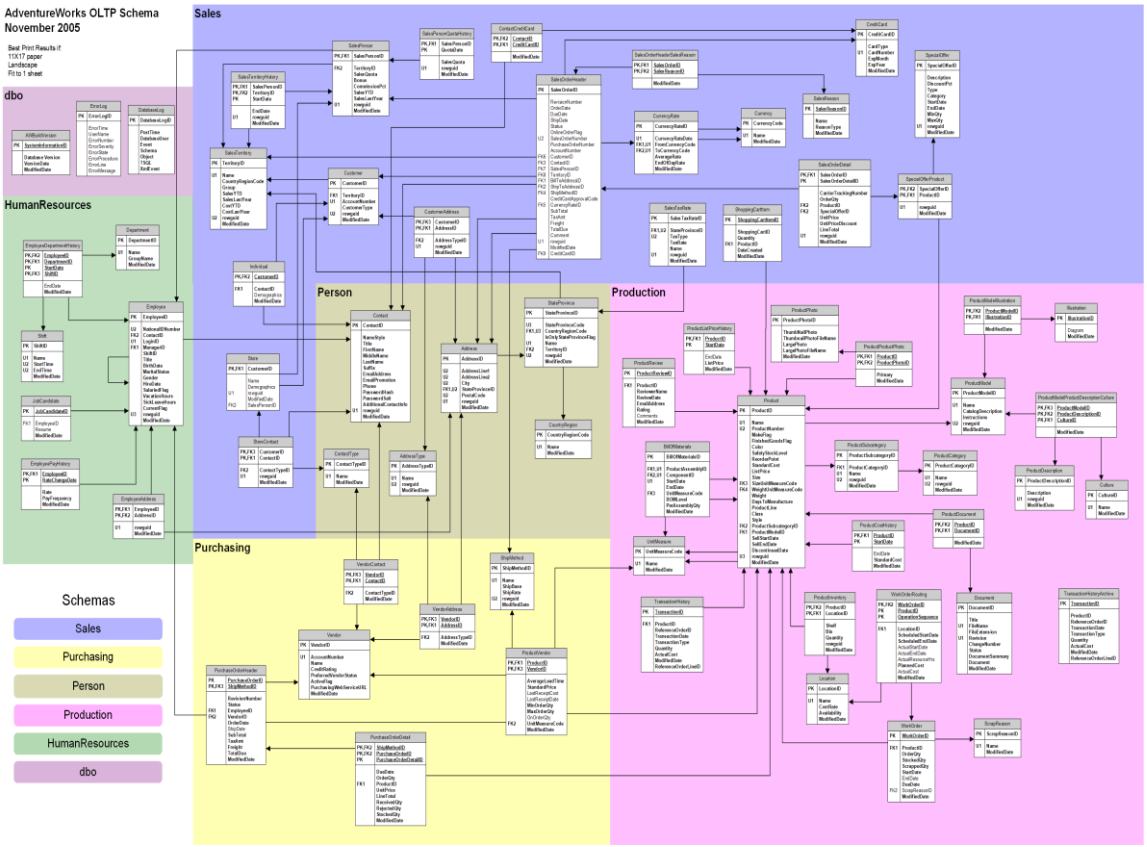
Ezekből a riportokból kiderül, hogy a cégnek milyen téren kellene változtatni ahhoz, hogy a nemek közötti bármiféle különbséget megszüntessék. Ehhez vizsgálnom kell, hogy az adott alkalmazott mióta van a cégnél, van-e gyereke, valamint a nőknél számításba kell venni a terhességi szabadságot is. Elvárás ezenkívül az adattárház rétegeinek olyan felépítése, mely a későbbiekben egyszerűen bővíthető és módosítható. Az adatbázisból az adattárházba való átmozgatáskor ügyelni kell arra, hogy a régi adatokat ne írjam felül, hanem mentsem el a változtatásokat is, hogy lássuk a cég fejlődését.

A szakdolgozatom során bemutatom az online felület kialakításához szükséges modell kiválasztásának indoklását. Az online felület megtervezését és megalkotásának menetét, az adatforrás réteg meghatározását, illetve a rendszerek és a kívánt adatok/táblák kiválasztását is. Ezen felül ki fogok térni az ETL réteg megtervezésére, az adattárház megalkotására és a fejlesztés során felmerülő opcionális/kialakult problémákra.

[1] ETL: Extract Transform Load

2. FORRÁS ADATBÁZIS KIVÁLASZTÁSA

A feladat elvégzéséhez elsősorban egy forrásadatbázisra van szükségem. A céget megvalósító adatbázis kiválasztásakor ügyelnem kell arra, hogy a valóságnak jól megfelelő adatbázis legyen. Ezenkívül az adatbázisnak tartalmaznia kell információkat az ott dolgozó személyek fizetéséről, nemükről, szabadságaik napjairól és családi hátterükről. A forrásadatbázisnak egy nyílt, szabadon elérhető adatbázist kerestem. Végül egy nem létező bicikli kereskedő cég adatbázisára esett a választásom, mivel ezt már régebben is használtam és tudom, hogy a benne lévő adatok megfelelnek a rá vonatkozó feltételeknek.



2.1. ábra Adatbázis E/K[2] diagram [14]

Mint a kép is mutatja (2.1 ábra) az adatbázisban sok számomra nem fontos tábla is található. A későbbiekben el kell döntenem, hogy mely táblákra és adatokra van szükségem a feladat megvalósításához. A hangsúly a személy (Person), Emberi erőforrások (HumanResources) és az Eladások (Sales) részeken van. Az internetes portálon keresztül egyelőre csak azokat a táblákat fogom felvinni, amelyek az adattárház számára is fontosak.

[2] E/K: Egyed kapcsolat diagramm

2.1. SZÜKSÉGES ÁTALAKÍTÁSOK AZ ADATHALMAZON

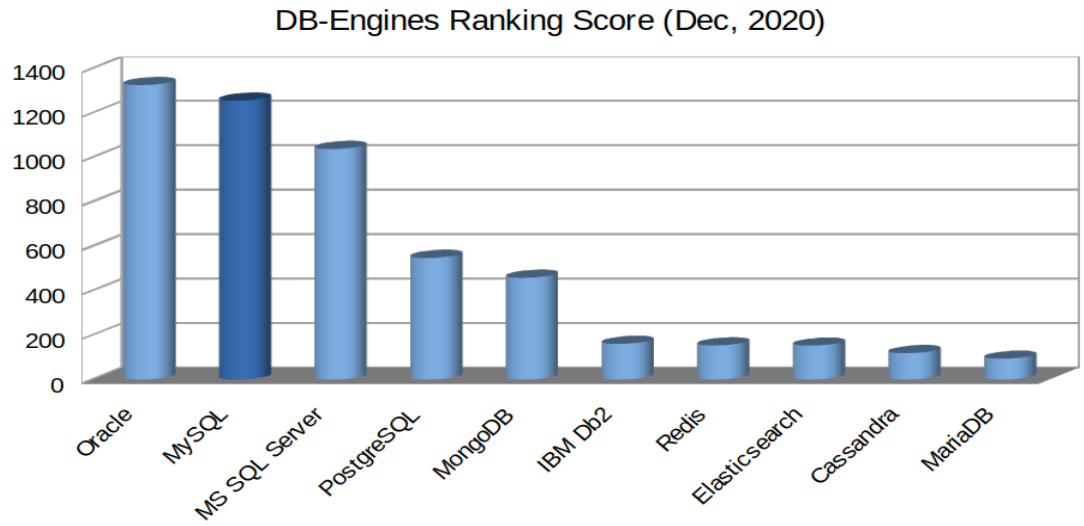
Az adathalmazon több átalakítást is eszközölnöm kell, hogy a riport készítés számára minél több lehetséges adat álljon a rendelkezésre. Ezenkívül a fizetések dollárban vannak megadva, valamint nem egységes fizetést kapnak, hanem különböző negyedévekre leosztva. Ezt az ETL folyamán belül standardizálni kell.

3. ADATBÁZIS-KEZELŐ RENDSZER KIVÁLASZTÁSA

Ahhoz, hogy az adatokat eltároljam, szükség lesz egy adatbázis-kezelő rendszerre. Mivel a forrás adatbázisom egy relációs adatbázis, ezért lehetőleg az adatbázis-kezelő is legyen relációs.

3.1. LEHETŐSÉGEK

Adatbázis-kezelő rendszerekből a három piacvezető relációs adatbázis-kezelő rendszer között kellett döntenem, hogy melyik lenne a megfelelőbb. A Microsoft SQL Szerver után a használati értékben egy viszonylag nagy ugrás figyelhető meg (3.1 ábra). Ez annak tudható be, hogy a PostgreSQL már nem egy szimpla relációs, hanem egy objektum alapú relációs adatbázis kezelő rendszer. Ki van bővítve extra funkciókkal, viszont nekem elegendő egy egyszerű, gyors és megbízható relációs adatbázis-kezelő rendszer. [8][27]



3.1. ábra Adatbázis kezelő rendszerek ranglistája 2020 decemberében [8]

3.1.1. ORACLE

A világranglista első helyén álló relációs adatbázis-kezelő rendszere az Oracle, ami az árazáson is meglátszik.

Jellemzői:

- Írási és olvasási sebesség nagy méretű adatokon (1m+ rekord) a leggyorsabb.
- Támogat AIX, HP-UX, Linux, OS X, Solaris, Windows, z/OS operációs rendszereket.
- .Net alkalmazás esetén beépített csatlakozási lehetőséggel bír. [8][9][28]

3.1.2. MICROSOFT SQL SZERVER

A Microsoft SQL Szerver a Microsoft vezető relációs adatbázis kezelő rendszere.

Jellemzői:

- A világranglistán a 3. helyen áll.
- Első kiadása 1989-ben volt, de azóta is rendszeresen fejlesztik, a jelenlegi legújabb kiadása 2019 novemberében jött ki.
- C++ programozási nyelven íródott.
- Natívan fut Linuxon és Windowson is.
- C# alkalmazás esetén itt is van lehetőség beépített csatlakozási lehetőségre.
- Olvasási sebessége kimagaslóan gyors, amit az adattárház folyamatos frissítésekor előnyt jelent. [8][9][29]

3.1.3. MY SQL

A My SQL egy Oracle által fejlesztett adatbázis-kezelő rendszer.

Jellemzői:

- Nyílt forráskódú.
- Az Oraclehöz képest minden téren lassabb, viszont ez az árazásban is meglátszik.
- Rendelkezik ingyenes community verzióval is.
- Támogatja a Windows és Linux operációs rendszereket.
- C# alkalmazáshoz való csatlakozáshoz egy plusz könyvtárra van szükségünk ahhoz, hogy minden beépített funkciót elérjünk. [8][9]

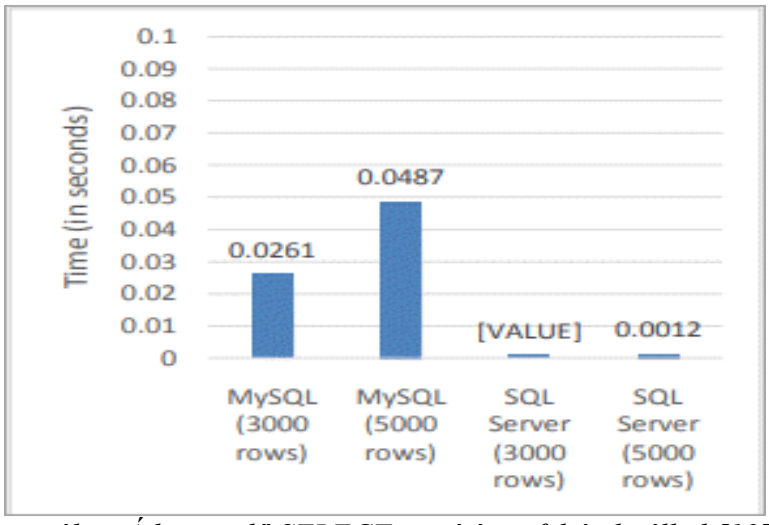
Név	Microsoft SQL Szerver	My SQL	Oracle
Leírás	Microsoft vezető relációs adatbázis-kezelő rendszere	Széleskörben elterjedt nyílt forráskódú adatbázis-kezelő rendszer	Széleskörben elterjedt adatbázis-kezelő rendszer
Elsődleges adatbázis modell	Relációs adatbázis	Relációs adatbázis	Relációs adatbázis
Másodlagos adatbázis modell	Dokumentum tároló Gráf adatbázis Téradatbázis	Dokumentum tároló Téradatbázis	Dokumentum tároló Gráf adatbázis RDF tároló Téradatbázis
Adatbázis motor rangja	#3 Pontozás:992.66	#2 Pontozás: 1236.38	#1 Pontozás: 1269.94

Weboldal	microsoft.com/en-us/sql-server	mysql.com	oracle.com/database
Dokumentáció	docs.microsoft.com/en-US/sql/sql-server	dev.mysql.com/doc	dosc.oracle.com/en/database
Fejlesztő	Microsoft	Oracle	Oracle
Kiadás dátuma	1989	1995	1980
Legújabb kiadás	SQL Server 2019	2021 április	2019 február
Licensz	kereskedelmi, de van ingyenes community verziója	Nyílt forrás kódú	Kereskedelmi

3.1. táblázat: Adatbázis-kezelő rendszerek összehasonlítása [8]

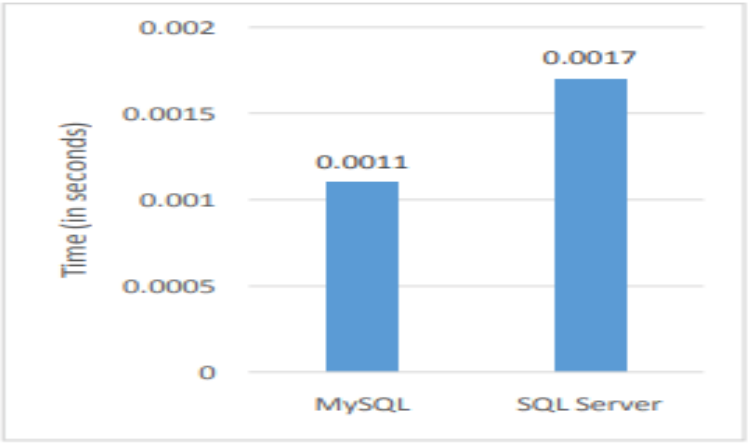
3.2. VÁLASZTÁS ÉS INDOKLÁS

Az Oracle lenne a legmegfelelőbb választás a sebessége és az egyszerű kapcsolódási lehetőség miatt, viszont ez nem rendelkezik ingyenes community verzióval, így ezt a lehetőséget el kell vetnem. A továbbiakban az MS SQL szerver és a MySQL szerver között kell választanom.



3.2. ábra Átlagos idő SELECT utasításra feltétel nélkül [10]

Az alábbi ábrán látható (3.2 ábra) sebesség különbsége hasonló arányban van jelen „join” használata esetében is.



3.3. ábra Átlagos idő 100 INSERT utasításra [10]

Mint a fenti ábrákból (3.2. és 3.3 ábra) is látszik a rekord beszúráson kívül minden téren gyorsabb az SQL szerver, mint a MySQL. Ezenkívül az alkalmazást C#-ban Visual Studio segítségével fogom elkészíteni, hasonlóan az ETL folyamathoz. Mivel a minta cégem egy bicikliket forgalmazó cég, így gyakori a „beszúrás" művelet az adatbázisba. Viszont az adattárház gyakori frissítése miatt gyakran fogunk „select" utasításokat is meghívni. Az „integrated-stack" elvét követve választottam adatbázis-kezelő rendszert, miszerint jobb akár a gyengébben teljesítő komponensekből összerakni a rendszert, ha tudjuk, hogy a kiválasztott komponensek garantáltan tudnak együtt dolgozni és könnyen megoldható az adatok átjuttatása egyik szoftverről a másikra. A Microsoft SQL szervernek van beépített kapcsolódási lehetősége .Net alkalmazásokhoz, valamint az ETL folyamatokat SSIS-en keresztül fogom futtatni, ami szintén egy Microsoft termék, így ez a legmegfelelőbb választás.[9][10][29]

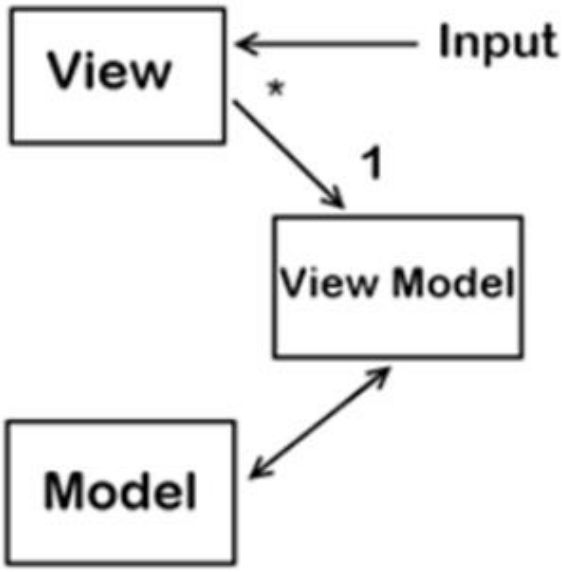
4. WEBES FELÜLET LÉTREHOZÁSÁHOZ ALKALMAZANDÓ MODELL KIVÁLASZTÁSA

Az alábbi fejezet a [13] forrás alapján készült.

