ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR

SZAKDOLGOZAT

OE-NIK	Hallgató neve:	**Kosztolányi Attila**
2022	Hallgató törzskönyvi száma:	**T/005694/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai Rendszerek Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Kosztolányi Attila**
Törzskönyvi száma: T/005694/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

Hívásadatok feldolgozása és webes felületen való vizualizálása
Call detail processing and visualizing in a web application

Intézményi konzulens: Dr. Nagy Enikő
Külső konzulens:

Beadási határidő: 2022. május 15.

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

A feladat

Egy telefonközpont üzemeltetéssel foglalkozó cég nyers adatainak feldolgozása jelenleg problémát jelent. Ennek megoldására egy felhasználóbarát, könnyen kezelhető és átlátható felületet kell kialakítani.

A hallgató feladata a projekt kapcsán meghatározott adatforrásokból és informatikai rendszerekből az adatok átvitelének megtervezése és megvalósítása ETL réteg segítségével adattárházba. Továbbá egy webalkalmazás tervezése és elkészítése, amely az adattárház segítségével alkalmas az adatok szűrésére és keresésére, riportolására, statisztikák vizualizálására a felhasználói igényeknek megfelelően, valamint a kimutatások felhasználásával képes optimalizálási lehetőségeket adni.

A dolgozatnak tartalmaznia kell:

- a feladat pontos ismertetését,
- rendelkezésre álló adatok felépítésének ismertetését,
- hasonló, már létező rendszerek bemutatását,
- a feladat megoldásához alkalmazható technológiák tömör bemutatását,
- a választott megoldások indoklását,
- a tervezés és megvalósítás részletes leírását,
- az elért eredményeket,
- a továbbfejlesztési lehetőségeket.

P.H.

ÓBUDAI EGYETEM Neumann János Informatikai Kar

.....
Dr. Fleiner Rita
intézetigazgató

A szakdolgozat elévülésének határideje: **2024. május 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
9. sz. melléklet

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Neptun kód: Tagozat:

Kosztolányi Attila. [redacted] Nappali

Telefon: Levelezési cím (pl: lakcím):

[redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

Hívásadatok feldolgozása és webes felületen való vizualizálása

Szakdolgozat / Diplomamunka[2] címe angolul:

Call detail processing and visualizing in a web application

Intézményi konzulens: Külső konzulens:

Dr. Nagy Enikő .....

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2021.09.18.	Témaválasztás egyeztetése	
2.	2021.10.12.	Szakdolgozat felépítése	
3.	2021.11.19.	Irodalomkutatás	
4.	2021.12.08.	Követelmények megbeszélése	

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, 20 21.12.08

.....
intézményi konzulens

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

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
9. sz. melléklet

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Kosztolányi Attila

Neptun kód: [redacted]

Tagozat: Nappali

Telefon: [redacted]

Levelezési cím (pl: lakcím): [redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

Hívásadatok feldolgozása és webes felületen való vizualizálása

Szakdolgozat / Diplomamunka[2] címe angolul:

Call detail processing and visualizing in a web application

Intézményi konzulens: Dr. Nagy Enikő

Külső konzulens: .....

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2022.02.25.	Adatbázis felépítése	
2.	2022.03.18.	Rendszerterv átnézése	
3.	2022.04.15.	Megvalósítás és eredmények	
4.	2022.05.06.	A 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, 20 22. 05. 09.

.....
intézményi konzulens

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

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
7. sz. melléklet

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. 09.

.....
hallgató aláírása

TARTALOMJEGYZÉK

ÁBRA- ÉS TÁBLÁZATJEGYZÉK

1. BEVEZETÉS

A telekommunikáció, vagy másnéven távközlés már az ókor óta velünk van valamilyen formában, ám mai értelmezésében mégis csak a 19. században tudott ugrásszerű fejlődésnek indulni, majd elterjedni. Az első elektromos jelátviteli technológia feltalálását Samuel Morse nevéhez kötjük, aki 1837-ben elsőként valósított meg távolra történő üzenetküldést. A következő nagy lépcsőfok a telefon feltalálása volt 1876-ban, amit Alexander Graham Bell érdemeként könyvelnek el. Mára már életünk szerves részét képezik a különböző telekommunikációs eszközök, én mégis egy kevésbé ismert, speciális szegmensével fogok foglalkozni ebben a dolgozatban, ami nem más, mint a telefonos alközpontok.

Szakdolgozatom témája egy hívásadatokat feldolgozó rendszer létrehozása, amely a nyers log fájlok rendezése és tisztítása után képes a rendelkezésre álló adatok alapján jelentések készítésére és infó grafikák megjelenítésére egy webes felület segítségével. Az elkészült rendszer egy telefonos alközpontok üzemeltetésével foglalkozó cég, az MD Vonal Kft. hatékony működését szolgálja majd, amelynél jelenleg én is dolgozom.

Ezzel a dolgozattal és a kitűzött feladat teljesítésével azt szeretném elérni, hogy a jelenleg hasonló célra használt eszköz helyettesítésére egy olyan alternatívát tudjak kínálni, amely informatívabb és a felhasználó szemszögéből könnyebben kezelhető, mint a jelenlegi, ezáltal alkalmas legyen egyszerű statisztikai mérésekre. További célom az adatok megjelenítése mellett az is, hogy felhasználó által paraméterezhető szűréseket tudjon a rendszer elvégezni, és ezek eredményét képes legyen a megfelelő formában visszaadni. Az így kapott kimutatások kifejezetten hasznosak lehetnek esetleges hibakereséseknél, esetleg olyan helyzetekben amikor egy konkrétabb dologra vagyunk kíváncsiak, például egy adott hívásra, egy bizonyos telefonszámra vagy egy pontosan meghatározott időintervallum történéseire. A program feladata ezenkívül, hogy a létrejött statisztikák alapján, amolyan döntéstámogató rendszerként elősegítse az esetleges optimalizálási folyamatokat egy adott cégen belül. A későbbiekben bemutatásra kerül természetesen a jelenleg használatban lévő rendszer, annak előnyeivel és hátrányaival együtt, valamint a dolgozat végén ismertetem a potenciális továbbfejlesztési lehetőségeket.

2. A KÖRNYEZET BEMUTATÁSA

2.1. MD Vonal Kft.

Az MD Vonal Távközléstechnikai Szolgáltató korlátolt felelősségű társaság 2003-ban alakult meg, mint vállalkozás, de már 1993 óta foglalkoznak Ericsson típusú alközpontok forgalmazásával, telepítésével, teljeskörű üzemeltetésével és javításával, illetve az ezekhez tartozó kiegészítők, mint például a tarifázó egység vagy gyenge áramú hálózatok karbantartásával. Természetesen ide sorolandó a végfelhasználóknál lévő telefonkészülékek felügyelete és probléma esetén a javítása is. Az Ericssonon belül érdemes két fő típust megemlíteni. Az egyik a BusinessPhone, ami kisebb irodai környezetekhez alkalmas és maximum 300 mellékig bővíthető. A másik típus az MD110-es alközpont, ami egy professzionális modell és ennek megfelelően az üzleti kommunikáció széles tartományában nyújt megoldást kis és középvállalatoknak vagy akár az országos szintű hálózatoknak is. A cégnek számos kisebb-nagyobb ügyfele van országszerte melyek saját alközpontjai folyamatosan jegyzik a hívás rekordokat. Ezek bizonyos időközönként feldolgozásra majd elemzésre kerülnek. Ahhoz, hogy sok egymástól független és olykor több száz kilométerre lévő központnak tudják biztosítani a magas rendelkezésreállást, elengedhetetlen feltétel a távoli hozzáférés és adatgyűjtés lehetősége. Ily módon a gyors és precíz beavatkozás bárhonnan és bármikor megoldható, ugyanakkor, ha ez valamiért mégsem lenne elérhető, a folyamatos ügyelet részeként viszonylag rövid időn belül személyes jelenlétet is garantálnak. [1]

2.2. Telefonközpont

A telefon alközpont (PBX – Private Branch Exchange) – vagy egyszerűbben csak telefonközpont – a telekommunikáció egy speciális szegmense. Egy olyan berendezés, amely munkahelyek külső és belső telefon- valamint fax forgalmát biztosítja. Segítségével a cégnek egy saját, belső, vezetékes készülékekből álló telefon rendszere alakítható ki, melyen keresztül könnyen és gyorsan elérhetik közvetlen munkatársaikat. A szervezeten kívüli hívásokra pedig a cég városi - vagy fő - vonalait tudják használni, amelyeket egy telekommunikációs szolgáltató biztosítja részükre. Ezáltal minden alkalmazottnak külön hívószáma lehet anélkül, hogy mindegyikőjüknek külön fővonalat kellene előfizetni.[2]

Mindezek mellett számos kényelmi funkció válik elérhetővé a felhasználók számára. Ilyen például a hangposta beállítása, vagy az átirányítási lehetőség belső- vagy akár mobil telefonszámra is, ami kifejezetten hasznos lehet abban az esetben, hogyha valaki otthonról dolgozik. Elérhetővé válik továbbá az átkapcsolási funkció egy másik mellékszámra, vagy átvétele egy másik mellékről, feltéve, hogy az nem fogadja a bejövő hívást. Lehetőség van a mellékek egy személyes kóddal való lezárására és

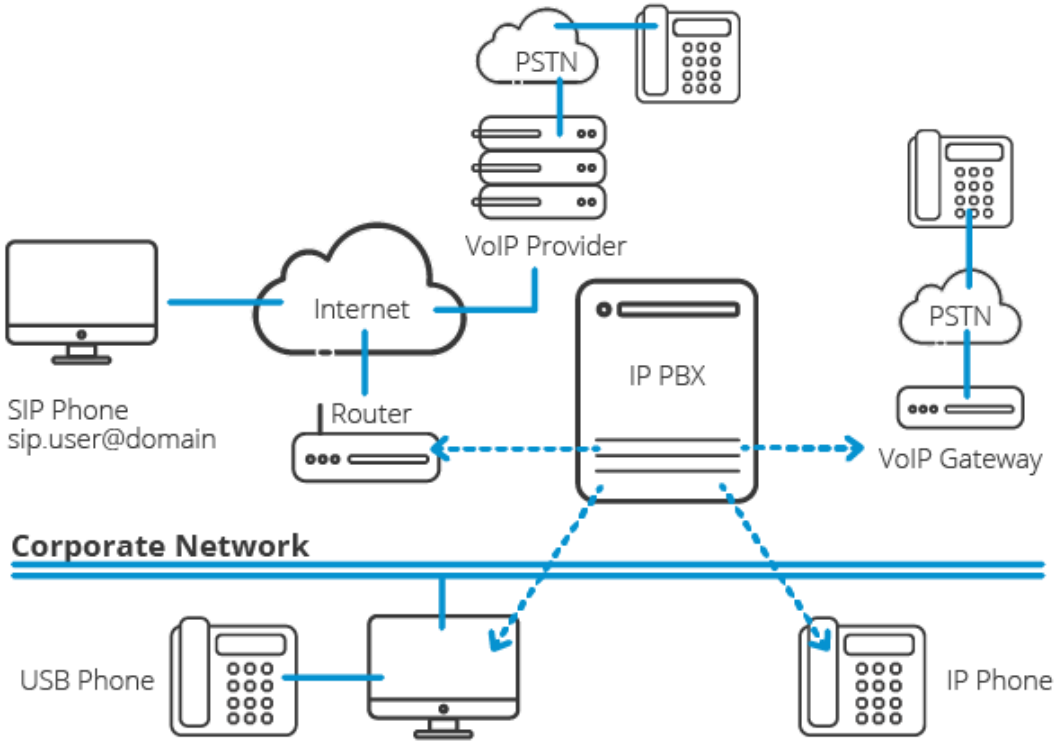
feloldására, ha nem szeretnénk, hogy bárki más használni tudja rajtunk kívül. Vannak a telefonkészüléken szabadon programozható gombok is, amelyekre beállíthatunk például gyorshívást előre beállított számokra, kezelhetjük a hívásnaplót, vagy engedélyezhetünk egy második vonalat is.

Ha az üzemeltető oldaláról vizsgáljuk meg, láthatjuk, hogy nagyon sok kontrollálási és felügyeleti lehetőséget biztosít az eszköz, ami költségmegtakarítást is jelenthet a cégnek. Ez többféleképpen is elérhető, például az emelt díjas és külföldi telefonszámok hívásának tiltása vagy időintervallumokra vonatkozó szabályok alkalmazása általános módszernek mondható. Természetesen az imént említett korlátozások akár mellékenként vagy csoportos szinten is alkalmazhatóak, így a cég működésének megfelelően több szintű jogosultságrendszer is kialakítható. Jóllehet az ellenőrzöttség tudata az alkalmazottakban már önmagában a telefonköltségek 20-40%-os csökkenéséhez vezet. [3]

Ahhoz, hogy a központ az adott cégen kívüli hívásokra is alkalmas legyen, szükség van arra, hogy valamilyen külső hálózathoz csatlakozzon. Ezen technológiák összefoglaló neve a „nyilvános kapcsolt telefon hálózat" (PSTN – Public Switched Telephone Network), amely magában foglalja a következőkben említésre kerülő megoldásokat. A külvilág és a telefonközpont kapcsolatát tekintve három főbb típust különböztetünk meg. A legkorábbi a hagyományos analóg telefonvonal, ami adottságait tekintve igencsak korlátozott. Az egyre kevésbé jelenlévő analóg technikát a digitális vonalak váltották le, ismertebb nevén az Integrált Szolgálatú Digitális Hálózat (továbbiakban: ISDN - Integrated Services Digital Network). Az ISDN nagy lépés volt akár adatátviteli sebességet, akár a felhasználási lehetőségeket tekintve, napjainkban mégis inkább az internetalapú (továbbiakban: IP – Internet Protocol) kommunikáció a legelterjedtebb. A pontosság kedvéért, ezt telefonos kontextusban inkább VoIP (Voice over IP – hangátvitel internet protokollon keresztül) protokollnak nevezik, mely magában foglal több, ennek a megvalósítására szolgáló szabványt is, mint például a SIP (Session Initiation Protocol). Az IP telefónia egyetlen hátránya, a hagyományos egyszerű telefonvonalas megoldással szemben, hogy a biztonság érdekében ezeket keresztül kell vinni különböző tűzfalakon és hálózati címfordító (NAT – Network Address Translation) eszközökön, melyek helyes konfigurálása okozhat kellemetlenségeket. Ettől eltekintve magasan az elődjei fölött helyezkedik el a jól integrálhatóságával, nagy sebességével, alacsony költségeivel és rugalmasságával. Negyedik opcióként megemlíthető még a felhő alapú telefonközpont is, ami tulajdonképpen ugyanúgy az IP-re támaszkodik, csak már hardvert sem kell hozzá vásárolni.

Ilyen módon a hívásadatok gyűjtésének módja is megkülönböztethető. Tárolás tekintetében kétféle lehetőség adott: az „online" és a pufferelt üzemmód. Az online tárolás esetében az adatgyűjtés folyamatos, tehát a telefonközpont által küldött rekordok

egyenesen a célhelyre mennek, ami lehet például egy adatbázis vagy egy erre a célra dedikált számítógép is. Pufferelt tárolásnál szükség van egy külön adatgyűjtő hardverre, ami lehet a központ saját beépített merevlemez egysége (HDU – Hard Disk Unit) is, de választhatunk egy úgynevezett adatgyűjtő dobozt is, ami bármilyen másik számítógéptől és szoftvertől független működést biztosít. Ezzel a megoldással élve nem probléma, ha a célhely valami hibából kifolyólag éppen nem elérhető, sőt általában nem is kell annak lennie, tehát leállítható/kikapcsolható, ha éppen nem használják. Ezáltal csökken a hibalehetőségek száma és nő az adatbiztonság. A tárolás formái mellett, a csatlakozás tekintetében is több opció létezik. A központból érkező, illetve az adatgyűjtő dobozban lévő adatokat soros port (RS-232 szabvány) vagy LAN (Local Area Network – Helyi Hálózat) port (más néven Ethernet port) segítségével érhetjük el. Továbbá, ha a telefonközpont saját merevlemezén lévő fájlstruktúrát szeretnénk elérni, választhatjuk az FTP (File Transfer Protocol – Fájl átviteli protokoll) kapcsolatot is a hívásadatok kinyerése céljából. Ezen adatokat tömbösítve is megkaphatjuk, alapértelmezetten beállíthatjuk, hogy napi, heti vagy havi bontásban szeretnénk hozzájutni.[4]



2.1. Ábra: A PBX kapcsolatai [4]

A fenti 2.1. ábra egy internetalapú telefonközpont általános kapcsolatrendszerét mutatja. Jól látható, hogy elválasztja a vállalat belső hálózatát a külvilágtól, valamint, hogy környezettől függően képes többféle interfészhez csatlakozni. A mellékletek között a 11.1. ábrán a telefonos alközpontok egyszerűsített és általános működése is megtekinthető.

Felmerülhet a kérdés, hogy van-e még létjogosultsága ezen rendszerek alkalmazásának, az asztalhoz kötött vezetékes telefonoknak vagy éppen a vonalakat kezelő telefonos alközpontoknak, éppen azokban az időkben, mikor már meglehetősen fejlettek a mobilhálózatok, úgymint a 4G vagy a lassan, de biztosan fejlődő 5G, és a legtöbb ember birtokában van legalább egy mobil készülék. A válasz igen, mert bár nem támogatja a mobilitást és ezzel együtt a manapság rendkívül népszerű otthoni munkavégzést sem, mégis vannak olyan munkakörök, ahol nélkülözhetetlen, hogy az ember akár egész nap telefonáljon. Ilyenek például a segélyhívószámmal rendelkező intézmények vagy nagyobb vállalatokra jellemző ügyfélszolgálatok, mint például a „help desk” vagy a „call center”, ahol saját tapasztalataim alapján akár a több tízezer hívás sem meglepő havonta. A telefonközpont alkalmazása számos olyan előnnyel jár ilyen kihasználtság mellett, melyekkel a mobiltelefon egyszerűen nem tudja felvenni a versenyt. Egy cég szempontjából a már fentebb említett felhasználói és üzemeltetői előnyökön kívül jelentős lehet a költséghatékonyság, az időtállóság vagy az, hogy könnyebben korlátok között tartható.

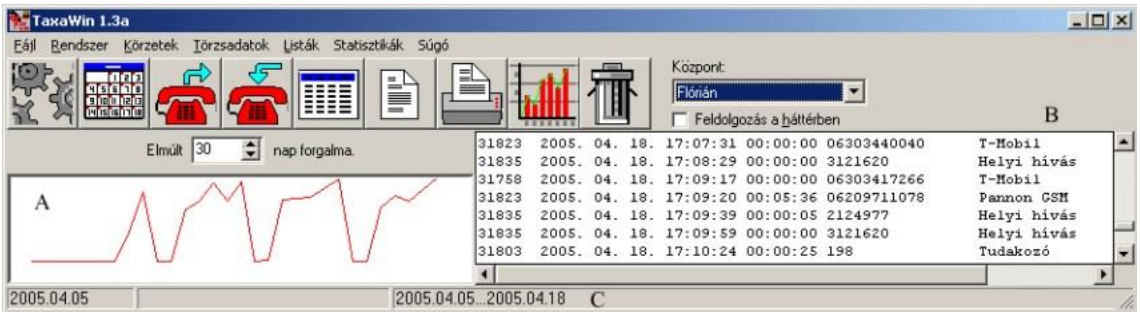
Létezik egy - a két oldal között elhelyezkedő – alternatíva, amit még nem említettem, de sok esetben megoldást jelenthet, ez pedig a „softphone”. Ez egy olyan alkalmazás, amely telefonkészülék („hardphone”) nélkül is képes hívások lebonyolítására akár számítógép, tablet vagy mobiltelefon segítségével. Ezáltal éppen a fent felvetett problémára, a mobilitásra nyújt megoldást mindamellett, hogy a hagyományos rendszerek által nyújtott lehetőségek és funkciók jelentős részét megőrzi. Természetesen ehhez is rendelkezni kell valamilyen internet alapú telefonközponttal, valamint egy kliens alkalmazással, ami a felhasználók eszközén futhat. Innentől a távoli munkavégzés már nem zárja ki a telefon használatának lehetőségét. A felhasználó teendője csupán annyi, hogy VPN-en (Virtual Private Network – virtuális magánhálózat) keresztül csatlakozzon a cég belső hálózatán lévő alközponthoz, és már használható is. Mindezek felett van egy kiemelkedő előnye az összes többi variációval szemben, mégpedig az, hogy lehetővé teszi a számítógép-telefónia integrációt (CTI – Computer Telephony Integration). Ez az ügyintézői munkafolyamatok egyszerűsítését szolgálja azáltal, hogy megkönnyíti a hívásvezérlést és az adminisztrációs feladatokat. Lehetővé teszi a valós idejű hívásinformációk megjelenítését, ilyenek például az ügyfél adatai és a folyamatos naplózás, aminek utólagos elemzése tájékoztatja a menedzsmentet többek között az egyes dolgozók teljesítményéről. [5][6]

3. MÁR LÉTEZŐ FELDOLGOZÓ RENDSZEREK ISMERTETÉSE

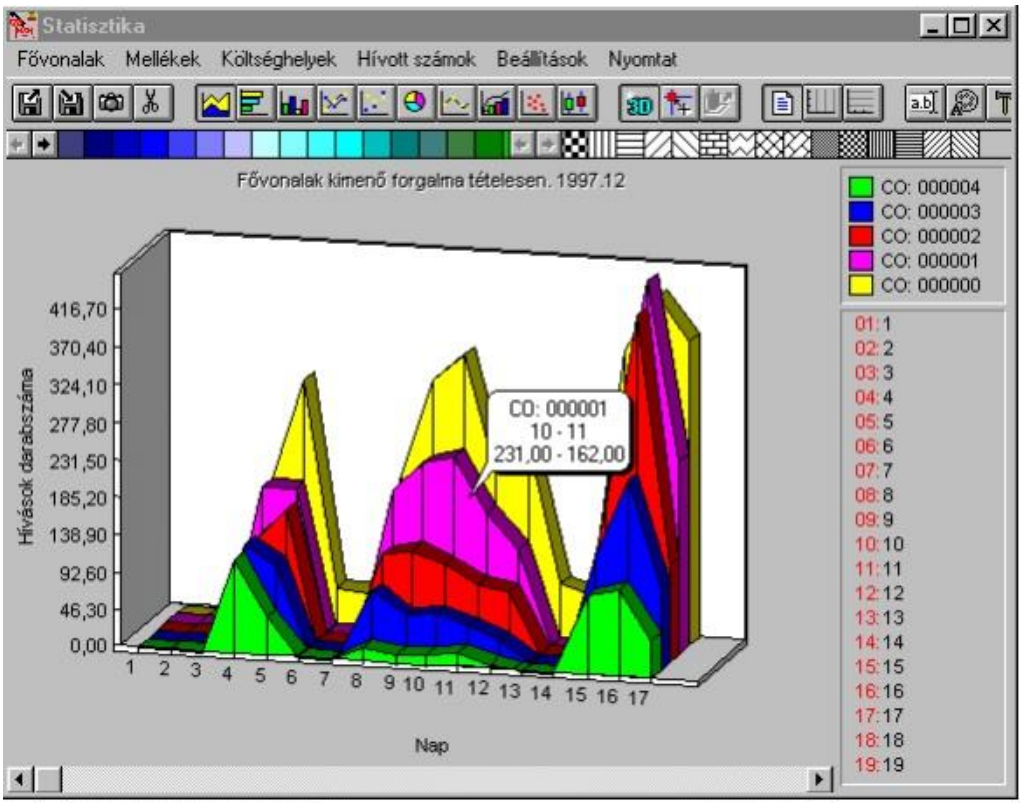
A tervezett rendszerem kidolgozásához szükségesnek gondolom felkutatni és megismerni a már létező, az enyémhez hasonló rendszereket, melyek többé-kevésbé foglalkoznak az általam is megoldandó problémákkal. Ezeket a rendszereket szeretném - a teljesség igénye nélkül - bemutatni ebben a fejezetben, kiemelve az erősségeiket és a gyengeségeiket, hogy ezáltal könnyen elhelyezhető legyen a projektem a már meglévő programok között, figyelembe véve például a megjelenítést, gyorsaságot, korszerűséget és kezelhetőséget.

3.1. Profitel – Taxawin jellemzése

A Taxawin nevű programmal szeretném kezdeni a sort, hiszen az MD Vonal Kft.-nél jelenleg ennek az 1.3-as verziója van használatban. A Taxawin egy nagyon régóta piacon lévő magyar eredetű termék, amit rendkívül széles körben használnak. Fejlesztője a Profitel Kft., ami az MD Vonalhoz hasonlóan szintén foglalkozik telefonközpont üzemeltetéssel, de kifejezetten Alcatel típusokkal. Ennek a rendszernek az első számú feladata - a hívásadat feldolgozás és elemzés mellett - a hatékony vállalati telefondíj elszámolás, melynek fő célja a költségek optimalizálása és a hatékonyság növelése. A Taxawin előnye, hogy megbízható, stabil a működése, ami leginkább annak köszönhető, hogy egy viszonylag régi és jól kiforrott program. Ez egyben a hátrányát is jelenti, mivel modern környezetbe már nehezen integrálható, nem kifejezetten gyors, valamint a megjelenése meglehetősen elavultnak mondható. Tulajdonképpen bármilyen központ adatait képes kezelni, csak első alkalommal egy úgynevezett „rekord leíró táblában” fel kell tanítani a programot karakterenként, hogy milyen formában fog érkezni az adat. Három modulból épül fel a szoftver, amelyek akár külön, dedikált számítógépeken is képesek futni, ezáltal elosztva a terhelést. A „Buffers” része felelős az adatok lekéréséért és szállításáért, a „Batch” modul végzi a feldolgozást, és van egy része egy grafikus felülettel (3.1 ábra), amelyen keresztül elvégezhetjük a beállításokat és láthatjuk a kész elemzéseket (3.2 ábra). [7][8][9]



3.1. Ábra: Taxawin 1.3 kezelőfelülete [8]



3.2. Ábra: Taxawin 1.3 Statisztika [8]

3.2. Codebase – Business Calls jellemzése

A Codebase közép- és nagyvállalati ügyfelei számára szolgáltat szoftveres megoldásokat, leginkább információfeldolgozási területekre fókuszálva. Ezek egyik szegmense a hívásadatok, amire kifejlesztették a Business Calls nevű feldolgozó rendszert, aminek a Taxawin-nel ellentétben a telefonköltség optimalizálási feladatát helyezték előtérbe. Kifejezetten az a célja, hogy elválassza az üzleti hívásokat a magán célú hívásoktól, és ezzel együtt azok költségeit is elkülönítse egymástól. Ennek kivitelezésére egy kétlépcsős rendszert alakítottak ki, amely egy dolgozói nyilatkozatból, továbbá egy vezetői ellenőrzésből és jóváhagyásból áll. Ezek alapján a megjelölt magánhívások költségei tovább számlázásra kerülnek a dolgozó részére. A felettesi ellenőrzésnek pszichikai szerepe is van a folyamatban, mivel – ahogy azt már fentebb is említettem – ennek tudata a tapasztalatok alapján költségcsökkenést eredményez. A forgalmazó szerint ezt tipikus esetben 40% körüli csökkenést jelent. Ezen kívül természetesen használható általános kimutatásokra és elemzésekre is, de általános cél a telefon költség racionalizáció a kiugró költségpontok azonosításával. [10]

3.3. Dircom – WinPCTari jellemzése

A Dircom Elektronika Kft. szintén komplex telekommunikációs megoldások értékesítésével és karbantartásával foglalkozó vállalkozás, azonban kifejezetten Panasonic típusú készülékekkel dolgoznak, és célközönségük inkább a kisvállalkozások és otthoni felhasználók. Az egyik korábbi, de még mindig forgalomban lévő terméküket a Tarifon továbbfejlesztett változataként jött létre a jelenlegi telefondíj számláló alkalmazásuk, a WinPCTari. A rendszereik méretéhez hasonlóan a feldolgozó szoftver is kisebb irodai környezetekre van szabva, ezáltal kevesebb vonal kezelését tudja csak biztosítani. Folyamatosan bekapcsolt számítógép szükséges a működéséhez, ami természetesen kiküszöbölhető egy – általuk is forgalmazott – előzőekben említett adatgyűjtő egységgel. Ezeken kívül a legtöbb dologban megegyezik a már említett rendszerekkel, ugyanúgy képes a díjak szétosztására magán hívások esetében, illetve exportálhatunk belőle különféle listázásokat, kimutatásokat. [11]

3.4. Összevetés

Mint ebből is jól látható, a célkitűzésemhez hasonló feldolgozó és elemző rendszerekből széles választék áll rendelkezésre a piacon bármilyen méretű és típusú telefonközponthoz. Bár a kutatásomban nem szerepelnek, mert igyekeztem a hazai termékek segítségével felmérni a helyzetet, de számos külföldi gyártó is foglalkozik efféle programok értékesítésével, melyek között találtam kimondottan modern, és előremutató darabokat, de a beszerzési/beüzemelési nehézségek, valamint magas áraik miatt ezeket elvetettem. Áttekintve az itt felsorolt opciókat, igazából egyik sem ideális választás az MD Vonal szempontjából. A Dircom által forgalmazott WinPCTari egyszerűen nem elég a cég ügyfélkörét tekintve, melyek között szerepelnek többezer mellékkel rendelkező intézmények, illetve nem lenne előrelépés a jelenlegi alkalmazáshoz képest. A Codebase által fejlesztett Business Calls egy ígéretesnek tűnő választás lenne annak modernségét és funkcióinak sokaságát tekintve, ám annak leginkább kiemelt előnye, a költség optimalizálás ilyen formájában nem releváns a cég számára. Szintén az ügyfélkört, valamint a mai telekommunikációs szolgáltatók konstrukcióit figyelembe véve megállapítható, hogy a legtöbb nagyvállalatnál előre rögzített vagy átalány díjas telefonvonal előfizetések vannak. Ennek eredményeként a díj számolás és a magánhívások költségének elválasztása teljesen eltűnik a szükségletek közül. A Taxawin jelenleg is használt 1.3-as verziója sem sokáig tartható meg, mert eltekintve a sebességgel kapcsolatos hátrányaitól és attól, hogy nem praktikus külön dedikált számítógépet fenntartani neki, a legnehezebben kikerülhető problémája az integrálhatóság hiánya. Ez a kiadás még Windows XP környezetben funkcionál a legjobban, és mivel a cégeknek biztonsági okokból muszáj folyamatosan lépést tartaniuk az aktívan támogatott operációs rendszerekkel, így ezzel már nem megoldható az üzemeltetés. Lehetséges lenne megvásárolni a legújabb 3.2-es frissítést a

Tawaxinhez, aminél már külön modulként egy webes felület is elérhetővé válik Taxawin netServer néven, azonban a költségeket tekintve nem reális alternatíva.[9]

Ezen gondolatokat összegezve az optimális megoldás az általam tervezett saját rendszer fejlesztése, mely teljes egészében az MD Vonal igényeihez igazítható. Elsősorban egy webalkalmazásnak tervezem, így bármikor és bárhonnan elérhető lesz, ezzel biztosítva a kellő rugalmasságot. Ezenkívül a fentebb említett érveket figyelembe véve, a díj elszámolási funkciókat ki lehet hagyni a programból, helyette kellő figyelmet fordítva a kimutatások részletességére és minőségére. Emellett igyekszem tökéletesíteni a lekérdezés optimalizálási megoldásokat és a keresési lehetőségeket a gyorsaság és pontosság érdekében. Ezeket szem előtt tartva a költséghatékony működés támogatása továbbra is célja marad az alkalmazásnak, csak más formában. A statisztikák alapján a program képes lesz rámutatni többek között a kiemelkedő eltérésekre vagy túlterhelt területekre, melyek szerint az adott cég lépéseket tehet majd az optimalizálás érdekében, például erőforrás gazdálkodás vagy munkaidő beosztás terén. Továbbá megfontolandó lehet még a szolgáltatóval kötött szerződés módosításának gondolata egy még inkább testreszabott, előnyösebb pozíció céljából.