A webes felület elkészítéséhez, valamilyen tervezési mintára lesz szükség, ami alapján az alkalmazás készülni fog. Ennek a célja, hogy a kódunk a jövőben könnyen módosítható és bővíthető legyen. Ha új funkciókra lesz szükségünk vagy új logikát akarunk mögé vinni, akkor könnyen módosíthassunk és ne egy úgynevezett spagetti kódot kelljen karban tartani. A modellek segítségével szét tudjuk szedni a program kódunkat kisebb egységekre és nem egy nagy egybefolyó kód lesz a végtermék. A modellen belüli elnevezések (például view, az az nézet) segítenek elkülöníteni a felelősségi köröket. Ez a jövőben segít a tesztelésben, fejlesztésben és az esetlegesen felmerülő hibák megtalálásában.

4.1. MODELLEK BEMUTATÁSA

4.1.1. MVVM (MODEL-VIEW-VIEWMODEL)



4.1. ábra MVVM modell ábrája [13]

- Főbb tulajdonságok:
  - GUI keretrendszerekben használt (WPF, UWP).
  - A felhasználói bemenetek a View-n belüli vezérlők eseményein keresztül jutnak el a View Model-hez, majd innen a Modelhez
  - Data-binding van a View és a ViewModel között.
  - Az adatmódosítások eseményeken keresztül mennek végbe.
  - Az adat módosítás közvetetten történjen, ezzel elkerülhető a két irányú függés.

- A modell ebben az esetben az üzleti logikát foglalja magában.
- MVVM tipikusan asztali alkalmazásokban használt, ahol az adatkötés XAML és INotifyPropertyChanged interfésszel történik.
- Ha a View-Model-ben szeretnénk módosítást végezni, akkor a View-Model egy observer pattern-t követ.

- Filozófia:

A View Model osztály a UI[3]-on keresztül megjeleníthető és bekérhető adatok attribútumokra való leképezéséért felelős.

- Egy publikus attribútum a ViewModel-ben megegyezik egy publikus attribútummal valamelyik vezérlőben.
- Egy megjelenített adat megegyezik egy publikus tulajdonsággal a ViewModel-ben.
- Egy bekért adat megegyezik egy publikus tulajdonsággal a VM osztályban.

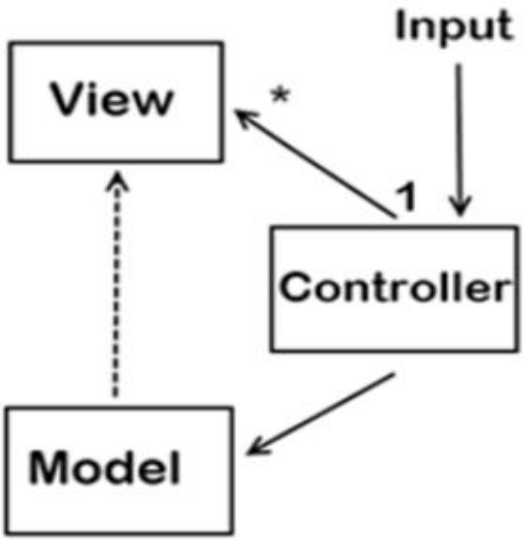
- Előnyök:
  - Az üzleti logika szét van választva a felhasználói felülettől.
  - Könnyen karbantartható.
  - Könnyen tesztelhető.
  - Egység tesztelés a View Model és a Model rétegre elvégezhető a View referenciája nélkül

- Hátrányok:
  - Sok kóddal jár, amit karban kell tartani.
  - Egyszerű felhasználói felület esetén egy egyszerűbb modell jobban alkalmazható
  - View és a View Model közötti kapcsolatot nem lehet egyszerűsíteni.

Összességében egy jól használható modell (4.1. ábra), ami az erős adatkötésre alapszik, ami a XAML-ben elérhető. A modellben a View Model felelős a Model-ben lévő adatok property[4]-vé képzéséért, amit majd a View-ban megtudunk jeleníteni.

[3] UI: felhasználói interfész.
[4] property: tulajdonság, kódban egy adattag például egy integer

4.1.2. MVC (MODEL-VIEW-CONTROLLER)

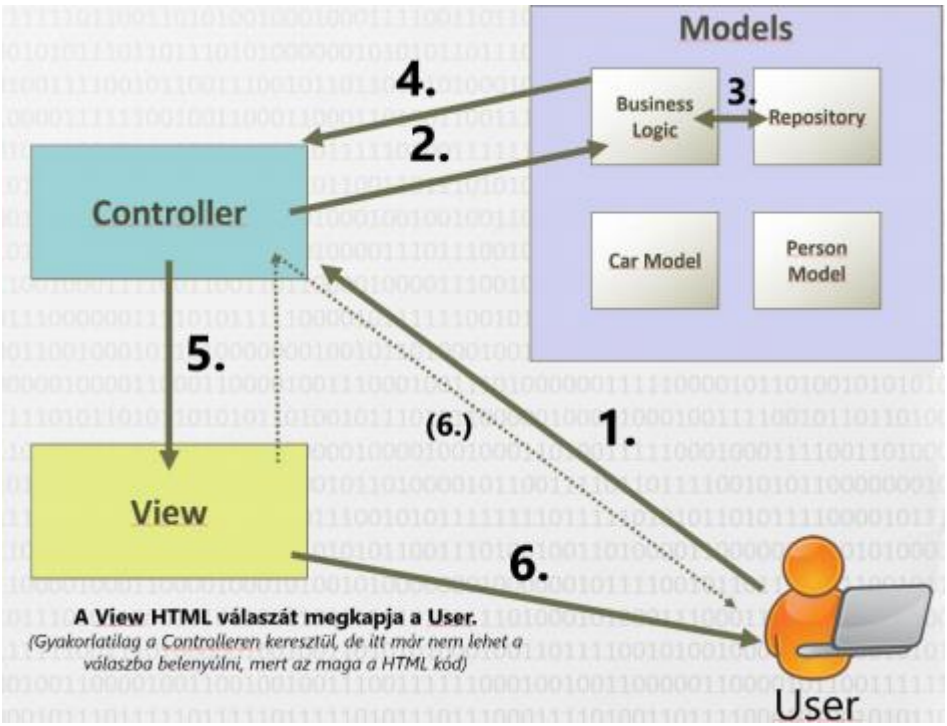


4.2. ábra MVVM modell ábrája [13]

Az MVC modellt (4.2. ábra) tipikusan webes keretrendszerek használják. Például az ASP .Net is, amit én is használni fogok a fejlesztés során.

- Főbb tulajdonságok:
  - A user a Controllernek küld kéréseket a böngészőn keresztül.
  - A Controller View-t választ és feltölti a modell-t adatokkal.
  - Események helyett validációt használ.
  - A View a modell adatait használja.
  - A Controller a View-t adja vissza a Usernak (webes felületnek)
  - Az üzleti logika a Controllerben vagy mögötte található
  - A Modelben csak Data bag típusú adatok fognak szerepelni, amik csak a nézethez szükséges adatokat tartalmazzák.
  - A Controller mögött több réteg is található
  - Támogatja a tesztvezérelt fejlesztést.
  - Teljesen szabadon állítható HTML és URL lehetőségeket kínál.

- **MVC munkamenet bemutatása**



4.3. ábra MVC workflow ábrája [13]

1. A felhasználó a webes alkalmazásban navigál. A routing alapján a kérése valamelyik Controller valamelyik metódusához érkezik meg.
2. Az adott Action (metódus), ha szükséges, a Domain Model réteg osztályaihoz fordul. (tipikusan BusinessLogic és adatmodellek)
3. Az üzleti logika az adatbázis adatait használja.
4. Az üzleti logika az összeállított választ (pl. Alkalmazottak listája, Fizetési statisztikák stb.) továbbítja a Controllernek.
5. A Controller továbbítja az üzleti logika válaszát a View-nak
6. A View HTML válaszát megkapja a felhasználó.

- **Előnyök**

  - A különböző komponensek fejlesztése párhuzamos módon megoldható.
  - Egyszerű modell a széttagolt egységeknek köszönhetően.
  - A webes kéréseket egyetlen controller vezérli.
  - A legjobb modell tesztvezérelt fejlesztés eléréséhez.
  - Web applikációkhoz tökéletesen passzol, mivel nagyobb csapatok tudnak dolgozni kisebb egységeken külön-külön.
  - Minden objektum külön-külön tesztelhető, nincs függés közöttük.
  - Az MVC megengedi a logikai csoportosításokat az azonos tevékenységekhez (action).

- **Hátrányok**
    - Üzleti logika egybemosódik a felhasználói felület kódjával.
    - Nehezen újra felhasználható és implementálható tesztelés.
    - Minél több a kód annál nehezebben áttekinthető az egy darab controller miatt.
    - A párhuzamos programozás megvalósításához több programozóra van szükség.
    - Több technologiában való jártasság erősen ajánlott.

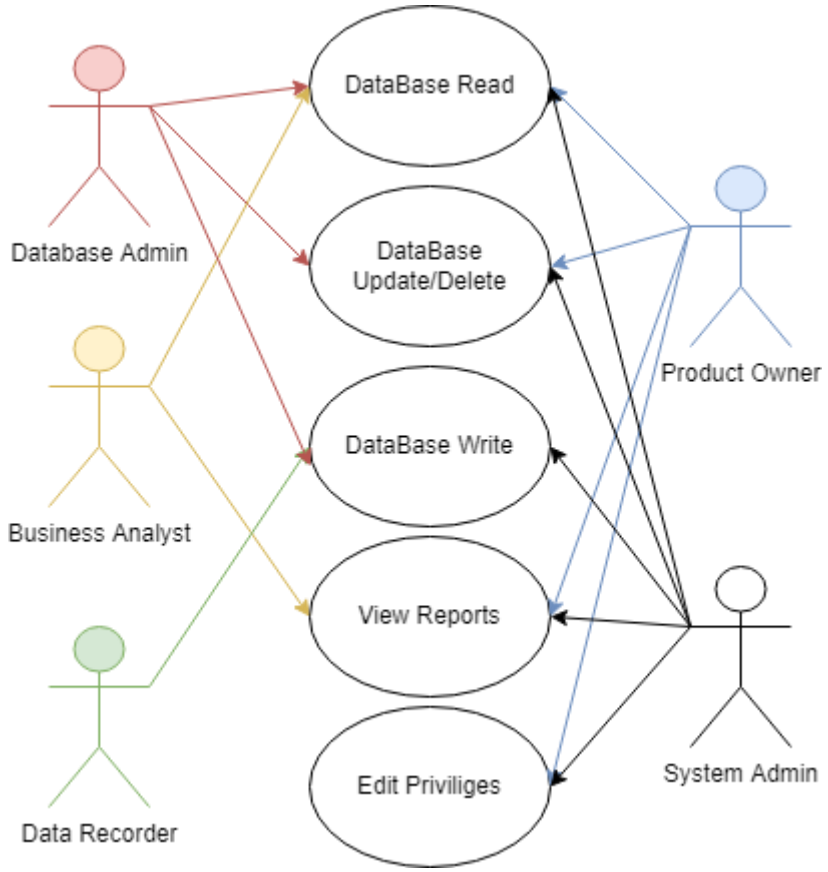
MVVM	MVC
A view a belépő pont az alkalmazásba.	A controller a belépő pont az alkalmazásba
„Egy a többhöz” kapcsolat a View és a View-Model között.	Egy a többhöz kapcsolat a controller és a view között.
View tartalmaz hivatkozást a View-Model-re.	A view nem tartalmaz hivatkozást a View-Model-re.
MVVM egy viszonylak új modell.	MVC egy idősebb modell.
A hibakeresés nehézkes lesz, ha komplex adatkötéseink vannak.	Nehezen olvasható, módosítható és egységtesztelhető.
Könnyen elkülöníthető egység tesztek és a kódolás cél vezérelt.	Az MVC komponensek tesztehetőek a felhasználó tudta nélkül.

4.1. táblázat MVVM és MVC összehasonlítása [13]

4.2. VÁLASZTÁS INDOKLÁSA

A web alapú alkalmazások többsége MVC vagy MVVM alapú modellekre épülnek. A .Net támogatja az MVC modellt, viszont a MVVM csak közvetett módon oldható meg. Korábbi fejlesztéseim során gyakran MVC modellt használtam, ezért ez közelebb áll hozzám. Az előbb felsorolt indokok miatt a feladatot is MVC modell alapján fogom elkészíteni.

5. JOGOSULTSÁGOK ÉS SZEREPKÖRÖK



5.1. ábra use-case diagram

Az internetes portálra minden alkalmazott a saját felhasználó név/jelszó párossal fog tudni belépni és csak azokhoz az opciókhoz fog hozzáférni, amihez a beosztása tartozik (5.1. ábra). Ezzel elkerülendő, hogy például egy elemző hozzáférjen a jogosultság módosítása menüponthoz és olyan dolgot módosítson, amihez nem ért vagy nem is szabadna hozzányúlnia.

5.1. SZEREPKÖRÖK

- **Database Admin**

Adatbázis adminisztrátornak joga van az adatbázis írásához és olvasásához is. Ezenkívül az adat rögzítők által rosszul felvitt adatok módosítására és törlésére is van lehetősége.

- **Business Analyst**

Az elemzőknek joguk van az adatbázisból való olvasásra, valamint a riportok megtekintésére.

- **Data Recorder**

Az adat rögzítő szerepkört betöltő embereknek csak az adatbázisba való írásra van engedélyük.

- **Product Owner**

A termék tulajdonosnak joga van az adatbázisból való olvasásra, törlésre és módosítására, valamint a riportok megtekintésére és a jogosultságok módosítására.

- **System Administrator**

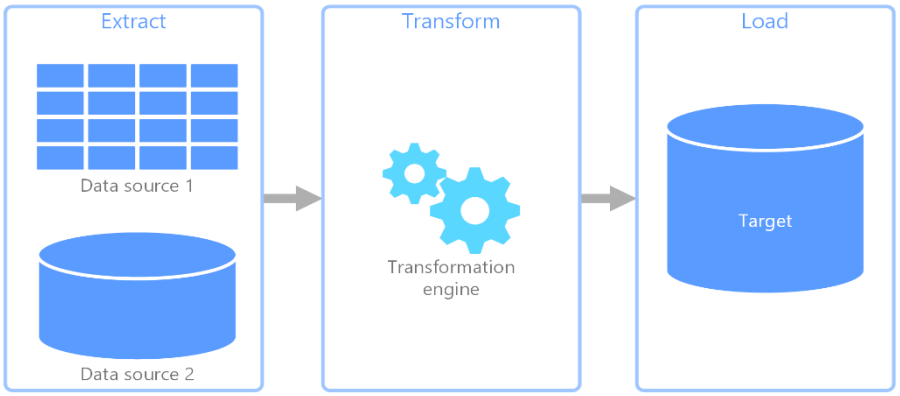
A rendszer adminisztrátornak mindenhez joga van. Joga van az adatbázisba való írásra és abból olvasásra, valamint az adatbázis módosítására. Ezenkívül hozzáfér a riportokhoz és a jogosultságokhoz.

6. ETL ESZKÖZ KIVÁLASZTÁSA

Az alábbi fejezet a [2][3][4][30] források alapján készült.

Az ETL folyamat alatt a forrásadatbázisból az adatok adattárházba való átmozgatását értjük. A folyamat közben az adatokon módosításokat és sokféle transzformációt végezhetünk.

6.1. ISMERTETÉS



6.1. ábra: ETL folyamat ábrázolása [1]

Az ETL (6.1. ábra) eszköz kiválasztásakor az SSIS[5] és a PDI[6] került a lehetséges eszközök listájára. Az SSIS a Microsoft tulajdonában lévő eszköz, ami szoros kapcsolatban áll az SQL szerverrel. Ezzel szemben a PDI egy nyílt forráskódú software. PDI-nak saját felhasználói felülete van és Java-ban íródott, igy bármely operációs rendszeren elfut. Ezzel szemben SSIS Visual Studioban lett megtervezve, igy csak Windows operációs rendszeren fut. A betöltési részben mind a kettő lehetőség ugyanazokkal a funkciókkal bír, nagyjából semmiféle jelentős eltérés nincs közöttük.

6.2. TRANSFORM LEHETŐSÉGEK

PDI-ban több előre beépített átalakítási lehetőség van és ezek sokkal lényegre törőbbek az SSIS-hez képest. Ezzel szemben az SSIS kevesebb előre létrehozott átalakítási lehetőséggel rendelkezik, viszont egyedi, személyre szabott lehetőségek egyszerűen kódolhatók Visual Studioban.

6.3. ÁRAZÁS

A PDI community verziója ingyenes viszont az enterprise verziója viszonylag drága. Ezzel szemben az SSIS ára felhasználási módtól függ, de van elérhető ingyenes community verziója is.