4. FORRÁSADATOK

A dolgozatomhoz felhasznált adatbázis alapját egy szerkezetét tekintve valós, ugyanakkor tartalmilag fikcionális hívásnapló képezi. Ennek az az oka, hogy ezek a hívásadatok érzékeny információnak számítanak. Egyik cég sem teszi szívesen publikussá vagy adja ki harmadik félnek a saját adatait, hiszen ezek magántelefonszámok, így ügyfélinformációkat tartalmaznak, amiket nagyon szigorúan védenek. Az ügyfelek védelmén kívül pedig - ha valaki venné a fáradtságot és részletesebben elemezné a megszerzett adatokat - komoly üzleti kockázatot jelentene. Az így kinyert információkból visszakövethetőek lennének például belső folyamatok, külső partnerrel való kapcsolatok vagy akár periodikus tevékenységek is, amelyek mind visszaélésre adnának lehetőségek.

A fent leírt okokból kifolyólag ehhez a feladathoz egy saját programot hoztam létre, amely képes ilyen adat rekordokat véletlenszerűen generálni és szerkezetileg egy valós környezetből vett mintával teljesen megegyező, komplett hívásnaplót előállítani. Az így létrejött forrásfájl ily módon nem tartalmaz majd valós adatokat. Amennyiben ez mégis előfordulna - például egy telefonszám valóban élő és aktívan használatban van -, az csupán a véletlen műve. Bár igyekeztem minél átfogóbban megírni a generáló alkalmazást, de a hívástípusok annyira sokrétűek, hogy természetesen vannak benne apróbb hiányoságok, amelyeket nem tudtam teljeskörűen lefedni. Több mint 15 darab különböző hívás jelenik meg a kreált naplófájlban, de még így is hiányoznak belőle például a külföldi telefonszámok, a rövidített vészhívó telefonszámok, illetve a rejtett hívóazonosítóval rendelkező beérkező hívások. Tapasztalataim szerint ezek egyébként is viszonylag elhanyagolható számban vannak jelen a többi típushoz képest, tehát a logfájl értékét ez nem rontja, de érdemes továbbfejlesztési lehetőségként gondolni rá.

A dolgozatomban a véletlenszerűen létrehozott rekordok egy létező vállalat struktúráját mintázzák, annak saját belső mellékszám állományával, szervezeti felépítésével és kapcsolatrendszerével. Erre a feladatra egy kitalált bank tökéletesen megfelelő választás lesz, hiszen kellően sok divízióval és osztállyal rendelkezhet, illetve egymástól független székházakban vagy fiókokban folyik a működése. Azáltal, hogy egy ilyen céges környezeti példán keresztül mutatom be a feladatot, szemléletessé és kézzelfoghatóvá válik majd a végeredmény, értelmet nyernek az elemzések/kimutatások a webes alkalmazás tesztelése során. Ugyanakkor - mivel a nyers log fájl nem tartalmaz erre vonatkozóan információkat - ezeket külön adatbázis táblákban tárolom, és csak az adattárház végleges felépítésénél fogom összekapcsolni. Ennek következtében a későbbiekben akár egy különálló, opcionális modulnak is tekinthető a létrejött alkalmazás szempontjából.

Fontos megjegyezni, hogy ezek az adatok kizárólag a vonalas telefonok hívásait tartalmazzák, mivel csak ezek mennek keresztül az alközponton, így a mobil telefonokról indított hívásokról nincs információnk. Ezeket csak a főközpontból lehetne elérni az adott telekommunikációs szolgáltató segítségével. Továbbá érdemes megjegyezni azt is, hogy a fizikailag különböző helyszíneken lévő központokat logikailag össze lehet kapcsolni adatvonalak segítségével és ezáltal van lehetőségünk egybefüggően kezelni őket. Ezért fordulhat elő például az, hogy két egymástól távoli, akár különböző városban lévő mellékszám közötti beszélgetés belső hívásnak minősül.

4.1. Adatrekordok struktúrája

Ebben a részben, az Ericsson típusú telefonközpontok által kibocsátott nyers adatokat tartalmazó fájl egy rekordjának felépítését fogom ismertetni. Ennek elsősorban azért van jelentősége, mert ez alapján tervezem majd meg az adatbázis tábláit, mezőit és kapcsolatait. Másodsorban pedig azért is elengedhetetlen átlátni a szerkezetét, mert ezen információk alapján lehet eldönteni a hívás pontos besorolását. Az alább látható példasor jól szolgál mintaként, mert teljesen valós, logikailag helyes a felépítése, azonban fontos leszögeznem, hogy ennek több formája is létezik. A következő alfejezetben ezt is részletesen kifejtem majd, de a lényeg, hogy például a hívások irányától függően, vagy a vonal választást figyelembe véve más és más formákat ölthet egy ilyen adatsor. Némelyiknél egy oszlop üresen marad, néhol pedig mást jelent, de éppen ez lesz majd a program egyik fő feladata, hogy ezeket könnyedén felismerve tudja kezelni a beérkező adatokat. Mivel az adatbázis szempontjából ez a minta tökéletesen megfelelő és elegendő a bemutatásra, így csak ezt az egyet fogom részletezni.

00776269 191209 102954 00132 DI 1501 06301234567 03000200150770010003 2738

Szegmens	Példa	Jelentés
1	00776269	Egyedi rekord azonosító
2	191209	Dátum (éé/hh/nn)
3	102954	Idő (óó/pp/mm)
4	00132	Hívás időtartama
5	DI	CIL kód
6	1501	Hívott fél („B” szám)
7	06301234567	Hívó fél („A” szám)
8	0300020015	Bejövő trunk száma
9	0770010003	Kimenő trunk száma
10	2738	Átvevő fél

4.1. Táblázat: Rekordok szerkezete

A fenti 4.1. Rekordok szerkezete című táblázatban összefoglalva látható egy minta hívásrekord szerkezeti felépítése, és lehetséges tartalma is. A következő néhány bekezdésben ezen részek értelmezését fogom részletesen kifejteni. Az ismertetést értelemszerűen balról jobbra haladva, és egy kivételtől eltekintve a szóközök által létrejött tagolás segítségével végzem.

Az első számsor nem tartalmaz túl sok információt számunkra, csupán az adatrekord sorszáma, voltaképpen egy egyedi azonosítója a rendszeren belül. Ez egy folyamatos, egyesével növekvő számsor.

A második és harmadik szakaszt egybe venném, tulajdonképpen összetartoznak, hiszen az első az adott hívás kezdetének dátumát, a második pedig a pontos idejét jelöli. A mintát tekintve tehát ez esetben egy 2019. december 9-i hívásról van szó, amit 10 óra 29 perc 54 másodperckor kezdeményeztek.

A negyedik szakasz az adott hívás teljes időtartamát jelöli, szintén óra-perc-másodperc felbontásban, ugyanakkor észre kell venni, hogy ez csak öt darab karakterből áll. Ennek a hossza fix, nem lehet sem több, sem pedig kevesebb. Az első karakter felel meg az órának, a második és harmadik a percnek és az utolsó kettő karakter a másodpercet jelent. Ebből egyenesen következik, hogy két számjegyű órát nem képes tárolni a rendszer, tehát a legtöbb, amit meg tud jeleníteni egy ilyen rekordban, az a 9 óra 59 perc 59 másodperc. Amennyiben egy hívás ennél tovább tartana, az esetben létrejön hozzá egy újabb adatrekord egy új azonosítóval, és elölről kezdi a számolást. A minta esetében ez az időtartam csupán 1 perc 32 másodperc.

A következő egység kitűnő a sorból, hiszen ez az egyetlen, amely számok helyett betű karaktereket tartalmaz. Ez az úgynevezett CIL kód (CIL Code – Call Information Logging Code), amely segítségével beazonosíthatjuk a hívás típusát. Ez a kód állhat egy, kettő, valamint három karakterből is. A telefonközpont összesen 65 féle hívást különböztet meg alapértelmezett módon, ennek megfelelően 65 darab eltérő kód is létezik. Egy külön táblázatban egybegyűjtöttem ezeket a standard kódokat, amelyekkel az Ericsson alközpont rendelkezik, ez megtekinthető a melléklet 11.2. táblázatában. Azért emeltem ki, hogy ezek csak az alapértelmezett értékek, mert bármikor szabadon átalakíthatóak az üzemeltető által, illetve új kódok is bármikor felvehetőek ebbe a listába. Hozzá kell tennem, hogy természetesen szinte sehol nem használnak ennyi kódot, a programhoz felhasznált minta adatokban is csak a töredéke megtalálható. A példa sorunkat tekintve pedig a „DI” (Diverted Incoming call) egy átirányított bejövő hívást jelöl. Az átirányított ez esetben úgy értendő, hogy bejött a hívás egy belső mellékre, amelyről a rendszer tovább küldte egy másik belső mellékre.

Úgy gondolom, hogy a hatodik és hetedik szakaszt szintén érdemes egy kalap alá venni, mivel ezek a konkrét telefonszámok. Elől a hívott, majd utána a hívó. A hívott számot „B” számnak, a hívó számot pedig „A” számnak szokás nevezi. Ez a két rész szintén rendkívül változó karakterszámból állhat, ugyanis egészen a háromjegyű segélyhívó számoktól, mint például a 112, a tíz-tizenkét karakterből álló külföldi telefonszámokig bármi előfordulhat. Jelen példában a hívott szám a 1501 -es belső mellék, melyet a +36301234567 mobiltelefonszámról hívtak.

Az utolsó előtti számsort -bár első ránézésre húsz karakterből áll-, logikailag mégis ketté kell bontani, és kétszer tízként értelmezni. Ez azért is fontos, mert vannak esetek, amikor csak az egyik vagy csak a másik jelenik meg egy rekordban. E két számsor a központon belül definiált útvonalakat (ROU - route) és azok vonalait (TRU - trunk) jelöli, az első tízes blokk a bejövő irányt, a második blokk pedig a kimenő irányt. Esetünkben tehát az első blokk a 30-as számú route-ot jelöli, és annak is a 2-15 számú trunk-jét, amiből azt láthatjuk, hogy a beérkező Telekom-os hívószámmal rendelkező mobilhívást a rendszer ebbe az irányba küldte, és a sok trunk közül éppen a 2-15 volt szabad. A második blokk a 77-es számú irányt jelöli és annak az 1-03 trunk-jét. Észrevehető, hogy a példa rekord vonatkozásában nem történt kimenő hívás, viszont ahogy korábban említettem, a „DI” CIL kódból tudjuk, hogy egy átirányított hívásról beszélünk, ez esetben az átirányítás útvonalát jelöli a második számsor. Arra, hogy a központ mi alapján küld hívásokat adott irányokba, legtöbbször a költséghatékonyság a válasz, de egyszerű szeparációs okokból is beállítható így, a későbbi elemzések megkönnyítése érdekében.

Az utolsó része a rekordnak szintén egy telefonszámot fog jelölni, viszont ez nem minden esetben lesz megadva, kizárólag bizonyos típusú hívásoknál. Ilyenek például az átirányítás, átvétel, átkapcsolás. Itt is szerepelhet akár mellék, akár mobil telefonszám, ezért a hossza ennek is változó lesz. Ebben a mintában a 2738 azt a belső mellékszámot jelenti, amelyre a 1501 át volt irányítva.

Összefoglalva tehát ez az egy sor arról számol be, hogy 2019. december 9-én 10 óra 29 perc 54 másodperckor valaki a 06301234567-es mobil telefonszámról hívta a 1501-es melléket a 30-as útvonalon keresztül, azonban mivel azon a melléken aktív átirányítás volt bekapcsolva, a 2738-as mellékre küldte a rendszer a hívást a 77-es útvonalon, ahol fel is vették, és végül 1 perc 32 másodpercig tartott a kapcsolat köztük.

4.2. Hívások osztályozása

A hívások megkülönböztetésével részletesebben is foglalkozni kell, hiszen elengedhetetlen része a programnak, hogy helyesen kezelje ezeket. Ennek ellenére nem fogom az összes lehetőséget felsorolni, ugyanis az túl sok lenne, ráadásul egy részükkel éles környezetben nem nagyon lehet találkozni, és a mintában sem fogunk. Inkább az a célom a hívások kategorizálásával, hogy egy minimális belátást, alapvető ismereteket adjak át az olvasónak, amelyek szükségesek a feladat értelmezéséhez. Mint ahogy az előző alfejezetben is említettem, elsődlegesen a CIL kódból és az adatrekordok felépítéséből lehet megállapítani a hívás pontos fajtáját, ám ez a készülő program dolga lesz, ebben a fejezetben inkább felsorolásként ismertetném a főbb előfordulásokat.

A hívás típusokat legfelső szinten három különböző csoportba sorolhatjuk egy telefonközponttal rendelkező cégnél vagy intézménynél. Az ott dolgozók indíthatnak kimenő hívásokat, fogadhatnak bejövő hívásokat, illetve kezdeményezhetnek belső hívásokat egymás között. Ezek mind külön kódokkal vannak bélyegezve az adatrekordokban, a bejövő hívásokat „I”, a belső hívásokat „J” jelzés szimbolizálja. A kimenő hívásoknak alapvetően nincs külön jele, de ha valami mással van kombinálva, akkor általában egy „O” betűt szokott kapni. Továbbra is ezen a szinten maradva, e három csoport mindegyikét további kettőre tudjuk bontani, a sikeres és a sikertelen hívásokra. A sikereseknek nincsen külön betű jelzés fent tartva, mivel az az alapértelmezett, és csak az attól való eltérést kell jelölni. Ennek megfelelően a sikertelen hívások általában „C” jelölést kapnak a log fájlokban. Azért nevezem ezt legfelső szintnek, mert a többi típus, ami a következőkben említésre kerül, ezen csoportok valamelyikéhez rendelhető hozzá.

Egy külön kategóriát alkotnak például a csoportok hívásai. Az telefonközpont két csoport változatot különböztet meg. Az első a hívás átvételi csoport (GP – Group Pickup), amely arra szolgál, hogy a benne lévő mellékszámok egy előre beprogramozott kóddal átvehetik egymás hívásait. Ez akkor lehet hasznos például, ha éppen nincs a közelben annak a telefonnak a tulajdonosa amelyikre a hívás érkezik. Ennek a jelölése a log fájlokban tipikusan valamilyen „E” betűvel kezdődő kód lesz. Azonban az ilyen csoportok számát nem tudjuk közvetlenül hívni, ez kizárólag belső funkciókat szolgál. Nem úgy, mint a következő típusnál, ami a „Group Hunting” (GH), melynek kifejezetten az a lényege, hogy felhívhatjuk a csoport fő számát, és nem kell ismernünk a benne szereplő konkrét mellékeket. Innentől kezdve a rendszer kezeli a helyzetet és egy előre definiált szabály szerint kapcsolja a beérkező hívást egy-egy mellékre. Ez a szabály lehet például az, hogy melyik melléken telt el a legtöbb idő az utolsó hívását követően, vagy működhet a kapcsolás sorrendben is, ahol mindig tovább ugrik a következő lehetőségre, ha az adott mellék éppen foglalt, egészen addig, amig nem talál

egyet, ami szabad. Az adatrekordokban ez a variáció általában „F” betűs kóddal jelenik meg.

Egy másik jelentősebb szegmenst alkotnak az átirányított és átkapcsolt hívások, melyek szintén sűrűn használt típusok. Az átkapcsolás azt jelenti, hogy két vonalba kapcsolt telefon közül az egyik „átküldi” a hívást egy harmadik félnek, és ő kiszáll a beszélgetésből. Ez egy elég gyakori jelenség például operátoroknál vagy központi helyeken, hiszen így újra tárcsázás nélkül, gyorsan a megfelelő vonalba kerülhet az ügyfél. Ezt szokás transzfernek hívni, ebből kifolyólag a jelölésére a „T” betű szolgál. Az átirányításokat ugyancsak további két részre lehet bontani, mégpedig belső és külső átirányításokra. A belső az, amikor mondjuk egy kollégánk mellékszámára továbbítjuk a hívásokat, ezt a rendszer „D” betűvel címkézi. A külső az, amikor például távmunka miatt a belső mellékünkre érkező hívásokat a mobiltelefonunkon szeretnénk fogadni, ennek jelzésére az „X” betű szolgál.

Van még egy nagyobb kategória, ami az „M” betűt kapta, ám ez inkább egy gyűjtemény, és nem is hívás típust jelöl, hanem a központon belüli útvonal választásokat. Ennek a neve LCR (Least Cost Routing – legolcsóbb útválasztás), és az a legfőbb feladata, hogy a különböző hívásokat mindig a megfelelő irányba küldje, elsődlegesen a költségek csökkentése tekintetében. Ez akkor lehet hasznos például, mikor több szolgáltatótól is van bérelt külső vonal egy cégnél, és figyelembe vesszük azok tarifáit. Ilyen esetben érdemes figyelni, hogy mobil hívásnál melyik szolgáltató telefonszáma lett tárcsázva, esetleg Magyarországon belüli körzetszámokkal vagy más országok hívószámaival is érdemes foglalkozni.

Érdekességként még megjegyezném, hogy a sikertelen hívásoknak is több formája létezik, mivel az sem mindegy, hogy mi okozta az eredménytelen kapcsolatot. Két triviális helyzet van: az egyik, ha valaki nem vette fel a telefont és a másik, ha azért nem tudta fogadni a hívást, mert a vonal foglalt volt. Illetve van egy harmadik, nem annyira általános helyzet, mégpedig az, amikor nincs hely a várakozósorban. Ez egy speciális, csak a „group hunting” esetében használt funkció, amikor minden mellék foglalt a csoporton belül, így a hívó egy várakozósorba kerül, de ez is csak korlátozott számú vonalat képes tartani. Bár a hívó ekkor foglalt jelzést kap, mégsem elhanyagolható információ a központ szempontjából.

5. TERVEZÉS ÉS RÉSZLETES SPECIFIKÁCIÓ

5.1. A rendszerrel szemben támasztott követelmények

A feladat az, hogy az eddig használatban lévő Taxawin rendszert lecserélhessük egy újabb és hatékonyabban működő alternatívára. Ennek érdekében az MD Vonal Kft.-vel való részletes egyeztetés után, illetve saját tapasztalataim alapján megfogalmazódtak az új feldolgozó programmal kapcsolatos igények és elvárások. Az alap megoldandó problémát jobban részletezve szükség lesz az érkező adatok valamilyen formában való tárolására, majd továbbítására egy olyan alkalmazásnak, amely képes azok helyes kezelésére és értelmezésére, ezek után pedig a felhasználó által könnyen olvasható megjelenítésére. Innentől pedig két részre választható a specifikáció, mivel a gyakorlati megvalósításban teljes mértékben szabad kezet kaptam, a szükséges rendszerfunkciókat és üzleti igényeket viszont rögzítettük.

Elengedhetetlen központi része lesz az alkalmazásnak egy dashboard, melyen látszódik az összes kimutatás, diagram, valamint a főbb mérőszámok. Ezen az oldalon egyes vizualizációknál lehetőség lesz különböző szűrők beállítására, mint például mellékszám, szervezeti részlegek vagy idő intervallum, melyek használata után a dashboard automatikusan frissül a feltételeknek megfelelően.

Egy külön oldalon lehetőségünk lesz lekérdezni a nyers adatrekordokat, ahol szintén alkalmazhatunk hasonló szűrési feltételeket vagy rendezéseket állíthatunk be. Az így megkapott listákat pedig lehetőségünk lesz exportálni több választható fájl formátumban is, mint például „xlsx”, „csv” vagy „pdf”.

Mivel a telefonközpont által létrehozott bemeneti log fájl tény adatokat tárol, ezért azok módosítására, illetve törlésére egyáltalán nem lesz lehetőség a webes felületen keresztül, továbbá nem is nagyon létezik olyan használati eset, amely ezt indokolttá tenné. A mellék táblák és kiegészítő adatok esetében már más a helyzet, hiszen bármikor megváltozhat például egy személy vagy egy szervezeti osztály neve. Viszont a log fájlhoz hasonlóan a webalkalmazásban ezekre sem lesz módosítási lehetőség, mivel az ETL folyamatok biztosítják majd az adatok naprakészségét, amelyek majd az adott forrásrendszerekből kerülnek beolvasásra és frissítésre.

A fenti kritériumok és funkcionalitások alapján megállapíthatjuk, hogy itt tulajdonképpen egy döntéstámogató rendszerről és annak kialakításáról van szó. Általánosságban „a döntéstámogató rendszer (angolul Decision Support System: DSS) olyan integrált számítógépes eszközök összessége, amely döntési modellek, adatbázisok és a döntéshozó saját ítélőképességének segítségével interaktív módon nyújt segítséget a nem programozható vagy részben programozható döntések meghozatalában. A

döntéstámogató rendszerek egyrészt információt szolgáltatnak a menedzsernek rendszeres vagy speciális jelentés formájában, nagy mennyiségű adatot kezelve és feldolgozva. Másrészt modellezési képességekkel rendelkeznek, különböző matematikai, analitikai modellek segítségével előrejelzéseket, elemzéseket készítenek, javaslatokat tesznek". [15]

A következő alfejezetekben a rám bízott háttér megoldásokat és megvalósítási technikákat fogom részletesebben kifejteni.

5.2. Adatbázistervezés

Az adatbázis megoldások számbavételénél egyaránt jelen volt a hagyományos relációs adatbázis struktúra és ezzel együtt az SQL (Structured Query Language), mely segítségével műveleteket és módosításokat hajthatunk végre az adatainkon, valamint egyéb alternatívák is, mint például a dokumentumtároló adatbázis. Ez utóbbi ismertebb társaival együtt, mint például a gráf adatbázisok, oszloptárolók vagy a kulcs-érték tárolók, mind a NoSQL (not SQL / not only SQL) adatbázisok körébe tartoznak. Ezek az elmúlt években egyre népszerűbbek lettek a modern alkalmazásoknak köszönhetően, melyek új kihívások elé állították az adatbázisokat. Nagy igény alakult ki például a horizontális skálázhatóságra és ezzel együtt az elosztott működés, vagy az óriási méretű adatmennyiségek, változó adatstruktúrák és lazább kapcsolatok kezelésére. [22] A telefonközpont adatait tekintve kifejezetten nagy adatmennyiségről beszélhetünk, főleg mivel folyamatosan akkumulálódik. Jellegét tekintve pedig minden egyes rekordot megfeleltethetünk akár egy dokumentumnak is, melynek nem léteznek szoros kapcsolatai. Ezek alapján alkalmasnak találnám a MongoDB használatát, mint dokumentum-orientált adatbáziskezelő. Azonban a bemeneti log fájl felépítését figyelemebe véve, fix mezőkkel és típusokkal rendelkezik ezáltal erősen strukturált. Továbbá kompatibilitási és erőforrás problémák kizárása érdekében előnyösebbnek látom, ha relációs adatbázisoknál maradok, és a Microsoft által fejlesztett MSSQL-t (Microsoft SQL) használom a projekt megvalósításához. Ehhez alkalmazkodva, Microsoft grafikus adatbázis kezelő termékét az SSMS-t (SQL Server Management Studio) alkalmazom majd az adatok tárolására.

A készítendő adatbázis középpontjában természetesen a telefonközpont által naplózott adatok állnak, azonban az értelmezés és kontextusba helyezés érdekében egyéb táblák is jelen lesznek. Ezek a táblák vagy automatikusan generálódnak vagy az adott cég egyéb forrásrendszereiből kinyerhetőek (ideális esetben). Mivel ennek a projektnek a keretein belül nem jutottam valós adatokhoz, a log fájlhoz hasonlóan ezen táblákat is én állítottam elő. Az adatbázis táblái és mezői a következő táblázatokban láthatóak.

Mező	Típus	Leírás
id (PK)	int	a hívás, központ által generált egyedi azonosítója
date	varchar	a hívás keltezése
time	varchar	a hívás befejezésének pontos időpontja
interval	varchar	a hívás pontos időtartama
cil_code	varchar	a hívás besorolásának kódja
B_num	varchar	a hívott telefonszáma
A_num	varchar	a hívó telefonszáma
rou_in	varchar	az útirány azonosítója, amelyen a hívás beérkezett
rou_out	varchar	az útirány azonosítója, amelyen a hívás kiment
diverison	varchar	átvett/átirányított telefonszám

5.1. Táblázat: Calls tábla és mezői

Mező	Típus	Leírás
code (PK)	varchar	CIL kód azonosító (PK)
description_EN	varchar	angol nyelvű leírás
description_HU	varchar	magyar nyelvű leírás

5.2. Táblázat: CIL data tábla és mezői

Mező	Típus	Leírás
dept_id (PK)	int	szervezeti egység egyedi azonosítója
dept_name	varchar	szervezeti egység neve
cost_center	int	költséghely kódja
location	varchar	szervezeti egység helye

5.3. Táblázat: Departments tábla és mezői

Mező	Típus	Leírás
ext (PK)	varchar	mellék szám
user_id	varchar	a felhasználó egyedi azonosítója
name1	varchar	a felhasználó vezetékneve
name2	varchar	a felhasználó keresztneve
dept_id	int	szervezeti egység egyedi azonosítója

5.4. Táblázat: Directory tábla és mezői

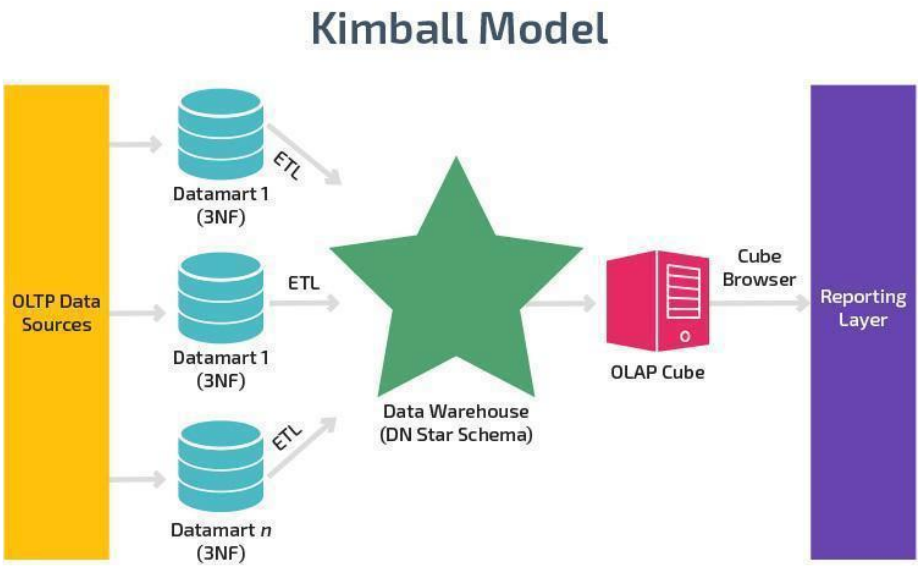
Mező	Típus	Leírás
date_id (PK)	varchar	dátum egyedi azonosítója
full_date	date	teljes dátum
day	int	nap száma
day_name	varchar	nap megnevezése
month	int	hónap száma
week	int	hét száma
quarter	int	negyedév száma
year	int	év

5.5. Táblázat: Date tábla és mezői

5.3. Adattárház felépítése

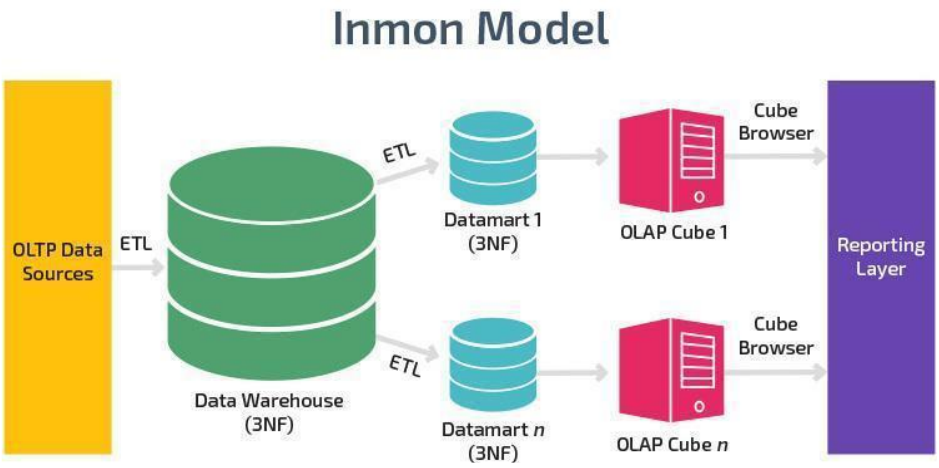
Az adattárház az üzleti támogató rendszerek lelke. Általánosságban elmondható, hogy az adattárház a számos különböző forrásból érkező adatot egészen az elemzési célú megjelenítésig elkíséri. Ezen folyamat alatt biztosítja a megfelelő tárolási formákat, valamint átalakítási lehetőségeket. Ez alapján megállapíthatjuk, hogy az adattárház tulajdonképpen adattároló és adatfeldolgozó rendszerek összessége. Az adattárházak világának két legismertebb alakja Bill Inmon és Ralph Kimball, ennek megfelelően két fő megközelítés alapján ismerhetjük meg az adattárházak felépítését.

Idézet Ralph Kimballtól: *"The conglomeration of an organization's data warehouse staging and presentation areas, where operational data is specifically structured for query and analysis performance and ease-of-use."* Kimball definíciója értelmében tehát az adattárház tulajdonképpen olyan adatpiacok összessége, melyek kifejezetten a lekérdezési, elemzési teljesítmény és egyszerű használat érdekében vannak strukturálva. Érdemes megfigyelni, hogy Kimball nem is említi a döntéstámogatást, ezzel szemben viszont hangsúlyt fektet a hatékonyságra, a felépítés és operatív műveletek gyorsaságára. Hátrányaként említhető például ezen struktúra karbantartásának bonyolultsága, és a sokszor redundáns adattárolás is, ami a normalizálás hiányának köszönhető. [16] Az elmélet vizualizációja az alábbi 5.6. ábrán látható.



5.6. Ábra: Kimball modell [14]

Idézet Bill Inmontól: *"A data warehouse is a subject oriented, integrated, nonvolatile, and time variant collection of data in support of management's decisions."* Inmon szerint tehát az adattárház egy témakör orientált, integrált, időfüggő de időben nem változó adatgyűjtemény, ami segíti a menedzsment döntéshozó folyamatit. [16] Az elmélet vizualizációja az alábbi 5.7. ábrán látható.

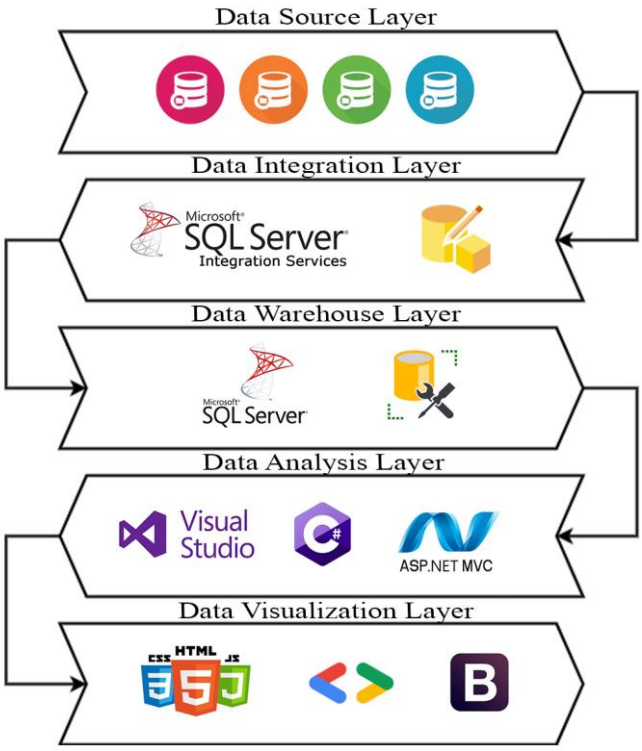


5.7. Ábra: Inmon modell [14]

Ennek értelmében az adattárház az adatok forrása, és felhasználási területek szerint tovább bontható úgynevezett adatpiacokra, melyek hatékonyan szolgálják ki az igényeket. Fontosnak tartom még kiemelni a definícióból az „időfüggő, de időben nem változó" részt. Ez azt jelenti, hogy ha a forrásrendszer adatai változnának, az adattárház a változást követi, de úgy, hogy a bent lévő adatot megfelelő időbélyeggel, érvényességi idővel látja el, majd felveszi az új állapotot is, megfelelő időbélyeggel. A bekerült adatok tehát tartósan meg is maradnak, azonban mindig csak egy adott időre vonatkoznak, ezáltal mindig nyomon követhetőek. Ezt historikus adattárolásnak nevezik és ez az egyik kiemelt előnye az adattárházaknak.