[5]SSIS: SQL Server Integration Services
[6]PDI: Pentaho Data Integration

6.4. ÖSSZEFOGLALÁS ÉS DÖNTÉS

Mind az SSIS és a PDI is egy megfelelő megoldás lenne az adattárház felépítésére. A PDI több transzformálási lehetőséggel és több webes kapcsolati lehetőséggel rendelkezik. Használható bármely operációs rendszeren és önmagában olcsóbb is lenne, mint az SSIS. Én az Integrated Stack elvét követve az SSIS használata mellett döntöttem mivel nekem elégnek bizonyulnak az SSIS community verziója által nyújtott lehetőségek. Ezenkívül az adatbázisomat Microsoft SQL szerveren fogom futtatni, amivel könnyedén tud együtt működni. Továbbá a feladatom megoldásához nem fogok más operációs rendszert használni a Windowson kívül, valamint a fejlesztési lehetőségek a Visual Studiora épülés miatt könnyen megoldhatók.

7. VIZUALIZÁCIÓS ESZKÖZ (BI[7]) KIVÁLASZTÁSA

Adattárházakból az adatok vizualizációjára a gyakorlatban általában valamilyen vizualizációs eszközt alkalmazunk az adatok megjelenítésére.

A vizualizációs eszköz kiválasztásakor döntenem kellett, hogy a Microsoft tulajdonában lévő Power BI-t, a piacvezető Tableau-t vagy pedig saját fejlesztésű riportokat jelenítek meg.

A Tableau alapvetően adatelemző cégeknek készült, ezzel szemben a Power BI általános használatra lett tervezve, ebből kifolyólag Tableau felhasználói felülete nehezebben értelmezhető, tanulható. Tableau nagyobb volumenű adatok feldolgozására képes a Power BI-al szemben, viszont ez az árazásban is meglátszik. Power BI Microsoftos rendszerekre épül (Azure,SQL,Excel). [5]

Saját fejlesztésű riportok esetében több fejlesztésre lesz szükségünk. Ezenkívül kevesebb előre elkészített vizualizációs lehetőséget kínál számunkra, mintha egy direkt erre a célra fejlesztett eszközt használnánk.

Tableau-nak nincs community verziója, ezért ezt a lehetőséget el kell vetnem. A Power BI a feladatnak minden téren megfelelne, viszont ahhoz, hogy a riportokat ne manuálisan kelljen kiimportálni, hanem legyen lehetőségem dinamikusan újra generálni, ahhoz előkell fizetni a használatára. Ezért végül saját fejlesztésű riportok készítése mellett döntöttem. Ehhez ingyenes nyílt forráskódú grafikon készítő könyvtárat fogok felhasználni. [6]

[7] BI: Üzleti Logika

8. RIPORT KÉSZÍTÉS FELÉ TÁMASZTOTT ELVÁRÁSOK

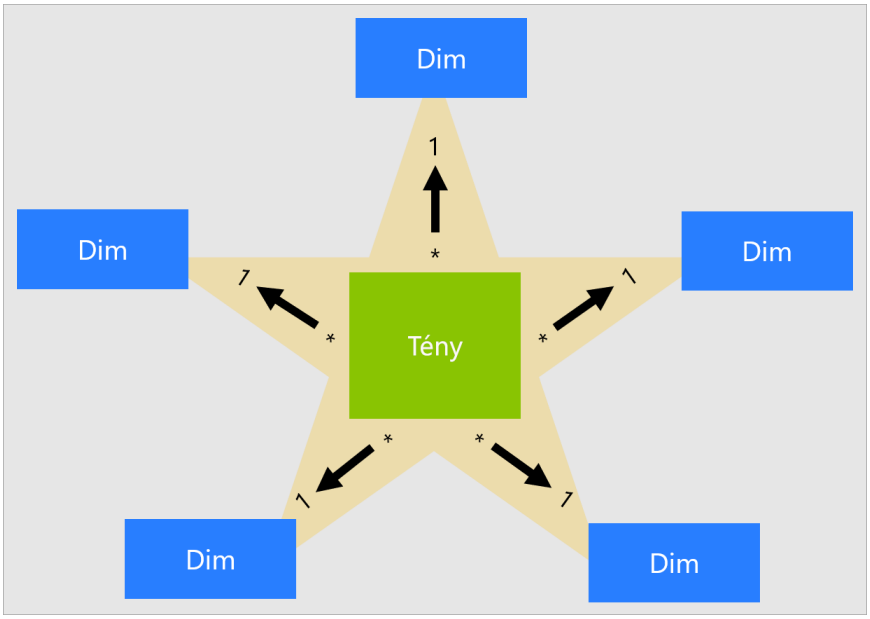
A riportok készítése során figyelmet kell fordítanom arra, hogy a forrás adatbázis folyamatosan változik és ezt a változást a riportokban is figyelemmel akarom kísérni. Ezért el kell dönteni, hogy milyen időközönként frissüljenek. Az ETL folyamat futtatását napi rendszerességre fogom állítani így minden nap végén az adott változásokat elmentem az adattárházba is. Célszerű lenne a riportokat is ilyen időközökként frissíteni és akkor a rendszer, amikor nincs terhelés alatt könnyen végre tudja hajtani ezt a feladatot.

A riportoknak könnyen áttekinthetőnek, valamint látványosnak kell lenniük, hogy a vezetőség is megértse. A riportok csak az alkalmazottakra fognak fókuszálni azon belül is a nemek közötti különbségekre. A cél az, hogy a vezetőség lássa, hogy az eszközölt változtatások mennyire váltak be és milyen további intézkedéseket lenne eszközölni. Ezért a riportokban a kiugró és eltérő értékeket valahogy ki kell emelnem, hogy a problémák könnyen észrevehetőek legyenek.

9. ADATTÁRHÁZ SÉMA KIVÁLASZTÁSA

Az adattárház séma az adatbázis logikai leírása. Tartalmazza az összes rekord típust, rekordjainak a nevét, valamint a leírását, beleértve az összes kapcsolódó adatelemet és aggregátumokat. A sima adatbázisokhoz hasonlóan az adattárháznak is egy előre definiált sémát kell követnie a gyorsabb futás és könnyebb karbantarthatóság végett. Az adattárházak általában csillag, hópehely vagy galaxis sémát követnek. Ez a három a legelterjedtebb adattárház séma.

9.1. CSILLAGSÉMA



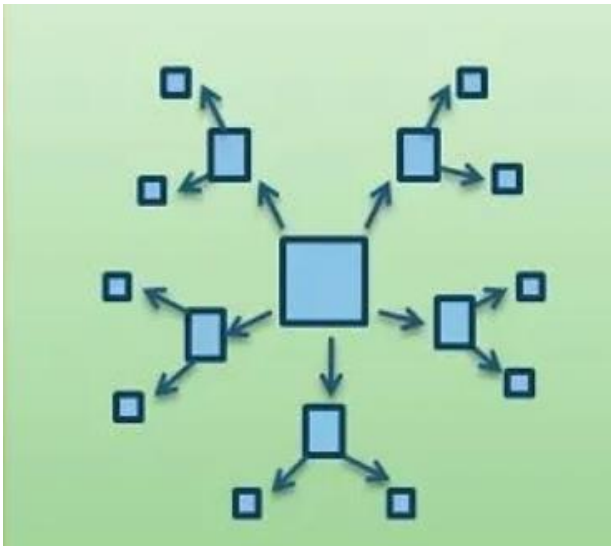
9.1. ábra csillagséma ábrája [25]

A sémák közül a legegyszerűbb, viszont a legszélesebb körben használt séma a csillagséma. A csillagséma egy ténytáblából áll, amely tetszőleges számú dimenzió táblára hivatkozik. Ezért is nevezik csillagsémának mivel a modell alakja hasonlít egy csillagra (9.1. ábra).

A ténytáblák általában metrikákat tárolnak bizonyos eseményekre. Általában számszerű értékeket tárolunk benne, valamint az idegen kulcsokat a dimenzió táblákra.

Dimenzió táblák a ténytáblában lévő entitások kontextusát és kiegészítő adatait tárolja. A legáltalánosabb példa a dátum dimenzió. [25]

9.2. HÓPEHELY SÉMA



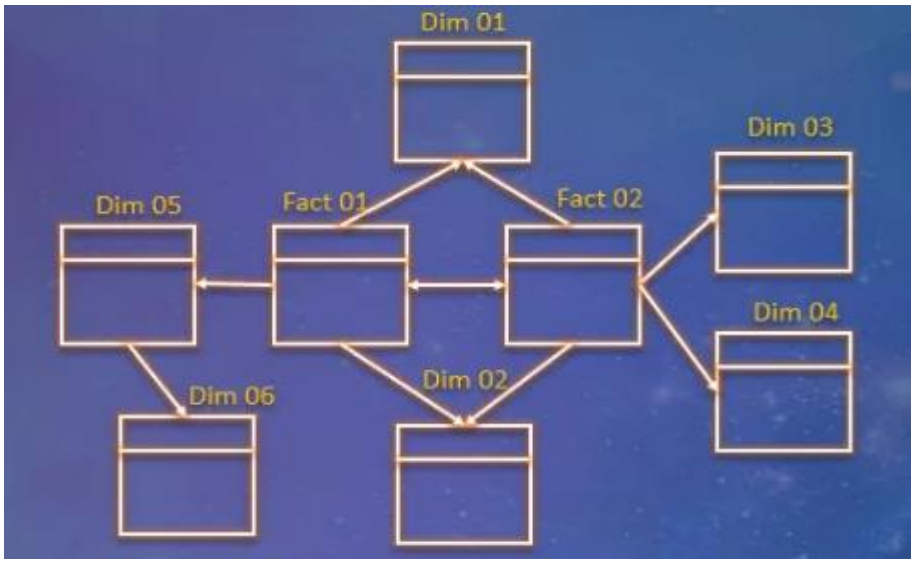
9.2. ábra hópehely séma ábrája [26]

A „Snowflaking” azaz „hópelyhesítés” egy eljárás melyben a csillagsémában lévő dimenzió táblákat normalizáljuk. A normalizálás hatására az adatokat több táblára bontjuk és ezért a modell egy hópehelyre fog hasonlítani (9.2. ábra).

Előnye a csillagsémával szemben az, hogy normalizáltuk a dimenzió táblákat ezért a redundanciát csökkentettük, azaz kevesebb tárhelyre lesz szükségünk, valamint még egyszerűbbé teszi az adattárház karbantartását.

Hátránya az, hogy a lekérdezések több joinnal valósulnak meg ezért lassabbak lesznek. [26]

9.3. GALAXIS SÉMA



9.3. *ábra galaxis séma ábrája [26]*

A galaxis séma másnéven ténykonstellációs séma eltér az előző sémáktól. Itt már több mint egy ténytáblánk van (9.3. ábra), amelyek a dimenzió táblákkal állnak kapcsolatban. A dimenzió táblák ebben az esetben is normalizálva vannak a hópehely sémához hasonlóan. A séma felfogható úgy, mint több csillagséma összekötve és normalizálva. Ezt a modellt abban az esetben szokták használni, ha több ténytáblára van szükség. Ez a mi esetünkben nem áll fent, mivel mi a nemek alapján szeretnénk felépíteni az adattárházat. [26]

9.4. DÖNTÉS

Mivel csak egy ténytábla lesz, ezért a galaxis séma nem célszerű. A hópehely séma előnye, hogy tárhelyet lehetne spórolni a teljesítmény rovására, viszont nekünk az lenne a cél, hogy egy egyszerű, de gyors modell alapján készítsük el az adattárházat, ezért a szakdolgozathoz csillagsémát fogok használni.

10. ÁLTALÁNOS KÖVETELMÉNY SPECIFIKÁCIÓ

Az első lépés a forrás adatbázis betöltése Microsoft SQL szerverre, mivel a forrás állomány csak egy .bak kiterjesztésű backup (biztonsági mentés) fájlként áll a rendelkezésemre.

Forrás-adatbázisból való adattárház kiépítését végzem el, ahol bizonyos attribútumokat visszamenőleg historikusan is tárolni kell. További feladatom még kiválasztani azokat a táblákat és azokat a mezőket, amelyekre a későbbiekben szükségem lesz és csak ezeket az adatokat kell majd tovább vinnem. Az adattárházból, egy kisebb, célorientált rész egy úgynevezett DataMart létrehozása, ahol el kell döntenem, hogy milyen sémát fogok alkalmazni. Ezt indokolnom is kell. Ha megvan, hogy milyen séma a legmegfelelőbb, akkor ebbe a sémába kell áttöltenem az adatokat az adattárházból.

Ezenfelül egy internetes portált kell még létrehoznom, ami több view/nézet-ből fog állni. Az első oldal mindenképpen egy bejelentkező felület lesz, ahol a dolgozók betudnak jelentkezni a saját felhasználó nevükkel és jelszavukkal. A bejelentkezés hatására a rendszernek tudnia kell, hogy milyen jogosultságokkal rendelkezik az adott dolgozó és az ehhez tartozó füleket érje csak el az online portálon. Például az elemzők ne tudjanak új felhasználókat felvenni. A felületnek összesen öt különböző fő funkciója lesz. Lesz egy, ahol a forrás adatbázis megtekintésére lesz lehetőség. Egy másikban pedig a forrásadatbázisban szereplő rekordokat tudjuk megtekinteni. Ezenkívül kell még egy, ahol a forrásadatbázisban szereplő rekordokat lehet módosítani. Az elemzők számára kell opciót biztositani arra, hogy az internetes portálon az aktuális adatokból riportokat tudjanak megtekinteni. A vezetőség és az adminisztrátorok számára kell egy felület, ahol a jogosultságokat tudják módosítani.

A cégnél nagyrészben magyar nyelvű dolgozók dolgoznak, viszont egy része nem tud magyarul csak angolul ezért a webes applikációnak tudnia kell támogatni az angol nyelvet.

11. FUNKCIONÁLIS KÖVETELMÉNY SPECIFIKÁCIÓ

11.1. BEJELENTKEZŐ FELÜLET ÉS JOGOSULTSÁGOK

A bejelentkező lapon három dologra lesz szükségünk. Egy mezőre, ahova a felhasználó a felhasználónevét írja. Egy másikra, ahova a jelszavát írja és egy gombra, ami hatására az illető be tud lépni, ha jó felhasználó/jelszó párost adott meg. Itt elvárás, hogy rossz jelszó, nem ismert felhasználónév vagy rossz kombináció hatására egy hibaüzenet formájába értesítsük a felhasználót a problémáról. Nem szükséges megmondani pontosan, hogy mi a hiba, ezért elég annyit közölni, hogy valami nem jó és próbálkozzon újra. Ha a felhasználó jó felhasználó és jelszó párost adott meg akkor az oldal irányítson tovább a következő üres felületre, ahol felül egy sávban jelenjenek meg az adott felhasználó által elérhető funkciók.

- Elemzők számára elérhető legyen: Adatbázis olvasása és riportok megtekintése.
- Adat rögzítők számára elérhető legyen: Adatbázis írása (új rekord felvétele).
- Adatbázis adminisztrátornak elérhető legyen: Adatbázis írása és olvasása, valamint az adatbázis frissítése.
- A rendszer adminisztrátor számára elérhető legyen: Adatbázis írása, olvasása, frissítése, riportok megtekintése, valamint a jogosultságok módosítása.
- A tulajdonos számára elérhető legyen: Az adatbázis olvasása, riportok megtekintése és a jogosultságok módosítása.

11.2. ADATBÁZIS OLVASÁSA A WEBES FELÜLETEN

A webes felületen legyen lehetőség a forrás adatbázis olvasására. Ha az arra jogosult dolgozó rákattint az adatbázis olvasása funkcióra, akkor egy új felület jelenjen meg ahol a dolgozó kitudja választani, hogy melyik táblát szeretné megtekinteni. A kiválasztás után az oldalon jelenítődnek meg az adott táblában szereplő rekordok táblázatos formában. Ha a táblában sok rekord található, akkor lehessen görgetni az oldalon.

11.3. ADATBÁZIS ÍRÁSA

A webes felületen legyen lehetőség a forrás adatbázisba új rekord felvételére. Ha az arra jogosult dolgozó rákattint az adatbázis írása funkcióra, akkor egy új felület jelenjen meg, ahol a dolgozó ki tudja választani, hogy melyik táblába szeretne írni. Ha kiválasztotta a táblát, akkor jelenjenek meg a táblában található oszlop nevek és alatta vagy mellette egy szövegdoboz, ahova a dolgozó betudja írni a felvinni kívánt adatokat. A rendszernek képesnek kell lennie felismerni, ha már létező elsődleges kulcsú adatot szeretnénk felvinni és ezt egy hibaüzenettel jeleznie kell a felhasználónak. Ha a felhasználó hibás adatot akarna felvinni az adatbázisba (például: dátum helyére nevet), akkor a rendszer ezt ne engedje és egy hibaüzenet formájában értesítse a felhasználót a hibáról. Ha a felhasználó jól írta be az adatokat, akkor legyen egy gomb, amivel a rekordot fel tudjuk

vinni az adatbázisba. Ha sikerült felvinni akkor a rendszer ezt egy üzenet formájában tudassa a felhasználóval.