A mai modern adattárházak többsége ez utóbbi elméletet követi inkább, és én is ezt fogom alkalmazni a dolgozatomban. A 5.8 ábra egy saját készítésű kép, amely az adattárházak rétegzett struktúráját mutatja be, viszont a benne szereplő szoftverek, nyelvek és keretrendszerek kizárólag ezen projektre érvényesek. Így tehát egy rendszertervnek is tekinthető, mivel jól összefoglalja a felhasznált eszközöket és a kivitelezés fontosabb lépéseit. A forrásadatok réteg az első, ami jelen esetben főleg a telefonközpontot jelenti. Ezt követi az adat integrációs réteg, ami magába foglalja az ETL folyamatokat. Ezzel a következő fejezetben részletesebben is foglalkozom majd. Ennek eredménye tulajdonképpen egy az egyben jelenti a következő, adattárház vagy adatpiac réteget, ahol már csak a szükséges adatok állnak rendelkezésre rendezett formában. Ebből az adathalmazból dolgozik az analitikai feldolgozásért felelős réteg, amit hozzáférési rétegnek is szokás nevezni, mivel ez az a pont, ahol egyéb külső

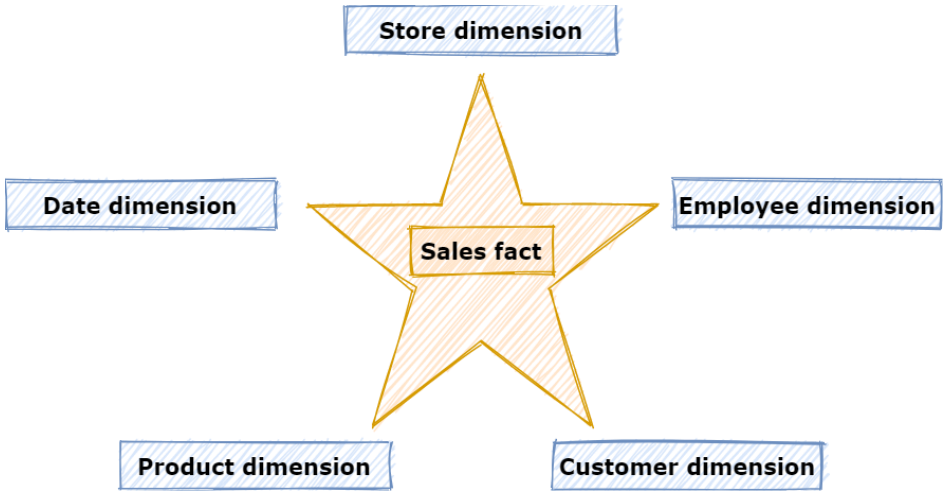
alkalmazások vagy kliensek hozzáférhetnek az előkészített adatokhoz. A következő, és egyben utolsó a megjelenítési réteg, amely a felhasználó számára készül, és fő feladata, hogy a tárolt adatokat könnyen értelmezhető módon, grafikus formában adja át. Ennek többféle felhasználási módja létezik, többek között lehet például döntéstámogató rendszer, adatbányás eszköz vagy rugalmas lekérdező program is.



5.8. Ábra: Adattárház rétegei [Saját kép]

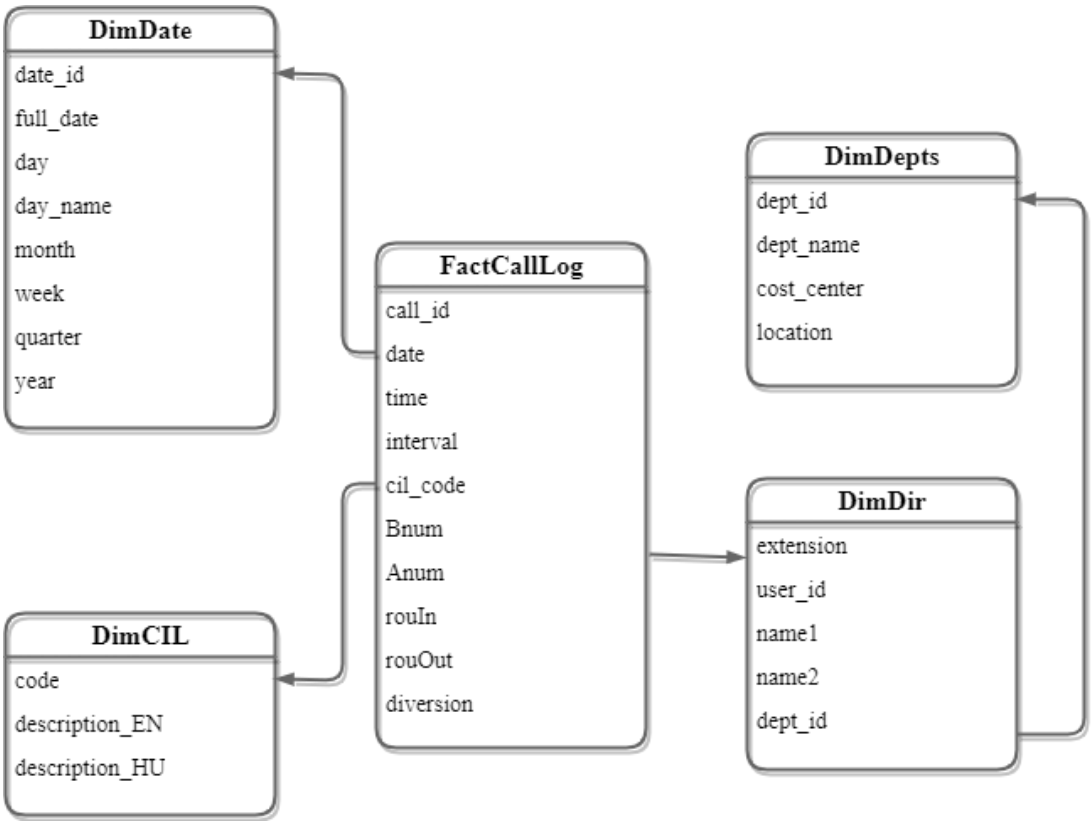
5.3.1 Dimenzionális adat modellek

Az adatmodellek felelősek az adattárházban szereplő adatok logikai szervezéséért, azok leírásáért. Az egyik fő céljuk az, hogy optimalizálják az adatbázist a gyorsabb adatelérés érdekében. A dimenzionális modell szintén Ralph Kimball nevéhez köthető, alapját a tény és a dimenzió táblák jelentik. Míg a ténytábla konkrét számszerű adatokat és referencia kulcsokat tartalmaz főként, addig a dimenziótáblák ezen adatokat helyezik kontextusba. Ezt a modellt több különböző felépítésű séma is megvalósítja, melyek közül kettőt szeretnék kiemelni. Az első és talán legkézenfekvőbb modell a csillagséma (5.9. ábra), ahol egyetlen tény táblával dolgozunk, és általában dimenziókból sincs túl sok, mindig csak azokat szükséges bele venni, amelyek relevánsak az adott felhasználási cél szempontjából, ezáltal gyors adatelérést biztosít, könnyű megérteni és karbantartani. [18]



5.9. Ábra: Csillag séma [23]

A másik ilyen séma, amit szeretnék megemlíteni, az a hópehely, ami tulajdonképpen a csillagséma egy kibővített változata. Alapjaiban ugyanazt nyújtja, csak megengedi a dimenzió táblák hierarchikus használatát, így tehát már dimenzióhoz is kapcsolhatunk újabb dimenziót. Jelen dolgozatomban ezt fogom használni. Ezek alapján az 5.10. ábrán látható a tábláim tervezett kapcsolatrendszere.



5.10. Ábra: Hópehely séma – Saját adatbázis

5.3.2 SCD

Röviden fogalmazva a Slowly Changing Dimension egy olyan technika, amely segítségével nyomon követhetjük az adattárházban történt változásokat. Alapvetően ez valósítja meg a korábban már említett „időfüggő, de időben nem változó” adatok kezelését a gyakorlatban, azonban ennél sokkal többre is képes. Több típusát tartjuk számon, melyek más és más igények kielégítésére szolgálnak. Nézzünk meg ebből párat:

- **SCD Type 0:** Nem csinálunk semmit a betöltést követően, azaz nem módosítjuk a meglévő rekordokat csak újakat töltünk a meglévőkhöz. Ezt nagyon ritkán szokták használni.
- **SCD Type 1:** Egyszerűen felülírjuk a meglévő adatokat az új adatokkal. Ezáltal az adattárház naprakész adatokat tartalmaz, ám nem tudjuk visszakövetni a változásokat.
- **SCD Type 2:** Amikor a tábla tartalmaz olyan extra oszlopokat, ami biztosítja a historikus adatok meglétét minden egyes frissítéskor, így visszanézhetjük, hogy melyik adat hogyan változott az idők folyamán.
- **SCD Type 3:** Új oszlopokat adunk a táblához úgy, hogy az egyikben a jelenlegi érték, a másikban pedig a historikus érték kerül tárolásra. Ha változik egy adat, akkor ugyanazon a soron a meglévő oszlopértéket átírjuk. Amit átírtunk adatot pedig bekerül a historikus értéket tároló oszlopba. Láthatjuk, hogy ez egy limitált lehetőség, hisz csak annyi szinten tudjuk nyomon követni a változást, ahány különböző oszlopot hoztunk létre. Ha csak egy jelenlegi és egy múltbéli oszlopunk van, csak ezt a két állapotot fogjuk tudni eltárolni.

[24]

Bár fontos képessége a historizálás az adattárházaknak, ebben a projektben én mégsem fogom használni, mivel az adatok jellege ezt nem indokolja, viszont SCD Type 0 és 1 verzióját többször alkalmazom majd az ETL folyamatok átalakítás rétegénél. A tény adatok esetében Type 0-át használok majd, hiszen a hívásokat tekintve sosem lesz két egyforma, ráadásul a rendszer szempontjából az a jó, minél több adat rendelkezésre áll. A dimenzió táblák esetében Type 1-et használok majd, mert nem fontos ismernünk a változásokat, csak az aktuálisan elérhető adtok számítanak.

5.4 ETL

Az ETL, ami az Extract, Transform és Load (5.11. ábra) fázisokat jelöli, egy olyan folyamat, amelynek keretein belül az adatot különböző átalakításokon átvezetve eljuttatjuk a forrás rendszerekből a célrendszerbe. Erre azért van szükség, mert számos eltérő forrásból származhatnak az adatok, más-más szerkezetben, melyek konzisztenciáját ezáltal biztosítani tudjuk. Így egy strukturálatlan adathalmazból a célrendszerbe már olyan formában érkeznek, ami hasznos lehet a további üzleti folyamatok számára. Ennek megoldására tehát az adatok átalakítására, rendezésére, illetve manipulálására az SQL Server Data Tools (SSDT) nevű, a Visual Studio egy kiegészítő eszközét fogom alkalmazni, amely használatával elérhetővé válik az SQL Server Integration Services (SSIS) platform. Az SSIS egy „adatmozgató” eszköz, ennélfogva alkalmas az adatok importálására és exportálásra a kezdetleges adatbázis és az adattárház között, valamint az ETL folyamatok megtervezésére és végrehajtására is tökéletes.

5.11. Ábra: ETL folyamat [17]

5.4.1. Extract

Az adat kinyerés az ETL folyamat első lépése. Feladata az, hogy a már korábban is említett számos forrásból összegyűjtse az adatokat egy úgynevezett „stage / staging” adatbázisba, ahonnan majd a következő réteg képes lesz dolgozni. Az adatok a gyűjtés során áteshetnek egy előszűrésen, ami alapján a sérült, hibás vagy nem oda illő adatokat ki lehet selejtezni. Az egyszerűség érdekében itt még az adatok típusával sem kell foglalkozni, az a legjobb, ha minden egységesen van kezelve, például szövegként. Ezen felül érdemes még megemlíteni, hogy az adatok bekerülésének módjáról is dönthetünk. Ez történhet előre ütemezetten, fixen beállított időközönként vagy esemény vezérelt

módon, ahol egy bizonyos mező vagy rekord megváltozása idézi elő a folyamat lefutását. Folyhat valós időben is, ahol bármilyen változás azonnal beíródik az adatbázisba. Beállíthatjuk azt is, hogy minden alkalommal az összes adat íródjon felül, ez az úgynevezett „Full extract”, vagy csak a változtatások kerüljenek módosításra, ez pedig az „Update”.

5.4.2. Transform

Az átalakítás a legfontosabb lépése az ETL folyamatnak, itt történik a valódi munka. Ebben a fázisban a stage adatbázisban lévő adatokat formálhatjuk úgy, hogy a végére egységes sémába illeszkedjenek. Ilyen átalakítások közé tartozik például az üres cellák kezelése, mely általában valódi „NULL” értékkel való helyettesítéssel történik. Itt állíthatjuk be az adatok valódi típusát is, figyelve akár a dátum formátumokra, karakterkódolásokra és nyelvi különbségekre. Ebben a szakaszban lesznek átvezetve az adatok az SCD-n is, amivel többek között biztosítható, hogy elkerüljük a redundáns adattárolást, ezen felül alkalmazhatunk külön duplikáció szűrést is. Végső lépésként kialakíthatjuk a táblák közötti kapcsolatokért felelős elsődleges és idegenkulcsokat is, ezzel előkészítve a sémák kialakítását a következő rétegben.

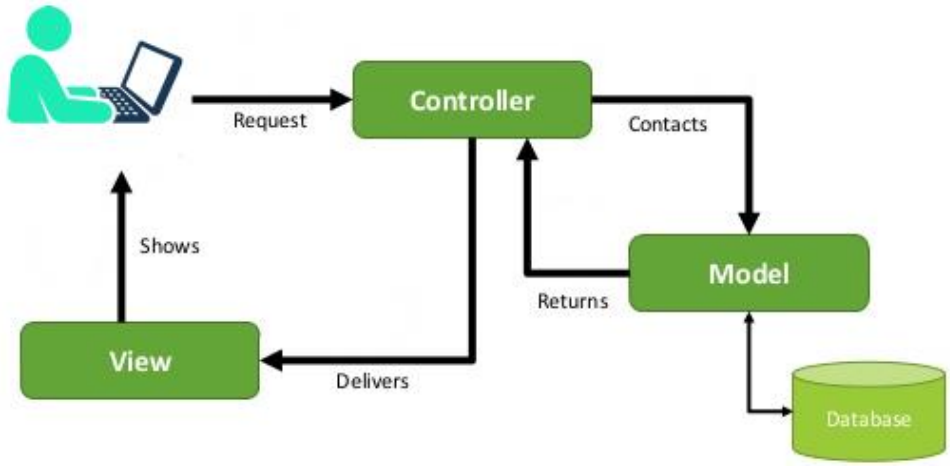
5.4.3. Load

Az ETL folyamat utolsó lépése a betöltés, ahol az átalakítás rétegben létrehozott rendezett és tisztított adatokat áttöltjük a végleges célhelyre, egy adattárházba. Itt alakul ki az előre megtervezett séma, ezáltal az adatok végleges logikai rendezettsége. Talán ez a folyamat legegyszerűbb lépése, itt már csak arra kell figyelni, hogy a tervezésnek megfelelő sorrendben kerüljenek át az adatok. Először a dimenziók, figyelembe véve az esetleges hierarchiát, majd végül a tény tábla.

5.5. Feldolgozó modul megtervezése

Az applikáció tervezésénél már az üzleti igényeket is komolyabban figyelembe kell venni, mivel itt már nem csak háttér folyamatok zajlanak majd. Gondoskodni kell az adatok helyes felhasználásáról, és a program optimalizált működéséről, ugyanis - bár közvetett módon, de - a felhasználó valójában ezzel fog interakcióba lépni. Mint azt már korábban említettem, a webapplikációt főként az elérhetőség szempontjából választottuk az asztali és telefonos alkalmazásokkal szemben, azonban ezen belül is több lehetőség áll rendelkezésemre. Számításba vettem webfejlesztéshez használt közkedveltebb programozási nyelveket, mint például Java, Python vagy C#, de a projektem szempontjából nincsenek szignifikáns különbségek közöttük. Ebből

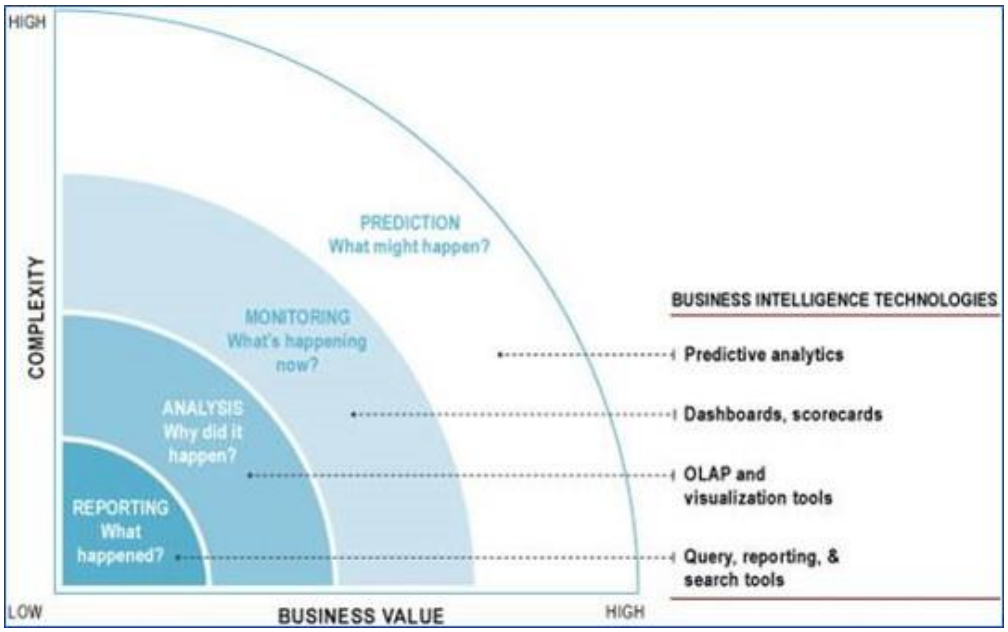
kifolyólag az alapján döntöttem, hogy melyik illik a legjobban az eddig meglévő komponensekhez, valamint hogy melyikben van a legszélesebb körű ismeretségem, így az alkalmazást C# nyelven implementálom. A választott programnyelvnek teljes támogatottsága van a Microsoft Visual Studio-ban, ezért ezt az IDE-t (Integrated Developer Environment – integrált fejlesztői környezet) használom majd. Ebben több beépített fejlesztői eszköz és kód ellenőrzés is rendelkezésre áll, ami a fejlesztés felgyorsítását eredményezi. Mi több, ezáltal elérhetővé válik az ASP.NET Core platformfüggetlen keretrendszer. Az ASP (Active Server Page) a Microsoft által jegyzett technológia, amely a dinamikus weboldalak készítését lehetővé tévő osztályok és komponensek együttese. Ezen keretrendszeren belül is az MVC (Model – View - Controller) tervezési mintát fogom alkalmazni, miszerint minden webalkalmazás három jól elkülönülő rétegre bontható. Ez a három réteg bár kapcsolódik egymáshoz, működésük mégis független, így akár modulokként is tekinthetünk rájuk, amiket szabadon alakíthatunk, cserélhetünk. A model réteg leíró példányaiban tárolhatjuk a view-ban később felhasznált adatokat. A view egy nézet, ami meghatározza, hogy milyen formában jelennek meg a model-ben szereplő adatok. A felhasználó is a view réteget látja, ezzel áll közvetlen kapcsolatban. Az előző kettőt pedig a controller köti össze, aminek legfőbb feladatai közé tartozik, hogy válaszoljon a felhasználói kérésekre, módosításokat hajtson végre a modellen vagy ellássa a view-t adatokkal, tehát, hogy foglalkozzon az üzleti logikával és vezérelje a működést. [12] Összefoglalva a 5.12. ábrán látható az MVC architektúra felépítése.



5.12. Ábra: Az MVC tervezési minta működése [13]

5.6 Adatvizualizáció

Kizárólag a számokat nézve nem könnyű felfedezni az összefüggéseket, viszont ha vizuális formátumban tekintjük át az adatokat, olyan mintákat, kapcsolatokat fedezhetünk fel, amelyek egyébként észrevétlenek maradnának. Általánosságban elmondható, hogy az adatvizualizáció egy hatékony eszköz információk megosztására, átadására, és segít meglátni a számok mögöttes értékét. Az adatvizualizációkkal megjeleníthető a teljesítmény, közvetíthetők a trendek, kiemelhetők a kapcsolatok. Interaktív jelentések, grafikonok, diagramok és más vizuális eszközök segítségével a vizualizációk lehetővé teszik, hogy a felhasználók gyorsan és hatékonyan megtalálják a hasznos üzleti információkat. Ha az adatok által közvetített információ jól átlátható, akkor hatékonyabb döntéshozatalt, tervezést, stratégiákat és műveleteket eredményező elemzéseket készíthetünk. [19]

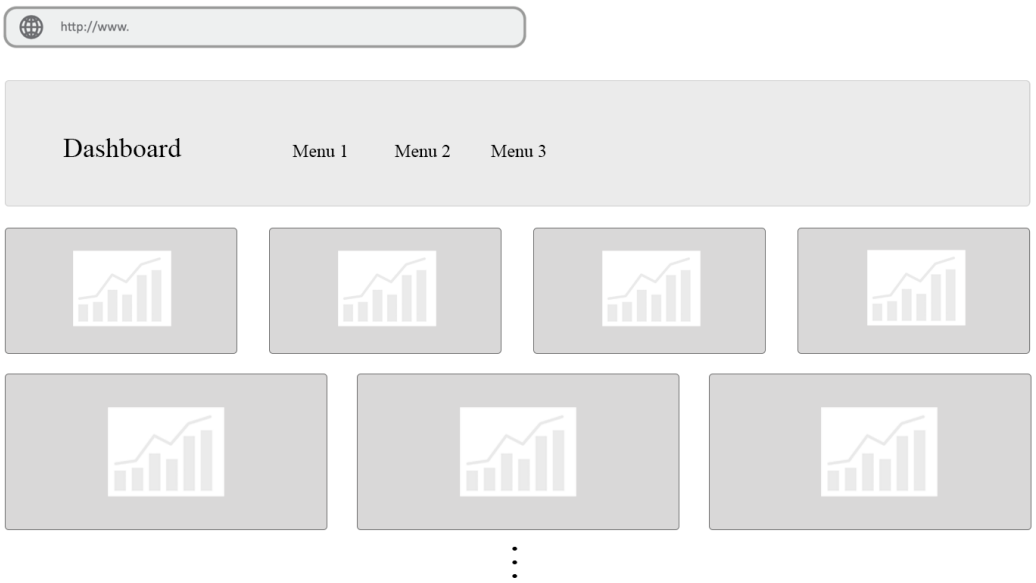


5.13. Ábra: Vizualizációk típusai [25]

Az ilyen adatábrázolásoknak céljukat tekintve több típusa létezik, aminek megfelelően változik a komplexitásuk és az üzleti értékük. Az 5.13 ábra alapján ezeknek négy főbb változatát különböztetjük meg. Megjeleníthetőek rajtuk riportok, analitikai vagy valós idejű adatok, fontosabb KPI-k (Key Performance Indicator - fő teljesítménymutató), és még az ismert adatok alapján levonható következtetések, előrejelzések is. A vizualizációk látványa is sokszínű lehet, alkalmazhatunk hagyományos elemeket, mint például az oszlop, kör vagy vonal diagrammok, kiemelhetünk mérőszámokat, valamint rendelkezésünkre állnak térképen való megjelenítési lehetőségek is. Játszhatunk a színekkel, méretekkel és formákkal is, amik segíthetik az átláthatóságot, kiemelhetik a lényegesebb dolgokat.

Az adatábrázolás ebben a projektben a webalkalmazás feladata lesz. Ezt az előző bekezdésben részletezett program foglalja magában, némi „külső” segítséget alkalmazva, például a diagrammok megjelenítése területén. A Google Chart egy JavaScript alapú könyvtár, ami teljeskörű .NET támogatással rendelkezik, és amivel könnyen kezelhető, látványos vizualizációkat hozhatunk létre a saját adatainkkal feltöltve. Ezen kívül a webes felület dizájnjához Bootstrap keretrendszert használok. A Bootstrap egy nyílt forráskódú, ingyenes webes fejlesztői keret, amely számos előre elkészített felhasználói felület komponenst tartalmaz, mint például menük, gombok, űrlapok. Ezeket főként HTML és CSS kódok alkalmazásával állítja elő, de néhány elemnek JavaScript része is lehet. [21]

Célom az, hogy kialakítsak egy hatékony dashboard felületet a cég elvárásainak megfelelően, ami képes különböző beviteli mezők segítségével alapvető szűrések elvégzésére. A dashboard-nak nem létezik igazán jól használható, kellően kifejező magyar megfelelője, de egy lehetséges definíció alapján azt mondhatjuk, hogy egy olyan egyoldalas, valós idejű használói felület, amely grafikai megjelenítéssel mutatja be az adott szervezet működése szempontjából kulcsfontosságú teljesítménymutatóinak aktuális állapotát, ezáltal lehetővé téve az információra épülő azonnali döntéshozatalt. [20] A következő, 5.14 ábrán egy ilyen dashboard-nak a látványtervét alakítottam ki, amilyet szeretnék létrehozni, persze jelenleg csak helykitöltő szövegekkel és ábrákkal, de a hangsúly a felépítésén van.



5.14 Ábra: Dashboard concept art [Saját kép]

5.7 Fejlesztői szoftverkörnyezet

Ebben a fejezetben a lehetőségek számbavétele és a tervezés során több szoftvert is megneveztem, amelyek kiválasztásra kerültek, és amelyekkel dolgozom majd. Ezeknek megléte és működése nélkülözhetetlen a projekt szempontjából, így az átláthatóság érdekében ismételten összefoglalom őket.

Az adatok tárolásáért a forrásoktól kezdve az adattárházig az SQL Server Management Studio lesz a felelős, amiben MSSQL nyelvet használok.

Az adatbázisok közötti mozgatásához és az adatok átalakításához, tehát az ETL folyamatokhoz Microsoft Visual Studo-t, azon belül pedig SQL Server Integration Services platformot alkalmazok.

Maga a feldolgozó program, a webalkalmazás szintén Microsoft Visual Studo-ban készül, ASP.NET Core keretrendszerben, MVC tervezési mintát követve. Az implementáláshoz a C# nyelvet választottam. A vizualizációk elkészítése ugyancsak ebben a fázisban történik meg, amihez Google Chart API-t veszek igénybe. A megfelelő megjelenítés kialakításához Bootstrap keretrendszert használok. Ezen adatábrázolások megjelenítésére bármilyen web böngésző alkalmas, de a tesztelés során Google Chrome mellett döntöttem.

Mivel a legtöbb fent említett fejlesztői eszköz és alkalmazás Microsoft termék, ennek köszönhetően kifejezetten jó az egymással való kompatibilitásuk, ez pedig jelentősen megkönnyíti a munkavégzést és felgyorsítja az adatáramlást.

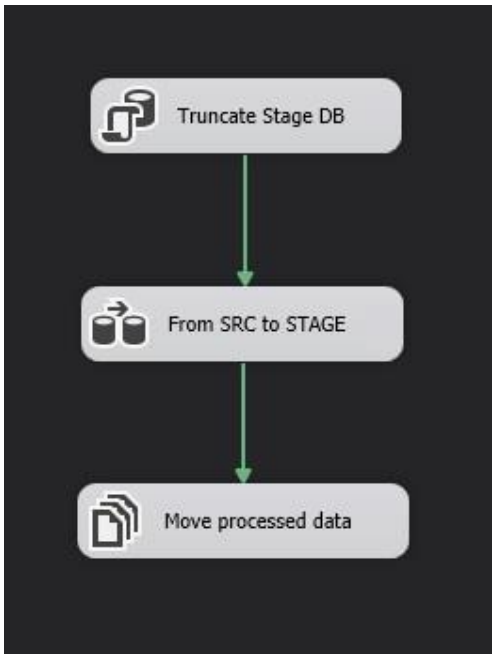
6. IMPLEMENTÁCIÓ

A megvalósítás lépései tökéletesen követik a tervezésben látottakat. A különböző részek egymásra épülése miatt ugyanazokon a fő fázisokon kellett végig mennem és ugyanabban a sorrendben. Természetesen az időközben felszínre került apróbb hibák kijavítása érdekében néha szükséges volt egy-egy visszalépésre vagy adat generálásra esetleg újra töltésre.

6.1 Adatok előkészítése

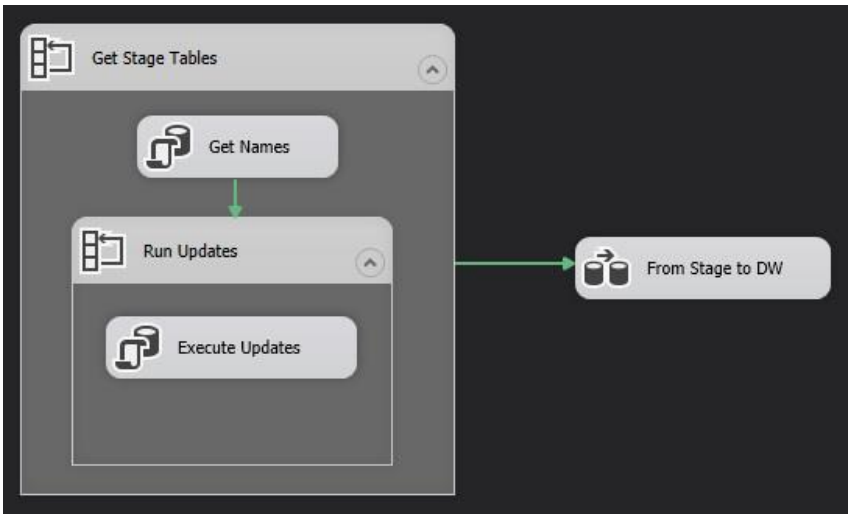
A feladat legelső lépéseként létrehoztam az ETL folyamatokhoz szükséges adatbázisokat. Majd a beérkező adatok szerkezetének elemzése után, azoknak megfelelően kialakítottam az adatbázisok tábláit a szükséges mezők, típusok és megszorítások beállításával. Miután minden készenállt az adatok fogadására, elkezdtem összerakni az ETL lépések folyamatait. Az adatbázisok elnevezései az egyszerűség kedvéért minden esetben tükrözik, hogy melyik réteghez tartoznak. Az első réteg, az Extract kimenete a már tervezés során említett „STAGE" adatbázisba kerül. Innen a Transform réteg átalakításai után az adatok átkerülnek a „DW" (datawarehouse) adatbázisba. Végül az eredményként kapott adatok a Load rétegben betöltődnek a végleges helyükre a „DM" (data mart) nevű adatbázisba. A feladat során több különböző méretű teszt fájlt is használtam, néhány százezres rekordszámtól egymillióig terjedően. A tapasztalat az volt, hogy a konvertálási és SCD folyamatok, valamint az adatbázisok közötti mozgatás erőforrás igénye meghaladta a jelenleg rendelkezésemre álló tesztkörnyezet képességeit, így kifejezetten sokáig tartottak. Természetesen éles vállalati környezetben teljesen máshogy alakulna, mivel általában sokkal több az erőforrás. Másrészt pedig az adatok folyamatos, rendszeres feldolgozása mellett, valószínűleg sosem lenne példa ekkora mennyiségű, egyszerre beérkező adatra.