11.4. ADATBÁZIS FRISSÍTÉSE

A webes felületen legyen lehetőség a forrás-adatbázisban rekord frissítésére/módosítására. Ha az arra jogosult dolgozó rákattint az adatbázis frissítése funkcióra, akkor egy új felület jelenjen meg, ahol a dolgozó ki tudja választani, hogy melyik táblában lévő adatot kívánja módosítani. Módosítás esetén jelenjen meg a rekordhoz tartozó attribútumok listája és mellettük egy szövegdoboz, ahova az új adatokat tudja majd a felhasználó beírni. Ha a felhasználó rossz típusú vagy hibás adatot adott meg, akkor a rendszer ezt legyen képes felismerni és a hibáról a felhasználót egy hibaüzenet formájában értesítse. Ha mindent jól adott meg, akkor legyen egy gomb, amivel a felhasználó a rekordot tudja frissíteni. A frissítés befejezése után a rendszer egy üzenet formájában értesítse a felhasználót a folyamat sikerességéről.

11.5. RIPORTOK

A webes felületen legyen lehetőség riportok megtekintésére. A riportokból három darab az elvárás. Egy, amiben látszik részlegekre, azon belül nemekre bontva a férfi és női dolgozók aránya. Egy másikban látszódjon összesítve, hogy a férfi és női dolgozók mennyi vakációt és mennyi szabadságot kapnak. A harmadikban pedig látszódjon a nemek fizetése részlegekre bontva. A felhasználónak a webes felületen legyen lehetősége grafikon formájában, valamint táblázatos formában való megtekintésre is. A riportokhoz szükséges adatok az ETL folyamat lefutása után automatikusan frissüljenek.

11.6. ADATTÁRHÁZ KIÉPÍTÉSE

A forrásadatbázis betöltése után egy adattárház kiépítése melyben a fontos attribútumok historikusan legyenek eltárolva (vakáció, fizetés, betegszabadság).

11.7. DATAMART KIÉPÍTÉSE

Az adattárházból egy általam kiválasztott, a feladatnak megfelelő sémába kell áttöltenem az adatokat. A DataMart kiépítésének a szempontja a gyors és könnyű riport készítés a cégen belüli nemek közötti különbségekről.

11.8. ETL FOLYAMAT ÉS ÜTEMEZÉS

A feladatom része még egy ETL folyamat megtervezése és kivitelezése is. Az ETL folyamatban a forrásadatbázisból kell az adatokat átmozgatni az előzőleg létrehozott adattárházba, majd az adattárházból a DataMartba. A folyamat legyen napi rendszereséggel ütemezve. A folyamatot ne manuálisan kelljen lefuttatni, hanem a rendszer automatikusan futtassa a nap végén.

11.9. FORRÁS ADATBÁZIS MÓDOSÍTÁSA

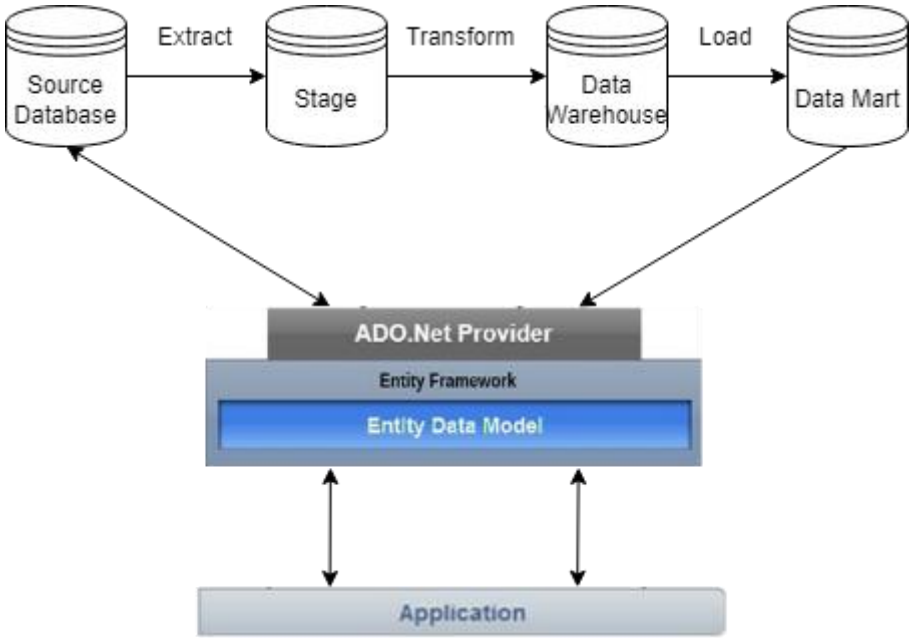
A forrásadatbázisban lévő adatokat kell olyan formára hoznom, hogy az jobban átlátható legyen. A fizetés idő az adatbázisban két különböző mezőből tevődik össze (fizetés, periódus). Ezt az adattárházba már úgy kell eltárolnom, hogy felaggregálom olyan módon, hogy csak egy egységes fizetés oszlop legyen.

11.10. TÖBB NYELV TÁMOGATÁSA

A dolgozóknak legyen lehetőségük a webes applikáción belül nyelvet váltani magyar és angol között. Ehhez legyen egy-egy gomb a menüpontok között. Ha a felhasználó rákattint erre a gombra, akkor az alkalmazás váltson nyelvet.

12. TERVEZÉS

12.1. RENDSZER TERV



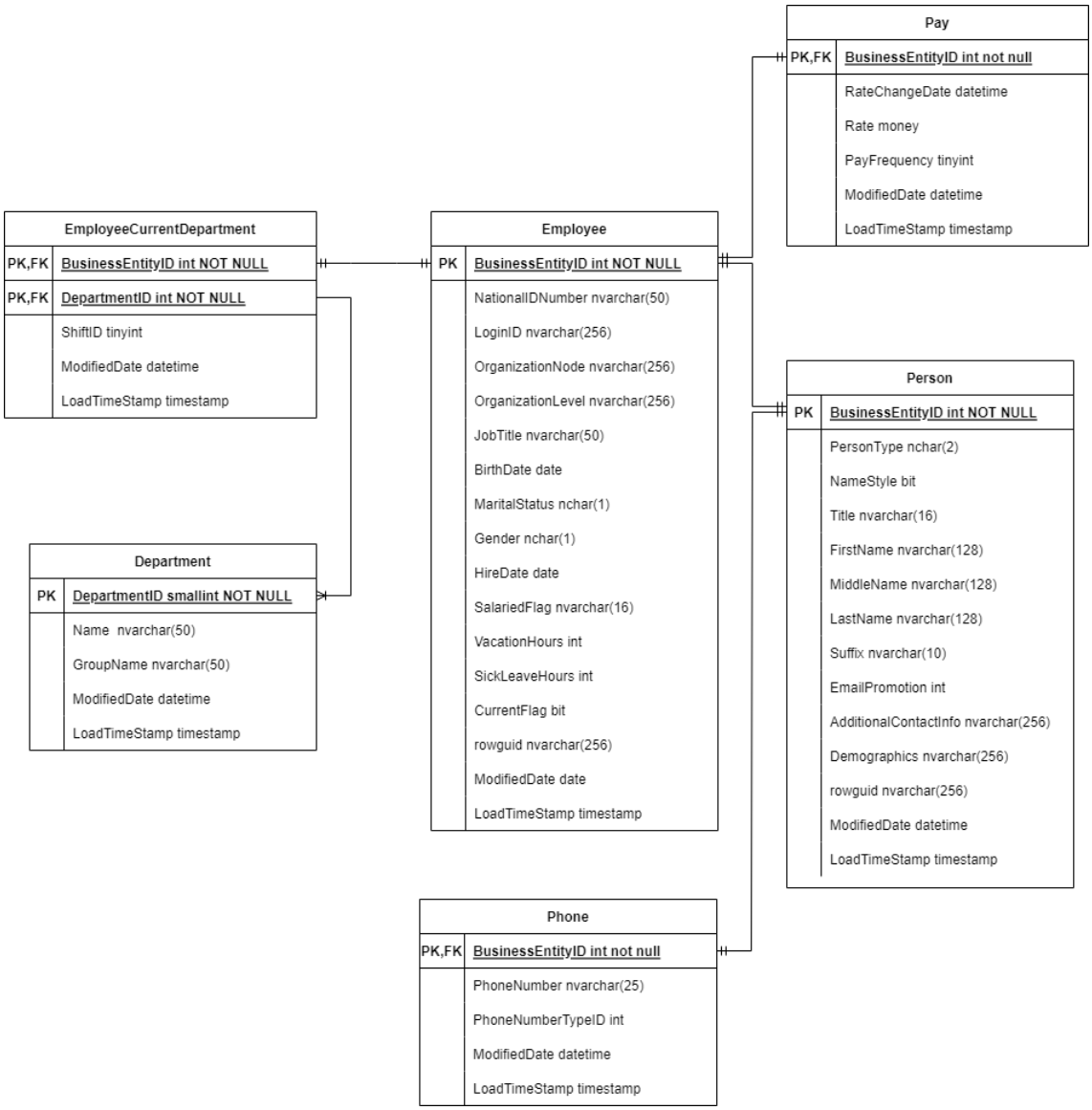
12.1. ábra rendszer terv

Mivel a forrás adatbázis Microsoft SQL Server biztonsági mentéseként áll a rendelkezésemre ezért első lépésként létre kell hoznom egy saját SQL szervert amire a mentést vissza tudom állítani. Az SQL szerveren található adatok grafikus felületén való megtekintésére és SQL parancsok futtatására a Microsoft SQL Management Stúdiót fogom használni, mivel ezt direkt a célra alakította ki a Microsoft.

A forrásadatbázist és az adattárház leképezésére és elérésére az Entity Framework keretrendszert fogom felhasználni. Ez egy modern objektum-adatbázis lekepező modul, ami megkönnyíti a fejlesztést. Az adatbázis modelleket automatikusan létrehozza az adatbázisból kinyert információk alapján. Ezenkívül támogatja a LINQ lekérdezéseket, változás-naplózást, adatbázis-frissítéseket, valamint a séma-migrációkat az alkalmazásunk számára. [15]

12.2. ADATBÁZIS TERVE

A forrás adatbázis részben bemutatott probléma, hogy egy hatalmas forrás adatbázis áll a rendelkezésünkre, viszont az adattárház és a DataMart kialakításához nem lényeges minden tábla. Az első lépés kiválasztani a szükséges táblákat, amelyből a követelményekben leírt riportokat el tudom készíteni (12.2 ábra).



12.2. ábra adatbázis terv

Employee tábla: A munkásokról tárol információkat, erre a táblára mindenképpen szükség van, mivel itt van eltárolva minden információ a vakációkról, betegszabadságokról, házassági állapotról, valamint a nemi hovatartozásról. (290 rekord)

Pay tábla: Erre a táblára is szükség van mivel innen kapjuk meg a fizetési információkat. (290 rekord)

EmployeeCurrentDepartment tábla: Ez egy saját magam által létrehozott tábla az EmloyeeDepartmentHistory-ból generálva, mivel a munkásoknak a részleg információi már alapból historikusan voltak tárolva a forrásadatbázisban, viszont csak az adattárházban kellene így tárolni. Ezért a feladat elkezdése előtt, elkészítettem ezt a kapcsolótáblát, ami csak a jelenlegi részlegüket tárolja. Ez csak egy problémát fog felvetni, hogy beszúráskor és módosításkor ezt a táblát is automatikusan frissíteni kell

majd. Ha kiderül, hogy az adattárház kiépítése után már nem szükséges a kapcsolótábla, akkor egy egyszerű query segítségével az employee táblához hozzáadható egy idegen kulcs a department táblára és ezzel elkerülhető a kapcsolótábla használata. (290 rekord historizálás nélkül, 296 rekord, ha megtartjuk a 6 historikusan tárolt változást.)

Department Tábla: A részlegekről tárol információkat. (16 rekord)

Person Tábla: A dolgozókról, valamint a vásárlókról tárol alapvető információkat, például a nevüket. Ez a tábla opcionális a feladatunk megoldásához, viszont ahhoz, hogy a vezetőség a dolgozókról a nevük alapján is tudjanak információkat megtekinteni, ahhoz szükségünk van a táblára. Másik felhasználás lehet, hogy a suffix tagból lekérjük, hogy kinek van phd-je és ez alapján is tudunk kimutatást készíteni. Valamint lehetőséget nyújt még ahhoz, hogy tesztelni lehessen az adattárház és web applikáció teljesítményét nagyobb adathalmazra is, mert ez a tábla sok rekordot tartalmaz. (19 975 rekord)

Phone Tábla: Telefonszám adatokat tárol a dolgozókról, valamint a vevőkről is. Szintén opcionális a feladatunk megoldásához. Viszont, ha a vezetőség hozzá akar férni a dolgozók telefonszámához, azt csak ezen a táblán keresztül teheti meg. (19 865 rekord)

12.3. ETL FOLYAMAT TERVEZÉSE

A forrás adatbázis felállítása után a következő lépés az ETL folyamat megtervezése. Ez három lépésből fog állni.

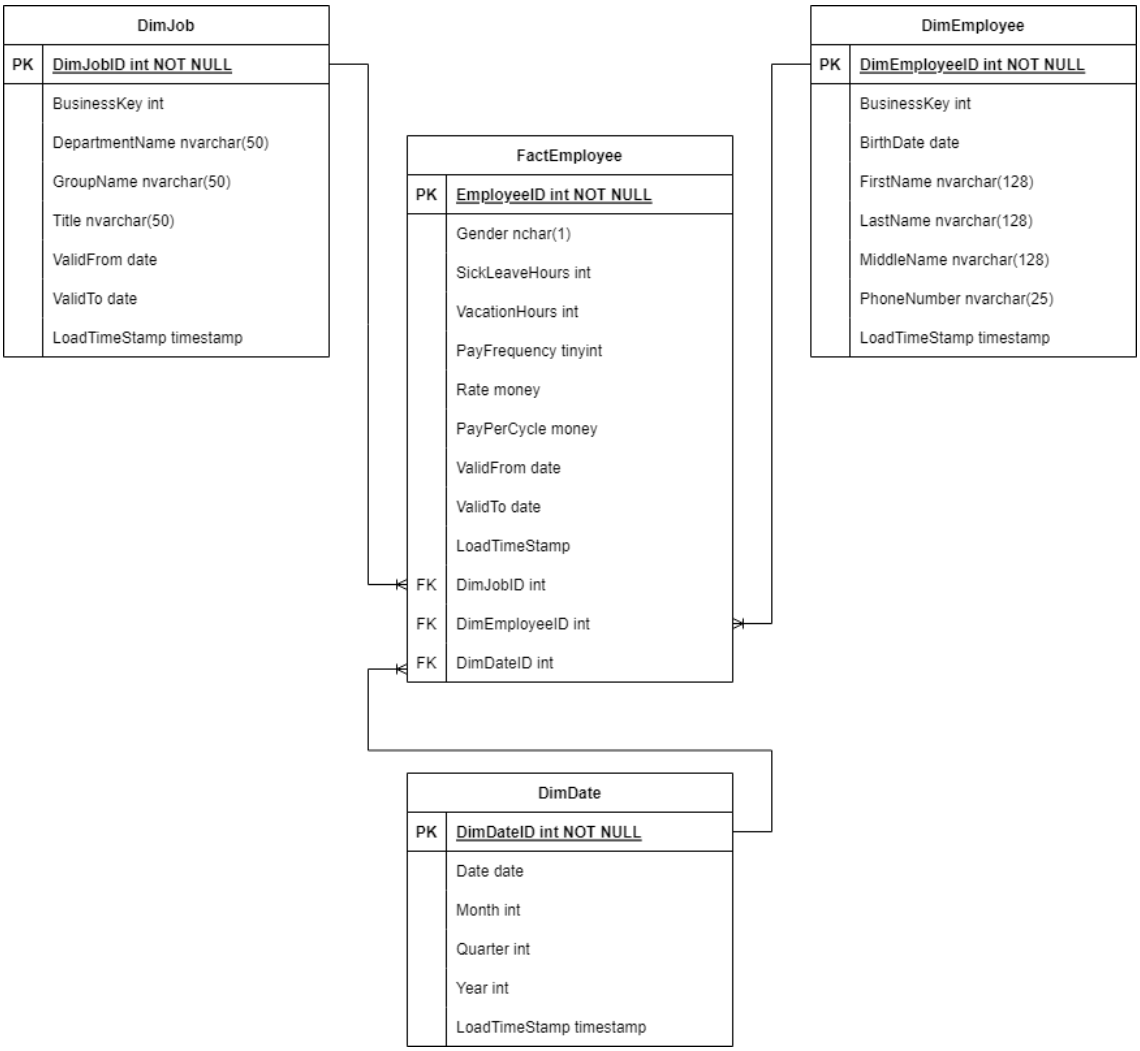
Az első lépés az extract fázis, amelyben szöveges típusra hozom az adatokat (nvarchar), hogy ha a későbbiekben más rendszerekből vagy egyéb forrásból való adatokat is integrálni szeretnénk akkor ebben a részben ezt egyszerűbben megtudjuk tenni mintha típusos lenne. A folyamat eredménye a stageing area-ba fog kerülni, ami egy ideiglenes adatbázis direkt erre a célra.