Az adatkinyerés folyamata a 6.1. ábrán látható. Minden esetben első lépésként kiüríti a teljes stage adatbázist, amire azért van szükség, mert ez mindig csak az éppen feldolgozandó adatok ideiglenes tárolására szolgál, melyekre a továbbiakban nem lesz szükségünk. Ezek után a rendelkezésre álló összes forrásfájlból vagy egyéb adatbázisokból összegyűjti a friss adatokat és elhelyezi az adatbázis megfelelő tábláiba. A folyamat végén, minden beolvasott forrásfájlt áthelyez egy erre elkülönített helyre - például archiválási célból, illetve, hogy ne keveredjenek a jövőben beérkező új adatokkal.



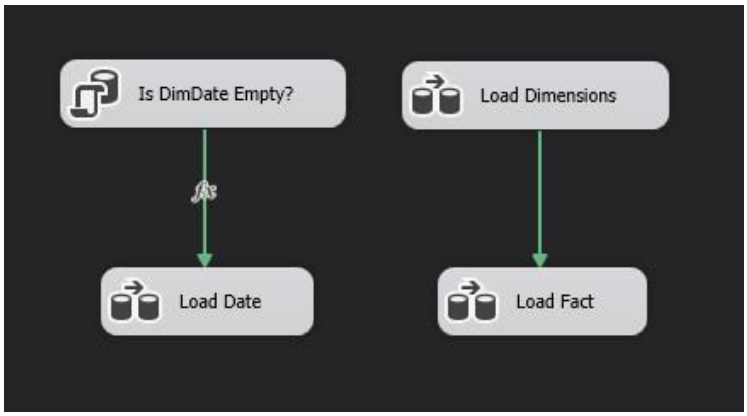
6.1. Ábra: Extract folyamat [Saját kép]

Miután a stage adatbázis feltöltődött, következhet a 6.2. ábrán látható átalakítási folyamat. Forrásrendszerektől függően előfordulhat, hogy az esetlegesen üres cellák valójában egy szöveges „null” értéket tartalmaznak. Mivel a későbbiekben a feldolgozás során ez gondot okozhat, ezért ezt szükséges kezelni. Két ciklus segítségével bejárom az összes tába összes oszlopát, és az említett eseteket felülírom valódi NULL értékekkel. Ezt követően a 6.2. ábrán látható „From Stage to DW” adatfolyam vezérlő belső műveletei futnak majd le, melyek az adatok stage adatbázisból való kiolvasásával kezdődnek. Ebben a lépésben kapja meg minden mező a neki megfelelő adattípust, így a sima szöveges adatokból például valódi dátum, valódi szám formátum lesz. Ezek után átvezetem őket a Slowly Changing Dimension (SCD) vezérlőn, amivel biztosítom, hogy a változó adatok mindig csak felülíródjanak, és ne történjen felesleges újra betöltés, ezzel elkerülve a duplikációkat. Tapasztalataim alapján ebben a részben kiemelten foglalkozni kell például a helyes karakterkódolásokkal vagy dátum formátumokkal is, mert apróbb eltérések nem várt hibákat okozhatnak, melyek kiküszöbölésére további konverziókat alkalmazhatunk. A folyamat legvégén az átalakított és korrigált adatok beíródnak a Transform réteg kimeneteleként szolgáló adatbázisba, ahonnan a következő, Load réteg már eléri őket, és megkezdheti a munkát.



6.2. Ábra: Transform folyamat [Saját kép]

Az adatelőkészítés utolsó lépéseként a 6.3. ábrán szereplő folyamatok futnak le. A dátum adatok létezésének ellenőrzésével kezdődik az egész. Mivel a dátum dimenzió tartalma állandó, így elég csak akkor betölteni a végleges adatbázisba, amikor nem létezik vagy üres. Ez olyan esetekben fordulhat elő, hogyha első alkalommal fut az alkalmazás, vagy ha valami okból kifolyólag előzetesen törlésre került. Ezzel párhuzamosan elindul a többi dimenzió tábla feltöltéséért felelős vezérlő is. A Load réteg különlegessége, hogy bár magukon az adatokon nem hajtunk végre további változtatásokat, a szerkezetükön igen, ugyanis itt alakíthatjuk ki a számunkra és az adott feladatnak megfelelő sémákat. Ez azt jelenti, hogy a tény és dimenzió táblákba már csak azok az adatok mennek tovább, amelyek valóban szükségesek és odaillőek. A létrejött adatpiacra támaszkodva egy elő aggregációs lépésként létrehoztam néhány kulcsfontosságú nézetet, amelyeket a program felhasználhat majd. Ennek az előnye, hogy a nézetek mindig a friss adatokat mutatják, és ezáltal megelőzhető néhány bonyolultabb lekérdezés feldolgozó programból való futtatása.

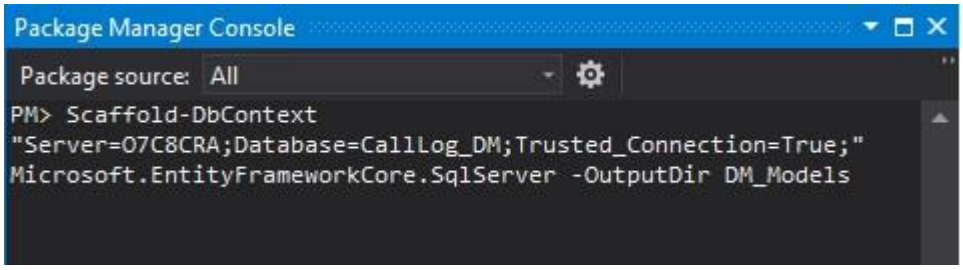


6.3. Ábra: Load folyamat [Saját kép]

Ebben a fejezetben részletezett lépések lényegesen több folyamatból állnak, mint amit itt módomban áll képek formájában bemutatni, ezért a mellékletben csatoltam néhány további ábrát (11.3; 11.4; 11.5;) róluk.

6.2 Adatok feldolgozása

Az adatpiac és a nézetek elkészültével hozzákezdhettem a webalkalmazás fejlesztéséhez, aminek az első mozzanata az adatbázis elérése volt. Mivel a platformfüggetlen .NET Core 5.0 keretrendszert használom, így először az ezzel kompatibilis Entity Framework Core objektum-relációs leképező keretrendszert is telepítenem kellett NuGet csomagok formájában. Ezek után a PMC (Package Manager Console - Csomagkezelő konzol) segítségével lefuttattam egy bizonyos „Scaffold-DbContext” parancsot (6.4 ábra) amivel létrehoztam a kapcsolatot az SQL szerverrel. Ezt követően elérhetővé váltak az adatbázisból leképezett táblák és adatok C# objektumok formájában.



6.4 Ábra: Scaffolding [Saját kép]

Erre azért volt szükség, mert „Database First” megközelítést alkalmaztam, tehát egy már meglévő adatbázist használtam fel a programban. Ahhoz, hogy az alkalmazás is ismerje és képes legyen kezelni ennek az adatbázisnak a tartalmát, sajátkezűleg kell létrehoznunk az adatbázist mintázó osztályokat, vagy alkalmazhatunk „reverse engineering” technikát. Ez lenne az fentebb említett „scaffolding”. Ez utóbbi használatával természetesen rengeteg időt és energiát takaríthatunk meg, így én is ezt választottam. Ennek az a lényege, hogy feltérképezi a megadott adatbázist, visszafejti a felépítését, majd az adatbázisséma alapján leképezi, létrehozza a megfelelő C# osztályokat modellekként. [1]

Miután az adatbázis elérhetővé vált a program számára, elkészítettem az MVC tervezési mintának megfelelően a további szükséges modelleket, valamint a kontrollereket és a hozzájuk tartozó nézeteket. A megjelenő felületek kialakításához Bootstrap-et, az adatok megjelenítéséhez egyszerűbb szűréseket, az interaktív diagrammok kirajzolásához Google Chart-ot, a táblázathoz pedig DataTables bővítményt használtam.

6.3 Adatok megjelenítése

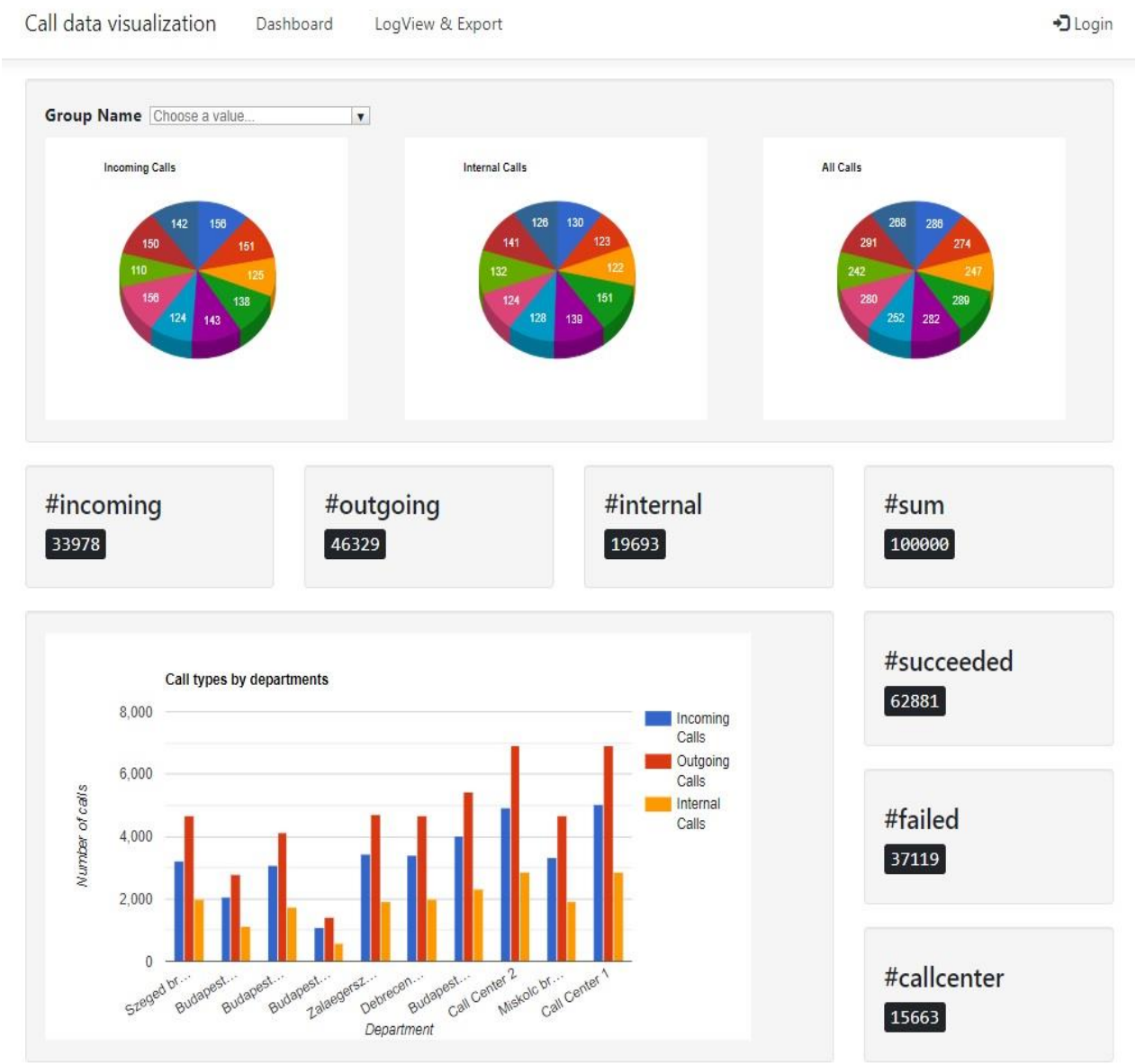
Az összes, eddig ismertetett adatmanipuláció, háttérfolyamat és szoftverfejlesztés azt szolgálja, hogy az ebben a fejezetben bemutatásra kerülő adatvizualizációk létrejöhessenek. Ez a webes felület az, amivel a végfelhasználó kapcsolatban áll és kommunikál. Tulajdonképpen úgy is tekinthetünk rá, mint egy interfészre az adatok és a felhasználó között. Ezzel a szakdolgozatom elején kitűzött cél megvalósult. A kezelőfelület a követelményeknek megfelelően két különálló részre van bontva.

Ennek az első modulja egy dashboard (6.5. és 6.6. ábra), amelyen az adatok szűrt és vizualizált formában találhatóak. Itt jelenleg néhány fontosabb, az általános üzemelés ellenőrzéséhez és a döntések támogatásához alapul szolgáló mérőszám, illetve lényegesebb osztályok, csoportok működését kifejező diagram és grafikon látható. Ezek közül is kiemelném azt a pár darabot, amelyeknél szűrési lehetőségek állnak rendelkezésre. Ezeknél többek között egy lenyíló menüből választhatunk ki bizonyos értékeket vagy egy csúszka segítségével állíthatunk be időintervallumot, melyek alkalmazásával tudjuk szűkíteni a megjelenő adatok körét. Kiemelnék még egy grafikont, ami nem a jelentősége, hanem fejlesztési szempontból érdekes. A térképes vizualizációnak ugyanis a Google Chart által biztosított JavaScript funkciókon kívül még egy Google Maps API elérésre is szüksége van, amit egy API kulccsal biztosíthatunk számára. Csak ennek a kulcsnak az átadása után lesz képes „Goecoding” segítségével azonosítani és megjeleníteni a szövegként megkapott helységneveket a térképen.

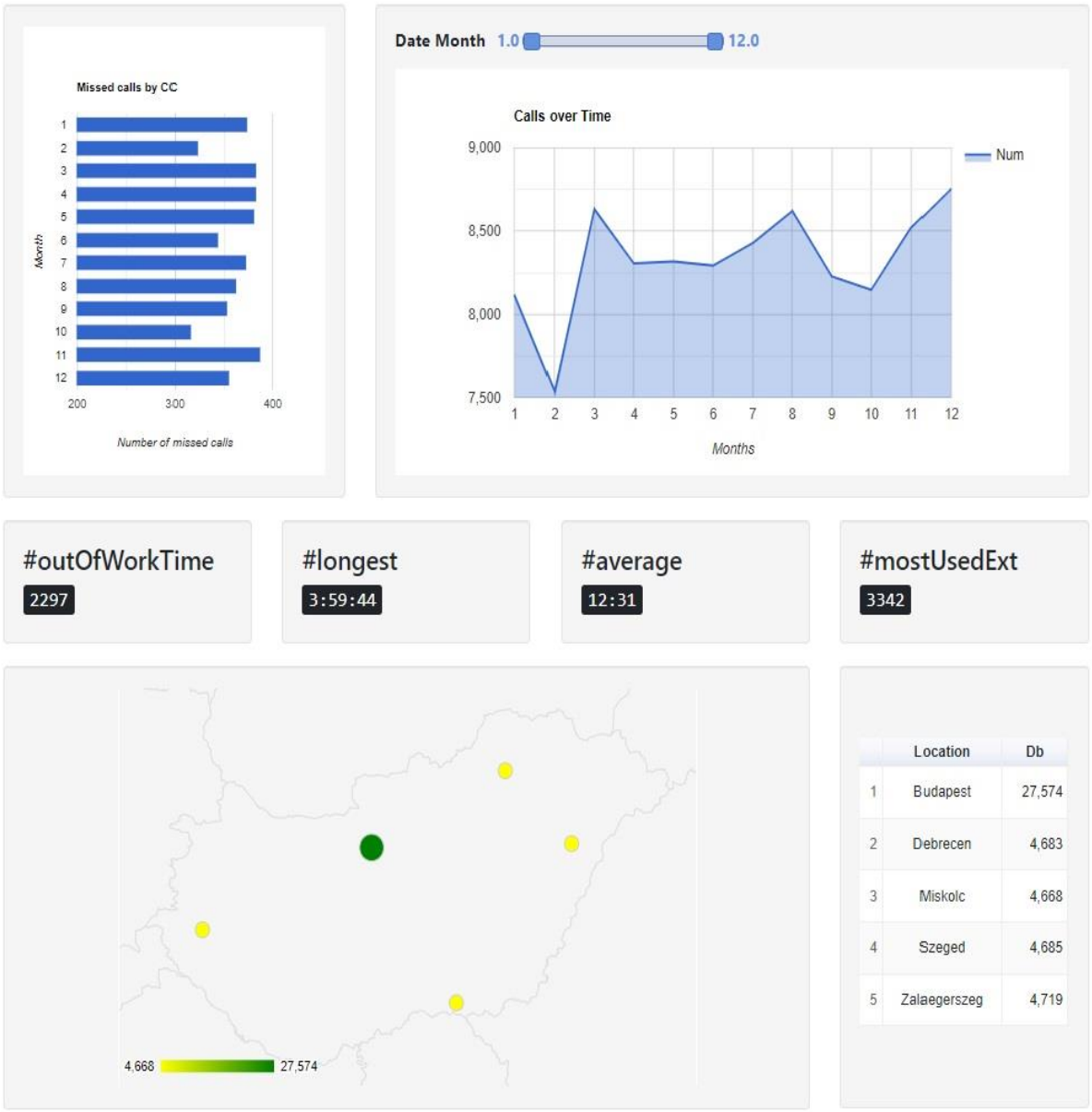
A kijelzett adatok, grafikonok a jövőben természetesen igény szerint alakíthatóak, formálhatóak, cserélhetőek. A most jelenlévőket a funkciók és lehetőségek bemutatására hoztam létre. Itt szeretném megemlíteni azt is, hogy a feltűnően egységes, már-már valótlannak tűnő kimutatások csupán az adatok véletlenszerű generálásának eredménye, valós környezetből vett adatokkal ez nem fordulna elő. Bár igyekeztem a generáló alkalmazás megírásakor súlyozásokat alkalmazni, kellően nagy, több százezres rekordszámnál a véletlenszerű adatok eloszlása túlságosan egyhangú.