Következő lépés a transform folyamat kiépítése melyben a stage adatbázisban lévő adatokon végzek el transzformációkat. Ebben a részben fogom elvégezni az adattisztítást, duplikáció kezelést, típus konverziókat, valamint az aggregációkat. A transzformációk eredményeit az adattárház rétegbe fogom betölteni. Az adatokat ebben a rétegben historizálni is kell, hogy az adatokon visszamenőleg is tudjunk elemzéseket végezni.

Az ETL folyamat utolsó lépése a load fázis, melyben az adatokat a kiválasztott adattárház sémába töltöm be (csillagséma). Ez a folyamat tipikusan hozzáírja az adatokat a már meglévőkhöz és csak nagyon ritka, szélsőséges esetekben kell teljes frissítést végezni, azaz a tábla tartalmát törölni és csak az újakat betölteni.

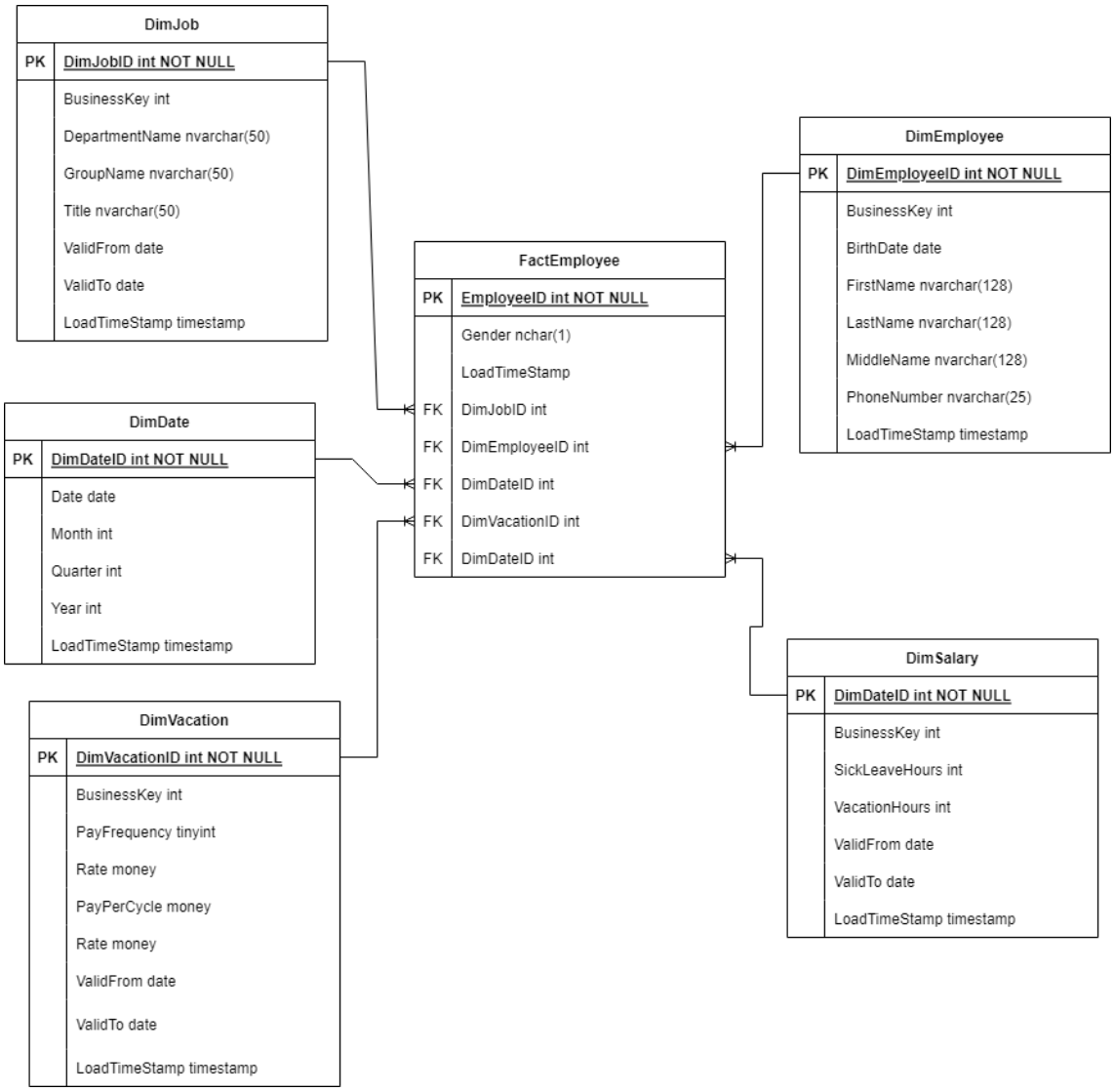
12.4. ADATTÁRHÁZ TERVE

Az első ötletem az volt, hogy a ténytáblában tárolom el a dolgozó nemét, vakációit, valamint a fizetését (12.3.ábra). Viszont ebből az elrendezésből nem derül ki, hogy a historizációnál a ValidFrom és ValidTo oszlopok a fizetésre vagy a vakációkra értendőek-e.



12.3. ábra adattárház első terv

Ezért a későbbiekben áttértem inkább egy másik elrendezésre, ahol a vakációkat és a fizetéseket is kiszerveztem egy külön dimenzióba (12.4. ábra). Ezzel az előző probléma elkerülhető, mivel a dimenzió táblákat külön-külön is historizáltam. Másik megoldás az lett volna, hogy a ténytáblába felveszek kettő ValidTo és ValidFrom oszlopot, amik a különböző attribútumokra vonatkoznak.



12.4. ábra adattárház második terv

Ez az elrendezés teljesítményben lassabb lesz, mint az első megoldás lett volna, viszont könnyebben karbantartható és a későbbiekben a több dimenzió tábla felhasználható egyéb célra létrehozott DataMartokban is.

12.5. ETL FOLYAMAT ÜTEMEZÉSE

Az ETL folyamat elkészítése után, az ETL package-et valamilyen módon napi szinten automatikus módon kell futtatni, lehetőleg munkaórákon kívül, talán este. Erre egy egyszerű módon lesz lehetőségünk SQL Server Managment-en belül lévő SQL Agent-el. Az SQL Server Agent-et direkt erre a célra hozták létre, hogy jobokat (munkákat) tudjunk futtatni időközönként, esemény hatására vagy igény szerint bármikor. [31]

12.6. WEBES APPLIKÁCIÓ TERVEZÉSE

A rendelkezésre álló információk alapján arra a döntésre jutottam, hogy az applikációt C# programozási nyelvben fogom elkészíteni. A fejlesztéshez Visual Studio-t fogok használni, mivel ez már számomra ismert és támogatja az MVC alapokon való fejlesztést .net keretrendszerben.

12.6.1. AZ ASP .NET CORE

Az ASP.NET CORE az újabb verziója az ASP.NET keretrendszernek. Ez egy Microsoft által fejlesztett ingyenes, nyílt forráskádul, cross-platform keretrendszer felhő alapú alkalmazások fejlesztésére. Úgy tervezték, hogy felhőben és on-premise rendszereken is működjön. Ezenkívül lehetővé teszi a modern felhasználói interfész keretrendszerek használatát, mint például a Bootstrap, ezzel is könnyítve a fejlesztést. [20][22]

12.7. FELHASZNÁLÓ ÉS JOGOSULTSÁG KEZELÉS

Az ASP.NET Core sok biztonsági funkciót és könyvtárat kínál a rendelkezésre, ezzel is könnyítve a munkánkat. A jogosultság kezeléshez egy ilyen könyvtárat fogok felhasználni. Név szerint az ASP.NET Core Identity-t. Ez egy API[8], ami támogatja a felhasználói szintű bejelentkezést, valamint intézi a felhasználókat, jelszavakat, szerepköröket és sok más egyebet. Egy egyszerű paranccsal létrehozza a teljes felhasználói adatbázist erre a célra. Beépített metódusokat bocsát a rendelkezésünkre a be és kijelentkezésre, továbbá sok egyéb funkcióhoz. A controllereken belüli metódusokat egy egyszerű attribútummal védetté tudjuk tenni és megadni, hogy milyen szerepkörök érjék el. [16][18]

12.8. WEBES APPLIKÁCIÓ DESIGN

Ez az a pont, ahol a terveim szerint a legtöbb időt fogja igénybe venni, mivel csak kevés előző tapasztalatom van a kinézet tervezésben. A tervezés megkönnyítésére fel fogom használni a Bootstrap keretrendszert. Ez a keretrendszer a legelterjedtebb HTML, CSS és javascript front-end keretrendszer, ami nyílt forráskódú. Ez a keretrendszer tartalmaz előre elkészített formokat, gombokat, táblázatokat és sok más egyebet, amivel kevés módosítással is egy szép letisztult felhasználói felület hozható létre.[22]

12.9. VIZUALIZÁCIÓ KÉSZÍTÉS

A diagrammok elkészítéséhez számos alternatívát megvizsgáltam és végül a ChartJS használata mellett döntöttem, mivel ez egy ingyenes nyílt forráskódú JavaScript könyvtár adatvizualizációkhoz. Támogatja a számomra fontos diagramm típusokat (bar-, line-, pie -chart).[21]

[8] API: Application Programming Interface – Alkalmazásprogramozási felület

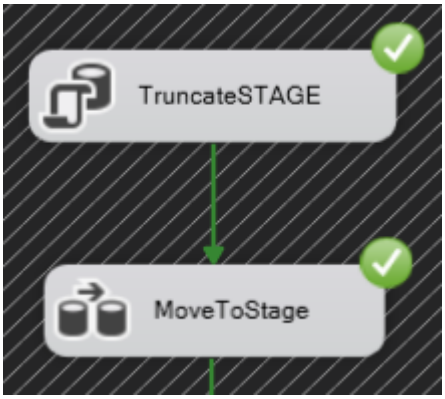
13. MEGVALÓSÍTÁS

13.1. ADATTÁRHÁZ ÉS ETL FOLYAMAT MEGVALÓSÍTÁSA

Első lépésként feltelepitettem az Microsoft SQL Server Express verzióját, valamint a Microsoft SQL Server Management Studiot. Miután ez elkészült, visszaállítottam a biztonsági mentést a .bak állományból. Ez lesz majd a forrás adatbázis a szakdolgozatomhoz. Ezután minimális módosításokat végeztem az adatbázison. A Pay táblába felvettem egy új PayPerCycle computed oszlopot. Erre azért volt szükség mert az adatbázisban egy, kettő vagy három periódusonként kapnak fizetést az alkalmazottak. Viszont így nehezebb egymáshoz viszonyítani a fizetéseket a munkások között. A PayPerCycle oszlopban a fizetést leosztottam a fizetési periódussal, ezzel jobban átlátható, hogy fizetési ciklusonként ki mennyi fizetést kap. Ezenkívül létrehoztam az EmployeeCurrentDepartment táblát az EmployeeDepartmentHistory által, mivel az adattárházban úgyis historikusan lesznek ezek az adatok tárolva, így a forrás adatbázisban nem célszerű. Csak azokat a rekordokat másoltam át, ahol az EndDate null értékkel egyezik meg, azaz csak azokat a rekordokat, amelyek még mindig aktuálisak.

Ezeken kívül létrehoztam az adattárház rétegeiben felhasznált adatbázisokat és táblákat.

13.1.1. ETL EXTRACT MEGVALÓSÍTÁSA



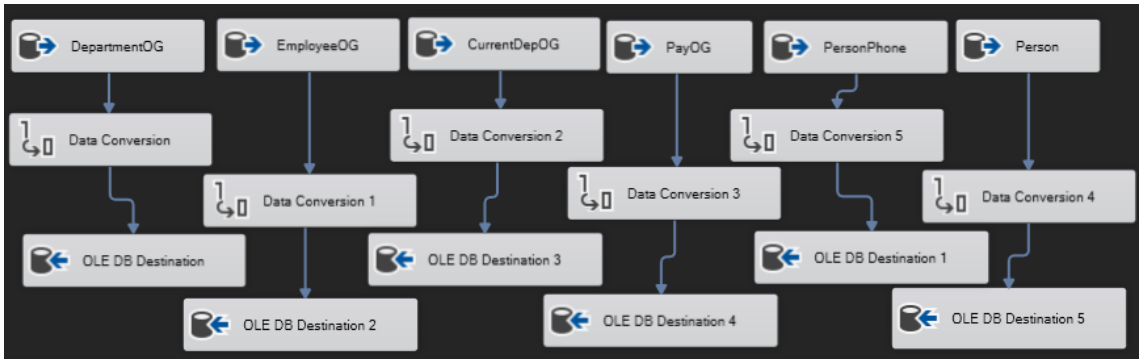
13.1. ábra ETL extract

Az ETL folyamat első lépése az Extract fázis melyben a forrásadatbázisból az adatokat szöveges típusban átmásolom a Stage adatbázisba. Ehhez Visual Stúdió fejlesztői környezetet használtam és azon belül létrehoztam egy új SSIS package-et. Mielőtt kimásoltam az adatokat a forrásadatbázisból, kiürítem a Stage rétegben lévő adatokat, azért, hogy csak az újonnan beérkező adatok legyenek benne. Ezt a képen (13.1. ábra) a TruncateSTAGE SQL parancs (13.2. ábra) lefuttatásával érem el.

ConnectionType	**OLE DB**
Connection	**STAGE**
SQLSourceType	**Direct input**
SQLStatement	**exec sp_MSforeachtable 'TRUNCATE TABLE?'**

13.2. ábra stage réteg truncate

Ha ez sikeresen lefutott akkor a MoveToStage részben elvégzem a típus konverziókat nvarchar típusra.

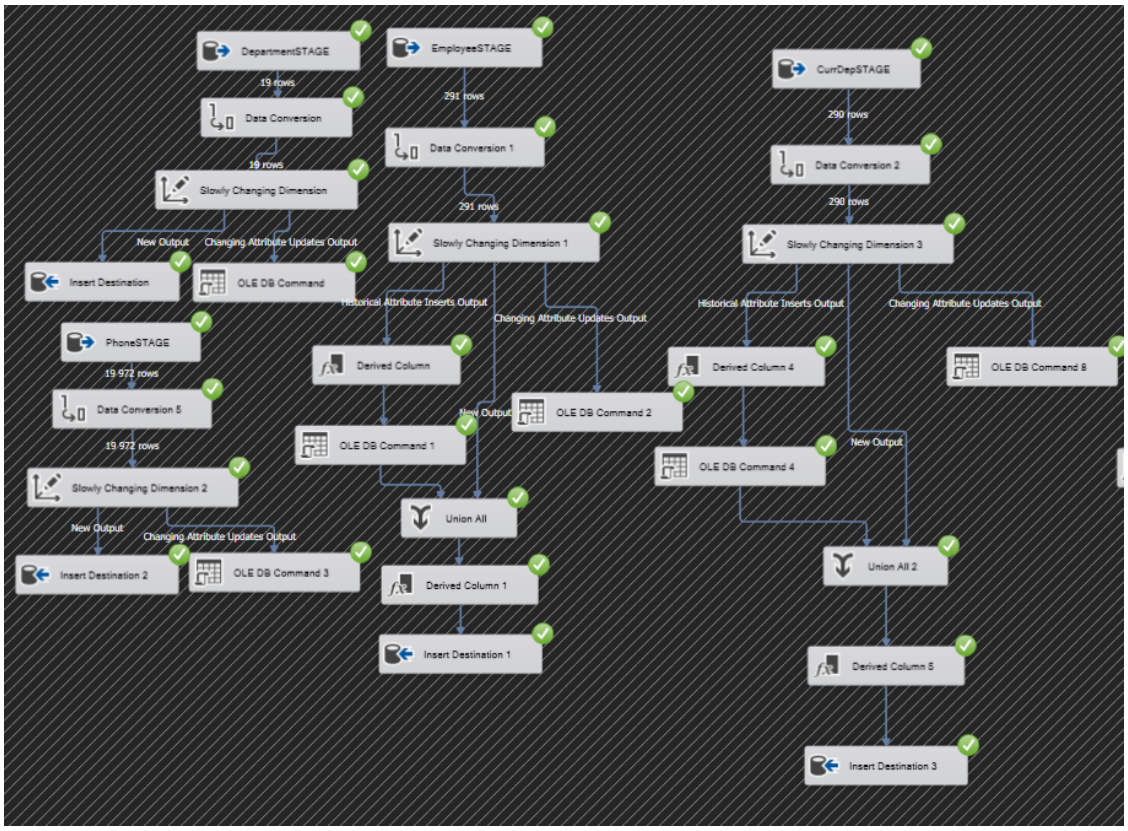


13.3. ábra extract data flow

Itt látható (13.3. ábra), hogy minden tábla egy Data Conversion-re van rákötve. Majd innen kerül betöltésre a Stage rétegbe. Ha a későbbiekben kiderül, hogy a cég más forrásból nem tervez adatokat integrálni az adattárházba, akkor a teljes Stage rész elhanyagolható és a forrásadatbázisból az adatokat már mozgathatjuk is az adattárházba.

13.1.2. ETL TRANSFORM MEGVALÓSÍTÁSA

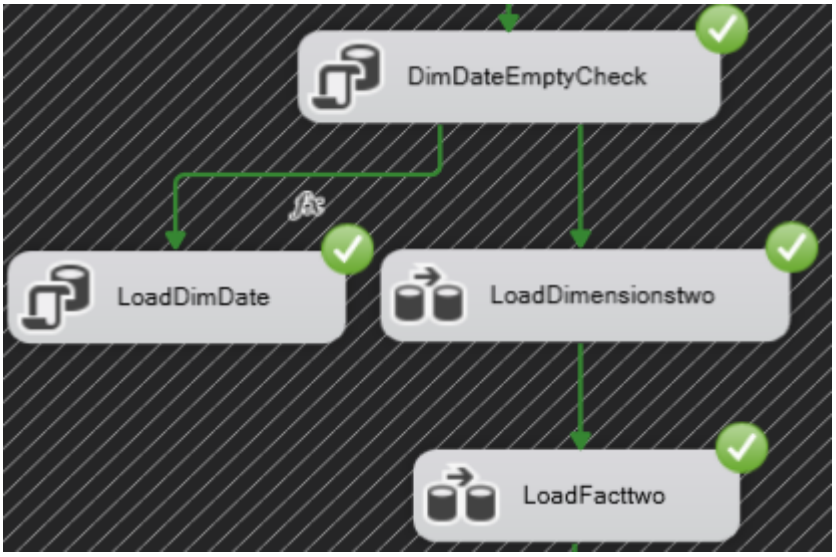
Ha a MoveToStage sikeresen lefutott, akkor a következő lépés a Transform folyamat (13.4. ábra) lefuttatása. Ebben a részben ismételten kell típus konverziókat végezni, mivel a Stage-ben minden adat szöveges típusként szerepel. Ezenkívül még el kell végezni a historizálást a fontosabb attribútumokra.



13.4. ábra transform részlet

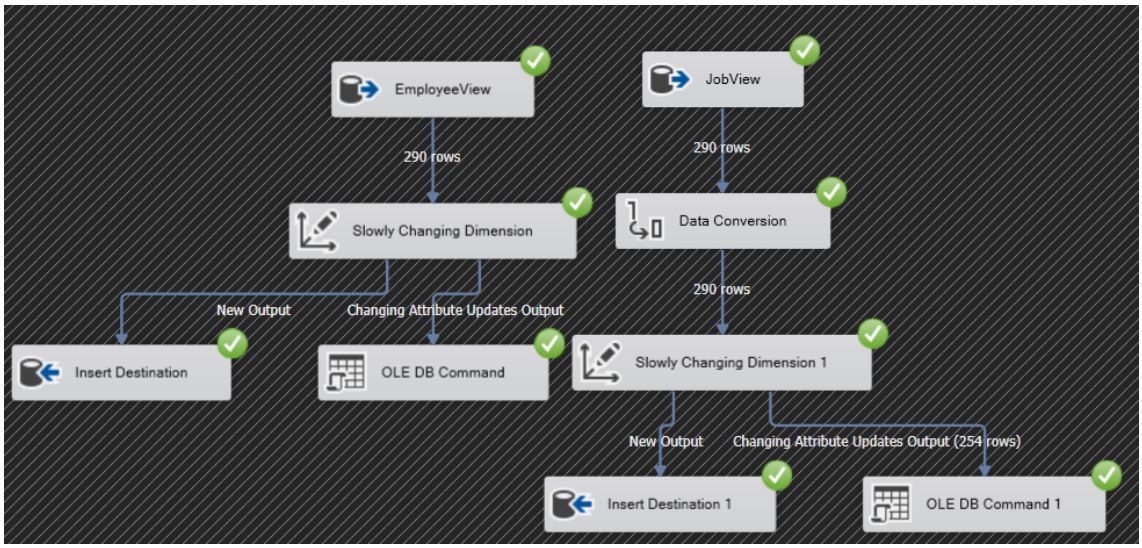
A historizálást Slowly Changing Dimension eszközzel végeztem el, melyben beállítottam, hogy mely attribútumokat milyen módon szeretném historizálni. Például a fizetés oszlopokra (Rate, PayFrequency) type 2 SCD-t alkalmaztam, azaz lesz egy ValidFrom és egy ValidTo oszlop melyben eltároljuk, hogy az adott értékek mettől meddig voltak érvényesek. Ha egy adott érték még jelenleg is érvényes akkor a ValidTo oszlop értéke NULL lesz. Ezenkívül bizonyos attribútumokat beállítottam fixed attribútumra, hogy ha valamilyen módon változás következik be az adott oszlopon, akkor azt hibaként kezelje. Ilyen például a születési dátum. Changing Attribútumot is alkalmaztam az olyan oszlopokra, amelyek változhatnak, viszont nem lényeges, hogy mikor. Ilyen például a részleg neve. A típus konverziók és historizálás után az eredményeket beírjuk az adattárház rétegbe (13.4. ábra).

13.1.3. ETL LOAD MEGVALÓSÍTÁSA



13.5. ábra load controll flow

Ha a transform sikeresen lefutott akkor elkezdhetjük feltölteni a DataMartot. A folyamat (13.5. ábra) a dimenzió táblák feltöltésével kezdődik, azok közül is a dátum dimenzió feltöltésével. A dátum dimenzió a jövőben nem valószínű, hogy változni fog, valamint nem is kell hozzá semmilyen forrás tábla csak létre kell hozni. A load rész azzal kezdődik, hogy megvizsgáljuk, hogy a dátum dimenzió üres-e. Ha üres akkor T-SQL eljárással feltöltjük adatokkal. Ha már van benne adat akkor békén hagyjuk és elkezdjük feltölteni a többi dimenzió táblát.

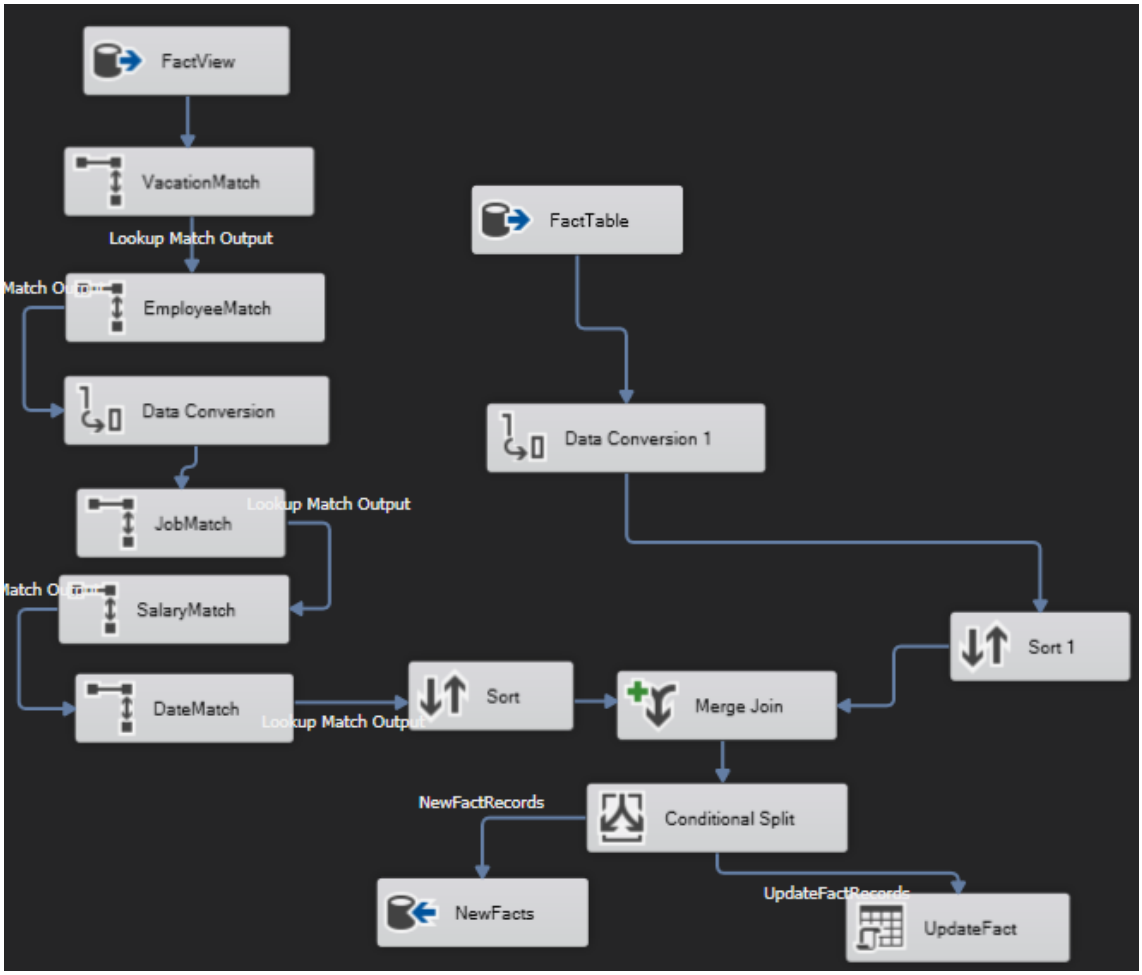


13.6. LoadDimensions data flow részlet

Ehhez először létre kell hozni nézeteket melyben az adatokat tároljuk. A nézeteket az adattárház rétegben hoztam létre, az adattárházban lévő adatok alapján ezért itt már nem

kell újra historizálást végezni. Ezért az összes attribútumot changing attribútumra állítottam, hogy csak felülírjük az adatokat (13.6. ábra).

Ha a dimenzió táblák feltöltésével elkészültünk, akkor a következő lépés a ténytábla (fact) feltöltése adatokkal.



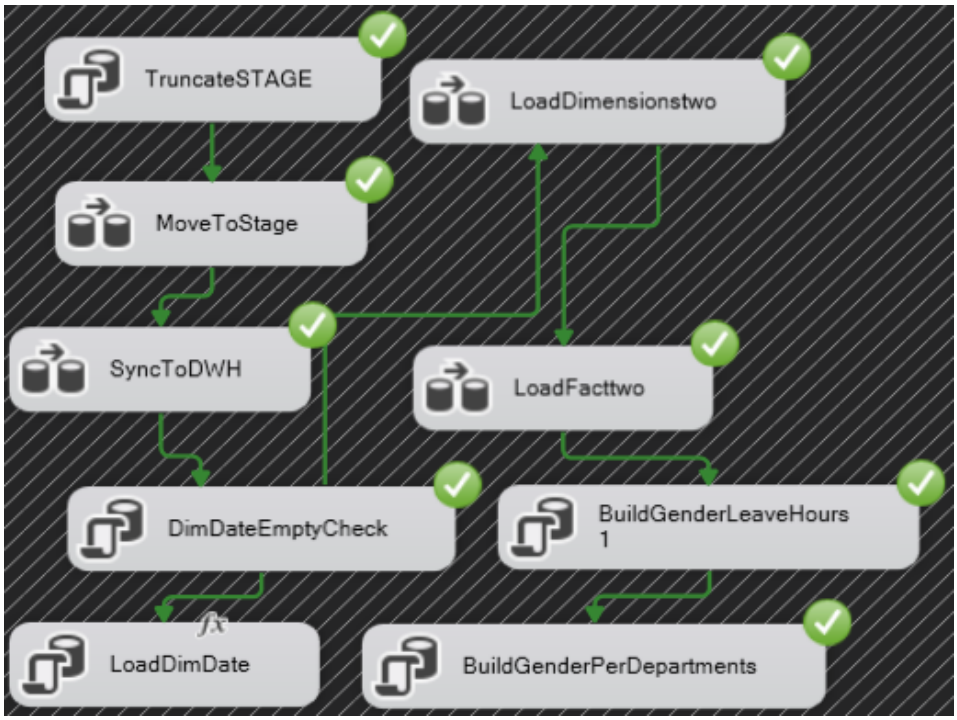
13.7. ábra load fact data flow

Először ehhez is létrehoztam egy nézetet melyben eltárolom a tény táblába érkező rekordokat. Ebben a nézetben csak a dimenzió táblák üzleti kulcsait (BusinessKey) tárolom. Ezért a beszúrás előtt lookup-ok segítségével (13.7. ábra) le kellett kérnem a Surrogate kulcsokat, amelyek a tényleges rekord azonosítók. Ezután egy Merge Join és egy Conditional Split használatával elkülönítettem az újonnan beérkező rekordokat, amelyeket beírtam a ténytáblába, valamint a frissítésre szoruló rekordokat, amiket frissítettem.

13.1.4. RIPORTOKHOZ TARTOZÓ MUTATÓK KISZÁMOLÁSA

Hogy ne C# alapon a webapplikáción keresztül kelljen a mutatószámokat kiszámolni, ezért létrehoztam 2 új táblát, amelyekben a mutatószámokat már előre az ETL folyamat lefutása végén kiszámoltatok.

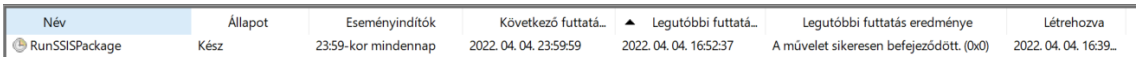
- GenderCountNPayPerDepartments tábla: Tartalma részlegekre osztva a férfi és női dolgozók száma. Valamint az átlag fizetés részlegekre osztva, nemenként.
- GenderLeaveHours tábla: férfi és női vakációk és betegszabadságok számát és átlagát tárolja.



13.8. ábra teljes ETL folyamat controll flow

Az ábrán (13.8. ábra) látható a teljes munkafolyamat. Jól látszik, hogy bizonyos részek csak akkor futnak le, ha az előtte lévő folyamat sikeresen futott le.

13.1.5. ETL FOLYAMAT ÜTEMEZÉSE



Név	Állapot	Eseményindítók	Következő futtatá...	Legutóbbi futtatá...	Legutóbbi futtatás eredménye	Létrehozva
RunSSISPackage	Kész	23:59-kor mindennap	2022. 04. 04. 23:59:59	2022. 04. 04. 16:52:37	A művelet sikeresen befejeződött. (0x0)	2022. 04. 04. 16:39...

13.9. ábra ETL ütemezése

Az ETL folyamat ütemezését terv szerint SQL Agent segítségével végeztem volna el, viszont ez a funkció az SQL Server Express verziójában ez nem érhető el. Ezért egy másik megoldással oldottam meg a problémát. Létrehoztam egy .bat fájlt mely DTExec használatával lefuttatja az SSIS csomagot. Ezután ezt a .bat fájlt a windows feladat ütemezőjével beállítottam napi szintű lefutásra minden nap 23 óra 59 perckor (13.9. ábra).

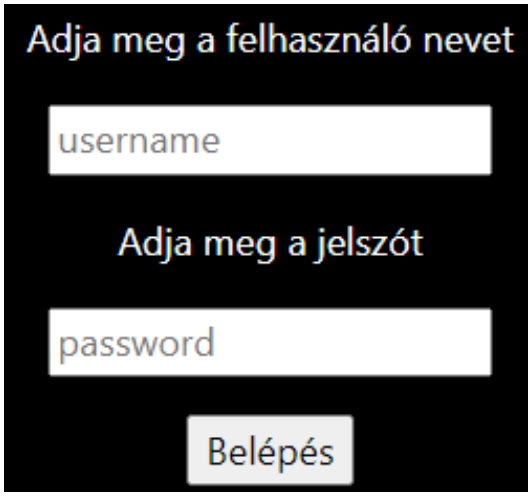
13.2. WEBES APPLIKÁCIÓ MEGVALÓSÍTÁSA

Az eddigiekben kigyűjtött információk alapján az alkalmazást Microsoft Visual Studio 2019 fejlesztői környezetben implementáltam az ASP.NET CORE MVC keretrendszert használva, C# nyelven.

Az alkalmazás Entity Framework Core adatelérési technológia használatával valósítja meg az adatbázissal való kapcsolatot. Ezen keretrendszeren keresztül futtatja a beszúrásokat, lekérdezéseket és az adatok módosítását is. Mivel a forrásadatbázis már a rendelkezésemre áll ezért az adatkezelés folyamatát database first („adatbázist elsőnek”) modell alapján végeztem el. Ez azt jelenti, hogy az adatbázisban szereplő táblák, megszorítások és kapcsolatok már kész vannak és az Entity Framework már ehhez a struktúrához kapcsolódik és ennek megfelelően hozza létre az entitásokat. Ezzel elérhető, hogy ne kelljen minden adatbázis modell-t kézzel megírni, hanem automatikusan típusosan létrehozza.

A kezdeti konfigurációk (Entity Framework modellek és kapcsolat, weboldal elrendezés(„layout”), szükséges csomagok telepítése) elvégzése után létrehoztam a szükséges nézeteket, controllereket és a még hiányzó modelleket. A controllereken belüli metódusokat elláttam a szükséges authorizációs attribútumokkal, hogy csak azok a felhasználók érjék el az adott funkciót, akiknek van is rá jogosultságuk. Valamint beállítottam, ha valaki a böngészőbe beakarná írni az elérési utakat, viszont még nincsenek belépve akkor ne 404-es hibaüzenetet kapjon, hanem irányítsuk vissza a bejelentkező oldalra.

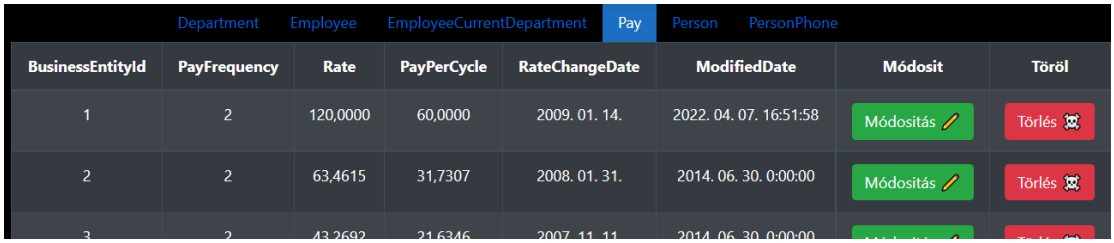
13.2.1. BEJELENTKEZŐ FELÜLET IMPLEMENTÁLÁSA



13.10. ábra bejelentkező felület

A bejelentkező felülethez (13.10. ábra) létrehoztam egy új modellt, amiben eltárolom a felhasználónevet, jelszót, valamint az adott felhasználóhoz tartozó jogosultsági kört. A modellen belül a felhasználónév és jelszó tulajdonságot elláttam egy [Required] attribútummal, hogy üres beviteli mezők esetén ezt a rendszer jelezze a felhasználó felé. Valamint a belépés metódusába le ellenőrzöm, hogy az adott felhasználónév és jelszó páros létezik-e. Ha nem, akkor visszairányítom a bejelentkező oldalra.

13.2.2. ADATBÁZIS MEGTEKINTÉSE, MÓDOSÍTÁSA ÉS TÖRLÉSE



13.11. ábra táblák megtekintése

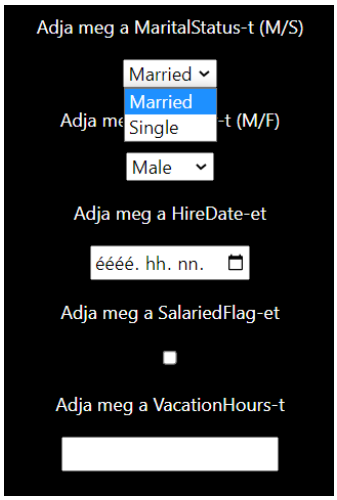
Az adatbázis megtekintésére létrehoztam a nézeten belül egy navigációs sávot, hogy egy nézeten belül több táblának a tartalmát is megtudjuk tekinteni anélkül, hogy a teljes oldalt újra kelljen tölteni. A táblázat kiíratásakor minden sor végére tettem egy módosítás és egy törlés gombot, hogy az adott rekordot tudjuk módosítani. A dizájn bootstrap elemek felhasználásával készült ezért a grafikára nem kellett nagy gondot fordítani (13.11. ábra).

A módosítás gombra kattintva egy új nézetre navigálom a felhasználót melyben az adott rekord minden adata előre kitöltött módon jelenítődik meg. Így elég csak a frissíteni kívánt adatot felülírni. A megfelelő formátumú bevitelre beállítottam, hogy például a dátumok helyére dátum választóból lehessen adatot felvinni, valamint az olyan adatok, amik csak bizonyos típusú adatot tartalmazhatnak, azt legördülő menüből tudja a felhasználó kiválasztani. A hibásan felvitt idegen kulcs hibák, túlcsorduló értékek és egyéb nehezebben lekezelhető hibák esetén egy új oldalra irányítom a felhasználót, ahol egy hibaüzenet formájában kiíratom, hogy a frissítés nem sikerült és a kivétel üzenetét továbbítom a számára.

A törlés gombra kattintva az adatbázisból az adott rekord törlésre kerül.

13.2.3. REKORD BESZÚRÁS

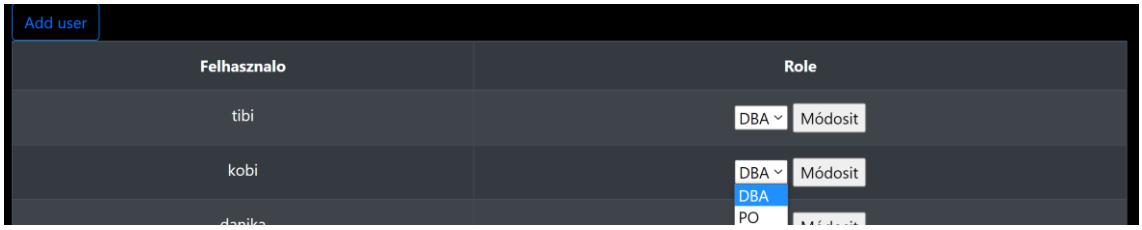
A rekord beszúrás menüpontra kattintva egy új oldalra irányítom a felhasználót, ahol az elérhető táblák listája jelenik meg. A tábla kiválasztása után egy új oldal jelenik meg ahol az adott táblának az attribútumai jelennek meg alatta egy beviteli mezővel.



13.12. ábra beszúrás példa

A hibásan megadott adatok elkerülésére itt is legördülő listákat, dátum választókat, rádió gombokat és a modellen belüli validációs attribútumokat hoztam létre (13.12. ábra). A hibás idegenkulcsok vagy már létező elsődleges kulcs felvitelekor egy új hiba oldalra navigálom a felhasználót, ahol erről értesítem.

13.2.4. FELHASZNÁLÓ MANAGEMENT



13.13. ábra felhasználói management felhasználói felület

A felhasználók menüpontra kattintva jelenítődik meg a felhasználók listája, mellettük, hogy milyen jogosultsági körrel rendelkeznek (13.13. ábra). Ennek a módosítására is van lehetőség legördülő menüből. Az oldalon van még egy gomb, amivel új felhasználót tudunk felvinni a rendszerbe.

A felhasználói adatbázishoz hozzáadtam egy új oszlopot, amibe eltárolom, hogy az adott felhasználó először jelentkezik-e be. Ha igen akkor a bejelentkezést követően egy új oldalra irányítom, ahol meg kell adnia egy új jelszót.

13.2.5. ÜZENETKÜLDÉS

Az applikáción keresztül van lehetőség más felhasználóknak üzenetet küldeniük. Erre minden bejelentkezett felhasználónak van jogosultsága. Erre is létrehoztam egy külön menüpontot, ami egy külön chat oldalra navigálja a felhasználót. Itt látja a feladó, valamint a bejövő üzeneteket a címzett felhasználó nevével. Alatta egy legördülő listából

ki tudja választani, hogy kinek akar üzenetet küldeni mellette pedig egy szövegdobozba tudja beírni az üzentet.

13.2.6. TÖBBNYELVŰSÉG

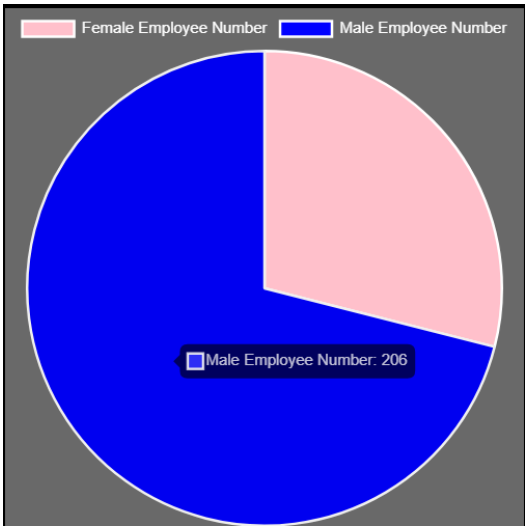
Az angol és magyar nyelv támogatása érdekében a főmenüre felvettem egy „En” és egy „Hu” menüpontot, amire, ha a felhasználó rákattint akkor az meghívja a LanguageController metódusát és beállítja a kért nyelvet. A kívánt nyelvet Cookie-k segítségével tárolom el az alkalmazás számára. Ezenkívül létrehoztam még két resource file-t, amelyekben eltároltam az adott menüpontok és szövegek fordítását angol és magyar nyelven. Ezután az ASP.NET CORE által támogatott localizer segítségével szedtem ki a lefordított szövegeket a megfelelő resource file-ból.

13.2.7. VIZUALIZÁCIÓK

Mint már a tervezés részben is írtam a diagramok elkészítéséhez a ChartsJS JavaScript könyvtárat használtam. Ennek az a hátránya a Power BI és hasonló szolgáltatásokkal szemben, hogy egy-egy vizualizáció elkészítése sokkal hosszadalmasabb folyamat. Valamint minden adatot külön kódon keresztül kell behívni és a felhasználói interakciókat is külön-külön kell lefejleszteni.

Minden diagram mellé megjelenítettem a felhasználó számára az ETL folyamán legenerált táblákat, amelyek tartalmazzák a vizualizációhoz felhasznált adatokat. Ahol az eltérés a férfi és női dolgozók között több mint 15%, ott a kisebb adatot kiemeltem pirossal, hogy a vezetőség lássa, hogy ott érdemes lenne valami változást eszközölni.

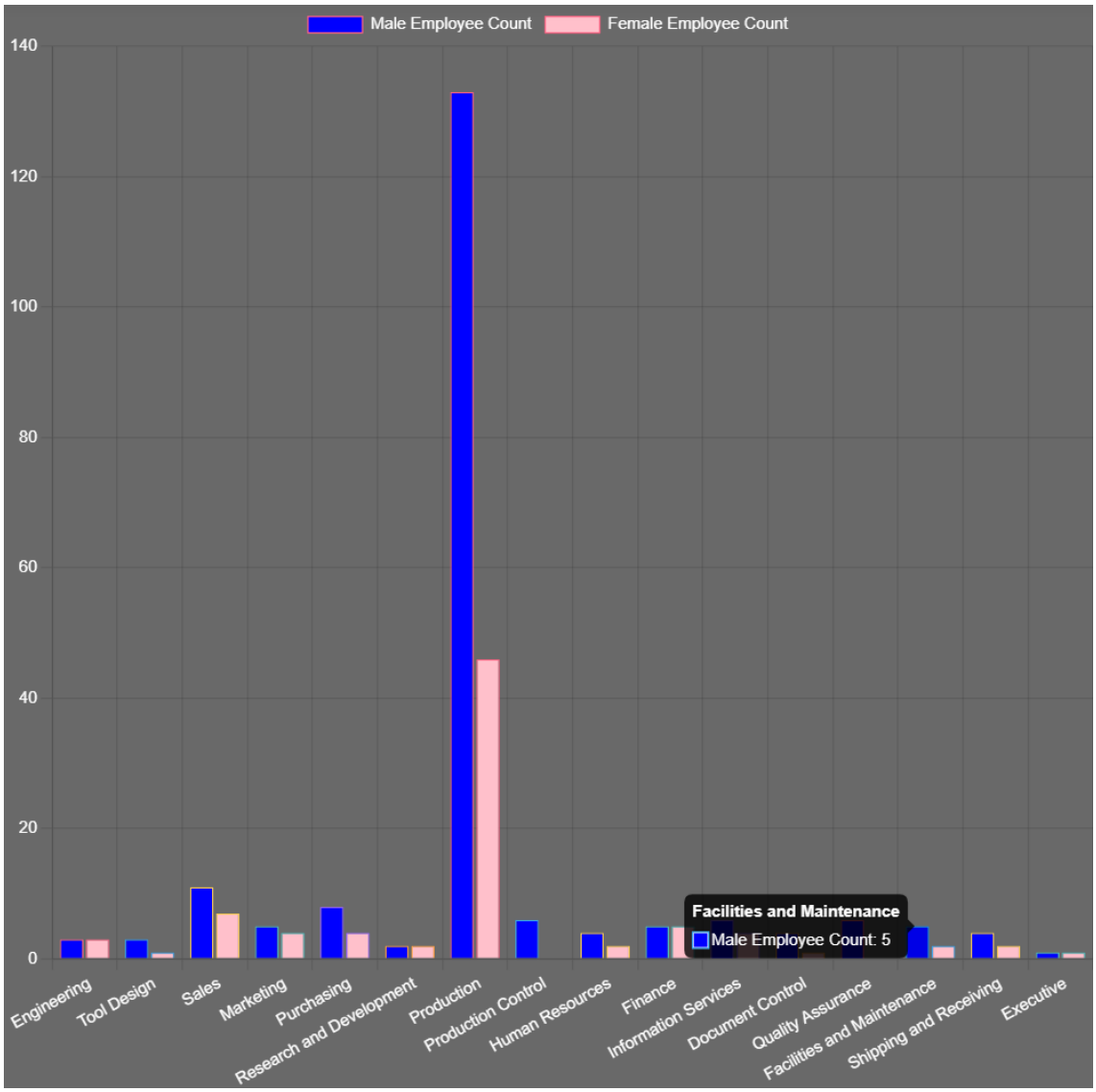
A dizájn tervezésekor próbáltam figyelni arra, hogy a háttér sötétsége miatt az ábrák is jól láthatóak legyenek, viszont az esti használathoz ne legyen túl világos.



13.14. ábra foglalkoztatottak száma nemenként

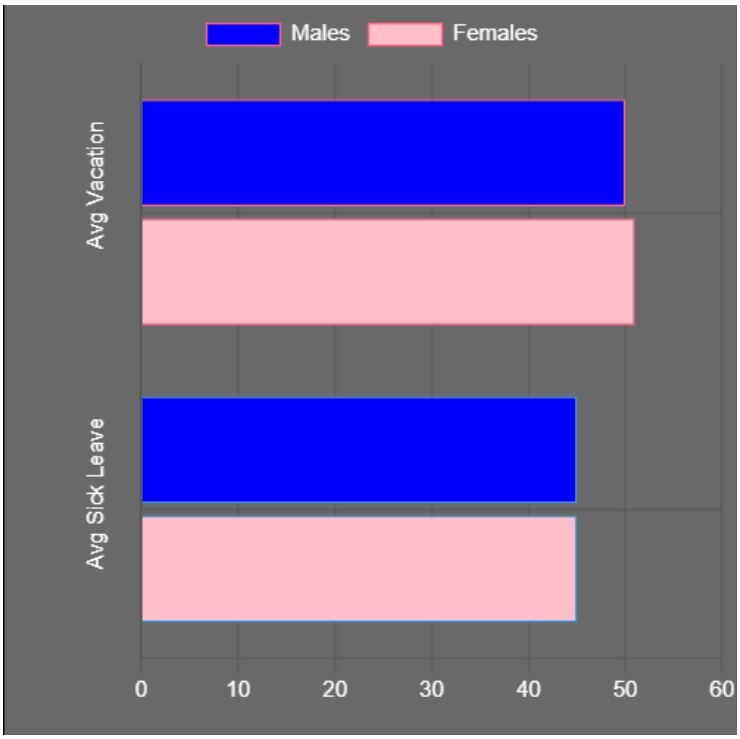
A követelményben nem volt benne, viszont én létrehoztam még egy pie-chart-ot, amin a vezetőség számára vizualizáltam az arányokat, hogy nemenként hány dolgozót

foglalkoztatnak (13.14. ábra). Az adatokat C# kódon belül LINQ lekérdezés segítségével végeztem el.



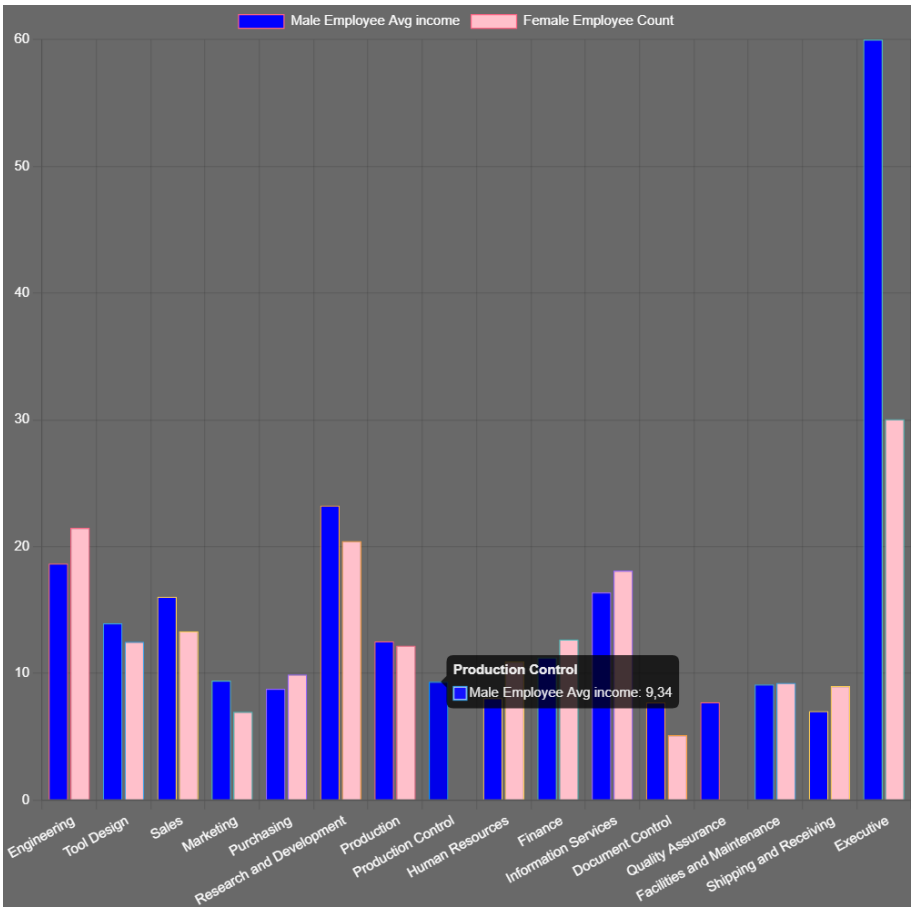
13.15. ábra nemek száma részleg szerint

Ezután, mint az ábrán (13.15. ábra) is látszik ahhoz, hogy a munkások számát jól tudjam viszonyitani egymáshoz egy oszlop diagramot készítettem. Ebből kiderül, hogy mely részlegekben dolgoznak a legtöbben, valamint könnyen észrevehető, hogy a nemek számában hol van jelentős különbség. A férfiakat kék színnel, a nőket rózsaszínnel jelöltem, mivel ezek a nemekhez legkönnyebben kapcsolható színek. A fenti „Male Employee Count” -ra kattintva tudjuk szűrni a diagrammunkat, hogy csak a férfiakat mutassa. Ugyanez érvényes a „Female Employee Count” -ra is.



13.16. ábra átlagos vakáció és betegszabadság

Az első ötletem a férfi és női szabadságok és betegszabadságok vizualizációjára az volt, hogy készítek kettő piechart-ot, viszont később ezt az ötletet elvettettem mivel így a betegszabadságot és a vakáció napokat nehezebb lett volna egymáshoz viszonyitani mint a horizontális oszlopdiagrammon. Ezért inkább a későbbi mellett döntöttem (13.16. ábra). Mint az ábrán is jól látszik a cégnél ezekben a metrikákban nagy különbség nem figyelhető meg, viszont, ha a későbbiekben ez változik akkor ez jól fog látszódni az oszlop diagrammon.



13.17. ábra átlagos kereset nemekre bontva részlegek szerint

A harmadik diagramon (13.17. ábra) vizualizáltam a béreket részlegek szerint csoportosítva azon belül nemekre bontva. Ebből látszik mely részlegekben kell béreket változtatni ahhoz, hogy a cégen belül minden részlegben ugyanakkora átlag jövedelmet kapjanak a dolgozók nemtől függetlenül. Mint az ábrán is látszik például az ügyvezetők (Executive) bérezése arányaiban nagyon eltér. Ebben a beosztásban érdemes lenne a női ügyvezetők bérezését emelni. Az ábrából kiderül még, hogy mely részlegben nem dolgozik valamelyik nemből munkás mivel ilyenkor az adott részleghez nem jelenítődig meg oszlop.

14. TESZTELÉS

A rendszer tesztelését manuálisan végeztem. Elsőként az adatbázis és az ETL folyamatot teszteltem. Az adatbázishoz manuálisan hozzáadtam új rekordokat, valamint már meglévőket frissítettem. Minden historizált attribútumból legalább öt rekordon belül átírtam az értékeit, aztán manuálisan lefuttattam az ETL folyamatot, hogy megbizonyosodjak arról, hogy jól működik-e a historizáció. Az ETL folyamat lefutása után láttam, hogy a folyamat az elvárt eredményre jutott. Amelyik attribútumra type 2-es SCD-t tettem azokat jól historizálta és a ValidFrom és ValidTo oszlopokat jó adattal töltötte fel. Megnéztem, hogy az adatkonverziók is jól lefutottak-e. Nem keletkezett hibás adat se az adattárházban se a DataMart-ban.

Az ETL folyamat tesztelése után a webapplikációt is manuálisan teszteltem. Modulonként egyesével végig mentem minden funkción és megnéztem, hogy hibás adatokat, nemlétező idegen kulcsokat, túlcsorduló értékeket vagy null értéket adok be a rendszernek akkor képes-e ezt lekezelni és a hibákat a felhasználó számára továbbitani.

A bejelentkező felületen minden rendben volt. Hibás vagy nemlétező felhasználó/jelszó páros esetén az adott oldalon megjelent a hibaüzenet. Üresen hagyott, de kötelezően megadandó mező esetén is még az adott oldalon kijelezte a felhasználó számára a hibát.

A felhasználó kezelés tesztelésére létrehoztam minden jogosultsági körből egy felhasználót és egyesével beléptem és megnéztem, hogy csak azokhoz a funkciókhoz férek-e hozzá, amelyekhez az adott felhasználóval jogosultságom van.

Adat módosításakor szintén beadtam hibás adatokat, már létező elsődleges kulcsokat, valamint nem létező idegen kulcsokat próbáltam felvinni a rendszerbe. Ennek a kezelésére try-catch kivételkezelést alkalmaztam. Ha a rendszer valamilyen kivételre jutna akkor a kivétel üzenetét továbbította a felhasználó számára egy új oldalon keresztűl. A probléma, hogy az adatbázis oldali hibák nehezen értelmezhetőek egy nem hozzáértő felhasználó számára. Ezt a jövőben jó lenne lekezelni és hibaüzeneteket átfogalmazni felhasználó barátabb szövegezésűre.

Ezután a vizualizációk tesztelése maradt. A vizualizációk megtekintésekor a rendszer lekéri az adatokat az adatbázisból és azokat az értékeket jeleníti meg amik éppen akkor a táblában találhatóak. Ezért elsőnek megnéztem, hogy a vizualizációk az adatoknak megfelelően jelennek meg. Amint megbizonyosodtam, hogy jól működik utána az adatbázisban az ETL folyamat tesztelésekor változtatott adatokat visszavontam és megnéztem, hogy ennek hatására a vizualizációban is megfigyelhető ez a változás.

15. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

A cég számára több olyan funkciót lehetne implementálni, ami hasznos lenne a vezetőség számára. Ilyen például a riportok kinyomtatásának a lehetősége, esetleg PDF-be való exportálhatósága. Ezenkívül mint a tesztelés részben is írtam, a hibakezelés esetén az adatbázis oldali kivételek hibaüzenetét egy az egyben átadom a felhasználó számára, viszont ennek a szövegezése nem a legkönnyebben értelmezhető egy nem hozzáértő számára. Ezért érdemes lenne a gyakran előforduló hibaüzenetek szövegezését átírni, hogy könnyebben értelmező legyen a felhasználó számára. Esetleg érdemes lenne egy menüpontot is létrehozni az alkalmazáson belül, ami a hibakódok megoldására adna útmutatást. Az idegenkulcs és elsődleges kulcs hibakezelését is meg lehetne valósitani az alkalmazáson belül, viszont ehhez minden módosításkor le kellene kérni a már létező kulcsok listáját, amit, ha sok felhasználó egyszerre akarna megtenni az leterhelné a szervert.

A szakdolgozatom során csak az adattárház kiépítése során felhasznált táblákat vittem át a webalkalmazásban. A későbbiekben érdemes lenne a cég számára az összes többi táblát is felvinni a rendszerbe.

Ha a jövőben a cég nőne, több alkalmazottat foglalkoztatna vagy esetleg több terméket forgalmazna, akkor érdemes lenne átállni on-premise rendszerről cloud based rendszerre. Ennek előnye, hogy az ára nagy adatmennyiségek esetén olcsóbb lenne, mint a földi rendszernek. Nem kellene a szervergépet karban tartani, tárhely dinamikusan bővíthető lenne, valamint az erőforrások könnyen skálázhatóak több számítási egység igénybevételével. Ezenkívül a cloud szolgáltatók általában adnak lehetőséget biztonsági mentés készítésére egy teljesen más területen lévő szerverre. Ezért, ha bekövetkezne bármilyen katasztrófa egy adott szerverparkban, akkor nem veszne el az adatunk, hanem egyszerűen visszaállítható lenne a biztonsági mentésből.

A vizualizációkat saját magam készítettem viszont ezt a jövőben érdemes lenne egy direkt erre a célra fejlesztett rendszeren keresztül végezni (Például: Power BI, Tableau).

16. ÖSSZEFOGLALÁS

A dolgozatomban bemutattam egy vállalati környezetben, hogyan lehet adattárházat kiépíteni ETL folyamat segítségével. Továbbá elkészítettem egy interneten keresztül elérhető online felületet, amely a követelményrendszerben kiírt funkciókat valósítja meg.

Első lépésként utána jártam a nemek közötti juttatások különbségeinek. Megismerkedtem az alapvető problémával, hogy a nők valamilyen módon hátrányban vannak a férfiakkal szemben. A problémák feltárására összeállítottam egy követelményrendszert, ami egy lehetséges megoldását kínál adattárház és vizualizációk használatával.

Ezt követően megvizsgáltam a követelményrendszerben megfogalmazott feladatok megoldására alkalmas technológiákat, szoftvereket, adattárház sémákat és az online felület kialakítására alkalmas modelleket. Ezután kiválasztottam azokat az elemeket amelyikről bizonyosságot nyertem, hogy tudnak együtt működni, valamint teljesítményben is elől járnak. Az adatbázis kezelő rendszernek a Microsoft SQL Szerver-t, ETL folyamat elkészítésére az SQL Server Integration Services-t választottam. Az online felület elkészítéséhez C# programozási nyelvet és MVC model-t használtam. Ezenkívül a felület elkészítéséhez felhasználtam több a szakdolgozatomban leírt könyvtárat.

Ezután összeállítottam a rendszertervet, elkészítettem az adattárház számára a már saját adathalmazra alkalmazható csillagséma modelljét, valamint meghatároztam az online felület számára a jogosultsági köröket.

A megvalósítás fejezetben bemutattam az implementáció menetét. A fejlesztés során nem merült fel olyan probléma, amit valamilyen módon ne tudtam volna kijavítani.

A megvalósítás után manuálisan leteszteltem a rendszert, hogy minden úgy működik-e, ahogy arra terveztem. Ezt részletesen kifejtettem a tesztelés fejezetben.

17. SUMMARY

In my thesis, I demonstrated how to build a data warehouse in an enterprise environment using the ETL process. Furthermore, I have created an online interface accessible via the Internet, which implements the functions described in the requirements specification.

As a first step, I investigated the differences in benefits between genders. I looked at the basic problem that women are in some way disadvantaged compared to men. To explore these problems, I put together a set of requirements, offering a possible solution using a data warehouse and visualisations.

I then looked at the technologies, software, data warehouse schemas and models for designing the online interface to solve the problems outlined in the requirements framework. I then selected those elements that I was confident could work together and were ahead in performance. I chose Microsoft SQL Server as the database management system and SQL Server Integration Services to build the ETL process. To build the online interface I used C# programming language and MVC model. In addition, I used several libraries described in my thesis to build the interface.

I then created the system design, created a star schema model for the data warehouse that could be applied to the existing dataset, and defined the scope of permissions for the online interface.

In the implementation chapter I described the implementation process. During the development, no problems were encountered that I could not fix in some way.

After implementation, I tested the system manually to make sure that everything worked as I had intended. This is explained in detail in the testing section.

Irodalomjegyzék

[1] Zoiner Tejada: Kinyerés, átalakítás és betöltés (ETL), (https://docs.microsoft.com/hu-hu/azure/architecture/data-guide/relational-data/etl) Utoljára megtekintve: 2021.11.09.

[2] Rakesh Parida: SSIS and Data Sources, (https://social.technet.microsoft.com/wiki/contents/articles/1947.ssis-and-data-sources.aspx) Utoljára megtekintve: 2021.11.09.

[3] Koen Verbeeck: SSIS and Pentaho, (https://sqlkover.com/ssis-and-pentaho-a-quick-comparison/) Utoljára megtekintve: 2021.11.09.

[4] Jake Stein SQL Server Integration Services (SSIS) vs. Pentaho Data Integration (Kettle), (https://www.stitchdata.com/vs/ssis/pentaho/) Utoljára megtekintve: 2021.11.09.

[5] Tamara Scott: Power BI vs Tableau: A Data Analytics Duel, (https://technologyadvice.com/blog/information-technology/power-bi-vs-tableau/) Utoljára megtekintve: 2021.11.09

[6] Tamara Scott: POWER BI VS TABLEAU, (https://technologyadvice.com/blog/information-technology/power-bi-vs-tableau/) Utoljára megtekintve: 2021.11.09.

[7] Hilary M. Lips: The Gender Pay Gap: Concrete Indicator of Women's Progress Toward Equality, (https://spssi.onlinelibrary.wiley.com/doi/abs/10.1111/j.1530-2415.2003.00016.x) Utoljára megtekintve: 2021.11.09.

[8] Md Kamaruzzaman: Top databases in 2021, (https://towardsdatascience.com/top-10-databases-to-use-in-2021-d7e6a85402ba) Utoljára megtekintve: 2021.11.09.

[9] solid IT gmbh: MsSql MySql és Oracle összehasonlítása, (https://db-engines.com/en/system/Microsoft+SQL+Server%3BMySQL%3BOracle) Utoljára megtekintve: 2021.11.09.

[10] Vijay Shinde: Adatbázis szerverek összehasonlítása, (https://www.softwaretestinghelp.com/sql-vs-mysql-vs-sql-server/) Utoljára megtekintve: 2021.11.09.

[11] Vijay Shinde: MySql alapok, (https://www.softwaretestinghelp.com/what-is-mysql/) Utoljára megtekintve: 2021.11.09.

[12] Martin Fowler: The Presentation Model Design Pattern, (martinfowler.com) Utoljára megtekintve: 2021.11.09

[13] Szabó Zsolt: Software technology and GUI design, (https://users.nik.uni-obuda.hu/prog4/) Utoljára megtekintve: 2021.11.09.

[14] Stackoverflow fórum az AdventureWorks adatbázissal kapcsolatban, (https://stackoverflow.com/questions/13220573/adventure-works-explanation) Utoljára megtekintve: 2021.05.08

[15] Entity Framework dokumentáció, (https://docs.microsoft.com/en-us/ef/) Utoljára megtekintve: 2021.05.08

[16] Identity könyvtár dokumentációja, (https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity-configuration?view=aspnetcore-6.0) Utoljára megtekintve: 2022.02.03

[17] Identity könyvtár dokumentációja, (https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-6.0&tabs=visual-studio) Utoljára megtekintve: 2022.02.03

[18] Authorization könyvtár dokumentációja, (https://docs.microsoft.com/en-us/aspnet/core/security/authorization/simple?view=aspnetcore-3.1) Utoljára megtekintve: 2022.02.03

[19] Authorization könyvtár dokumentációja, (https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-3.1&tabs=visual-studio) Utoljára megtekintve: 2022.02.03

[20] ASP.NET CORE dokumentációja, (https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0) Utoljára megtekintve: 2021.05.02

[21] Chart.js dokumentációja, (https://www.chartjs.org/) Utoljára megtekintve: 2022.02.08

[22] Bootstrap dokumentációja, (https://getbootstrap.com/) Utoljára megtekintve: 2022.02.08

[23] Globalization and the Gender Wage Gap, (https://academic.oup.com/wber/article-abstract/23/1/141/1709207?login=false) Utoljára megtekintve: 2022.02.08

[24] The gender wage gap and its institutional context, (https://journals.sagepub.com/doi/abs/10.1177/0950017012460322) Utoljára megtekintve: 2022.02.08

[25] Csillagséma dokumentációja, (https://docs.microsoft.com/hu-hu/power-bi/guidance/star-schema) Utoljára megtekintve: 2022.02.08

[26] hópehely és galaxis séma összehasonlítása, (https://hevodata.com/learn/star-and-snowflake-schema-analysis/) Utoljára megtekintve: 2022.02.08

[27] Postgre SQL (https://www.postgresql.org/) Utoljára megtekintve: 2022.02.08

[28] Oracle (https://www.oracle.com/database/) Utoljára megtekintve: 2022.02.08

[29] MS SQL server (https://www.microsoft.com/en-us/sql-server/) Utoljára megtekintve: 2022.02.09

[30] SSIS dokumentáció (https://docs.microsoft.com/en-us/sql/integration-services/sql-server-integration-services?view=sql-server-ver15) Utoljára megtekintve: 2022.02.09

[31] SQL Agent dokumentáció (https://docs.microsoft.com/en-us/sql/ssms/agent/sql-server-agent?view=sql-server-ver15) Utoljára megtekintve: 2022.02.09

Ábrajegyzék