A webalkalmazás második modulja egy olyan felület, amelyen a beérkező nyers napló fájl van kilistázva, mint az a 6.7. ábrán látható. Ezt a „LogView & Export” gombra kattintva érhetjük el, ahol az adatokat táblázatos formában találjuk. Ennek az a jelentősége, hogy esetleges vezetői kérésekre gyorsan reagálva adatot tudjunk szolgáltatni, illetve hibakeresés közben is rendkívül hasznos lehet. Alapértelmezett módban beérkezésük szerint vannak rendezve az adatok, tehát a legutóbb beérkezett, legfrissebb rekord lesz mindig felül, azonban tetszés szerint rendezhető bármelyik oszlop alapján. Ezen kívül egy szöveges beviteli mezőn keresztül szűréseket is alkalmazhatunk. A táblázat oldalakra bontott, lapozható, ami biztosítja az átláthatóságot

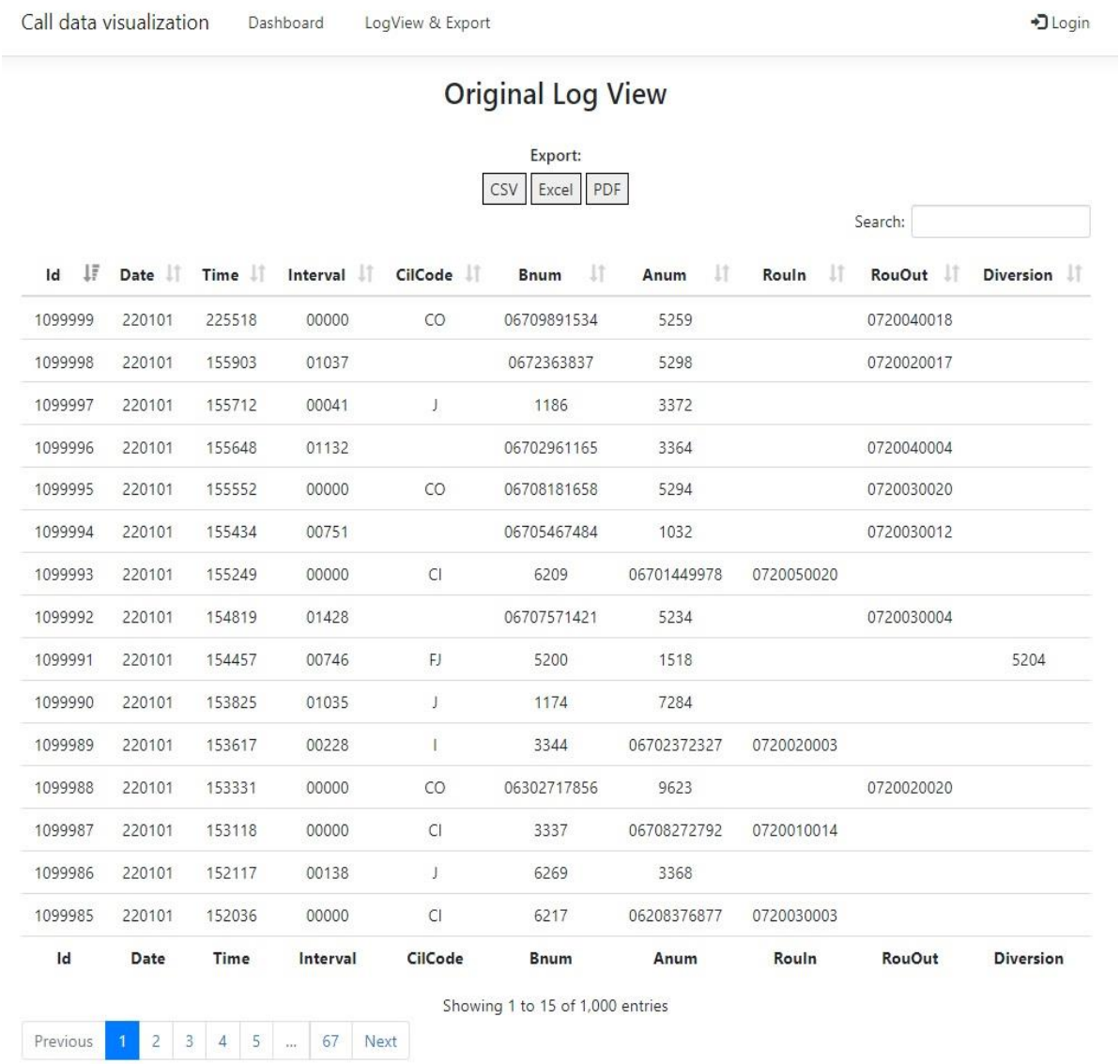
és meggátolja, hogy egyszerre túl nagy mennyiségű adatot kelljen betölteni. A táblázat felett található gombok használatával exportálhatjuk is a táblázat tartalmát a kívánt formátumokban. Exportáláskor mindig figyelembe veszi az éppen érvényben lévő rendezéseket és szűréseket, majd ezeknek megfelelően állítja össze a kimeneti fájlt. A táblázatos forma megjelenítésében a DataTables [26] nevű jQuery alapú bővítmény volt segítségemre.



6.5. Ábra: Elkészült dashboard 1 [Saját kép]



6.6. Ábra: Elkészült dashboard 2 [Saját kép]



6.7. Ábra: Napló nézet és Exportálás [Saját kép]

6.4 Tesztelés

Bár konkrét tesztelési technikákat - mint például a „Unit tesztek” - nem alkalmaztam, mert jelen dolgozatnak az nem része, de minden folyamatot, tevékenységet, metódust több alkalommal, több különböző teszt fájl alkalmazásával futtattam a fejlesztés során. Ezek eredményeit folyamatosan ellenőriztem és elemeztem annak érdekében, hogy a lehető legtöbb hibát felfedezzem és kijavíthassam. Ezen kívül a rendszer így, ebben a formájában nem fog éles környezetbe kerülni, inkább egy demó verziónak szánom.

7. TOVÁBBEJLESZTÉSI LEHETŐSÉGEK

Mint azt már megszokhattuk, az informatikai rendszerek rendszeres karbantartást és frissítést igényelnek, amibe beletartozik az új funkciókkal és lehetőségekkel való bővítés is. Ez természetesen erre a projektre is igaz, így ennek megfelelően szeretnék néhány fejlesztési lehetőséget megemlíteni, melyek kivitelezésére a dolgozat keretein belül nem volt lehetőségem, de a program előnyére válhatnának a jövőben.

- Először is fontosnak tartanám egy beléptetőrendszer bevezetését, különböző felhasználói körök és jogosultságok létrehozásával együtt, ezzel kaput nyitva több, egymástól független központ együttes kezelésének. Jelen állapotában csak egy adott cég belső környezetében való működésre alkalmas.

- Nem szokatlan jelenség, hogy több különböző típusú alközpont is megtalálható egy cégnél. Például elég sok helyen használnak Panasonic, Bosch, Avaya, Cisco vagy akár Siemens gyártmányokat is. Ezek mind eltérő működéssel, ki- és bementi fájl struktúrákkal rendelkeznek. A dolgozat elkészítése során kizárólag Ericssonra fókuszáltam, de nagy előrelépés lenne, ha többféle központ naplózását is képes lenne kezelni és feldolgozni a program.

- Bár már korábban említettem, mindenképp a továbbfejlesztések közé sorolnám azt is, hogy össze lehessen kapcsolni az adott cég egyéb rendszereivel vagy alkalmazásaival forrásadatok kinyerése céljából. Ilyen lehetne például egy Exchange szerver vagy Active Directory domain, melyekből kinyerhetőek többek között mellékszámok, szervezeti felépítések és költséghelyek is. Amennyiben sikerülne egyéb forrásokat is a rendszerhez csatolni, úgy az újonnan bekerült adatoknak megfelelően egyéb lekérdezésekkel és kimutatásokkal lehetne bővíteni a dashboard felületét is.

- Kiegészíthető lenne még egy hibafigyelő modullal is, ami képes lenne jelezni az alközpont által indikált rendellenességeket, ezáltal a döntéstámogató mivolta mellé kapna egy kis „felügyeleti rendszer” jelleget az alkalmazás.

- Végül, de nem utolsó sorban sokat hozzátehetne a felhasználói élményhez az is, hogy ha a dashboard-on szereplő vizualizációk valós időben tudnának automatikusan frissülni, azaz reagálni bármilyen adatbázis változásra, és nem kellene ezt manuálisan megtenni.

8. ÖSSZEFOGLALÓ

Jelen dolgozatomban egy telefonos alközpontokkal foglalkozó cég, az MD Vonal Kft. számára kívántam a megváltozott igényeiknek megfelelő alternatívát nyújtani. Egy telefonos hívásadatokat feldolgozó rendszert készítettem a jelenleg használt Taxawin kiváltása céljából. Úgy tapasztaltam, hogy az alközpontok szerepe és működése nem egy széles körben ismert, általános tudás, még azoknál sem, akik nap mint nap használják egy cégnél vagy egy szervezetnél, ezért a dolgozat elején betekintést engedtem a témába az olvasó részére. Ezután bemutattam a pillanatnyilag használt programot, valamint két másik lehetséges opciót. Megvizsgáltam azok előnyeit, hátrányait és felhasználási területeiket, majd arra a konklúzióra jutottam, hogy készítek egy saját alkalmazást a szükségletekhez igazítva. A központból érkező napló fájl, azaz a forrásadatok bemutatásával folytattam. Úgy gondolom, elengedhetetlen volt áttekinteni a rekordok összetételét a további döntések meghozása szempontjából, illetve ismertetni a feladat valódi kihívását, ami az adatok automatizált és helyes kezelése. Ezek után az alkalmazás tervezése következett, amihez először összegyűjtöttem, hogy melyek a pontos üzleti igények, majd az adat áramlási útját követve megterveztem a program szerkezeti felépítését. Ennek keretében bemutattam a forrás tábláimat, számba vettem adattárolási lehetőségeket, és ismertettem néhány modern adattárház megoldást. Kiemeltem még az ETL folyamatok fontosságát is, amely felelős az adatmanipulációért és az adattárház konzisztenciájának biztosításáért. Ezt követte a program másik nagy komponensének, az elemző és megjelenítő alkalmazásnak a tervezése, mely folyamán ismét áttekintettem a megvalósítási lehetőségeket, és formát öltöttek a felhasználó számára látható nézetek is. Az előkészítést követően a megvalósítás bemutatására került a sor, melyben konkrét lépéseket, alkalmazott technikákat és a projekt közben felszínre jövő kisebb problémák megoldásait ismertettem. Mindezek után prezentáltam az eredményeket, majd a munka során felmerült továbbfejlesztési lehetőségeket is kifejtettem.

9. SUMMARY

In this thesis, I tried to provide an alternative to the changed needs of MD Vonal Kft., a company dealing with telephone exchanges. I created a telephone call data processing system to replace Taxawin which is currently in use. As I noticed, the role and operation of PBXs is not a widely known, general knowledge, even among those who use it on a daily basis in a company or at an organization, so I introduced the topic at the very beginning. Then I presented the program which they currently using, along with two other possible options. I examined their advantages, disadvantages and areas of use, and then came to the conclusion that I am going to develop a new application customized by the requirements. I continued with the presentation of the log file from the PBX, i.e., the source data. I think it was essential to review the composition of the records for further decisions and to explain the real challenge of this task, which is the automated and correct handling of the data. After that I started to design the application, for which I first gathered the exact business needs, and then designed the structure of the program following the path of the data flow. As part of this, I presented my source tables, took stock of data storage options, and reviewed some modern data warehousing solutions. I even highlighted the importance of ETL processes, which are responsible for data manipulation and ensuring data warehouse consistency. This was followed by the design of the other large component of the program, the analysis and visualization application. During this phase I reviewed the implementation options and came up with a concept art regarding to the frontend page which the user communicates with. After the preparation, the implementation was presented, in which I described the techniques used for a specific purpose and the solutions to the minor problems that emerged during the project. After all of this I presented the results and listed the possibilities for further developments.

10. IRODALOMJEGYZÉK

[1] MD Vonal Kft.
(https://mdvonal.hu/magunkrol.php), Utoljára megtekintve: 2022. 05. 11.

[2] DIRCOM Elektronika - Telefonközpont
(http://dircom.hu/?page=a-kenyelem-es-a-hatekony-munka-felelose-az-irodai-telefonalasban-a-telefon-alkozpont-), Utoljára megtekintve: 2022. 05. 11.

[3] Dircom- Tarifon
(http://dircom.hu/?page=pc-tarifon), Utoljára megtekintve: 2022. 05. 11.

[4] What is PBX phone system?
(https://www.3cx.com/pbx/pbx-phone-system/), Utoljára megtekintve: 2022. 05. 11.

[5] Assono – BusinessPhone Call Center - CTI
(https://www.assono.hu/termekek/termekcsoportok-szerint/kommunikacios-rendszerek-2/aastra-businessphone/businessphone-call-center/businessphone-call-center.html),
Utoljára megtekintve: 2022. 05. 11.

[6] Opennet - Softphone
(https://www.opennet.hu/extrak/softphone), Utoljára megtekintve: 2022. 05. 11.

[7] Profitel Kft.
(https://www.profitel.hu/?lap=ceginfo), Utoljára megtekintve: 2022. 05. 11.

[8] Taxawin 1.3 dokumentáció
(https://www.profitel.hu/dnload/taxabook13.pdf), Utoljára megtekintve: 2022. 05. 11.

[9] Taxawin telefon hívásadat feldolgozó és elemző rendszer
(http://www.taxawin.hu/contents/show/taxawin), Utoljára megtekintve: 2022. 05. 11.

[10] Codebase - Business Calls rendszer
(https://www.codebase.hu/business-calls-rendszer.aspx),
Utoljára megtekintve: 2022. 05. 11.

[11] Dircom - WinPCTari telefondíj-számláló
(https://dircom.hu/?page=winpctari), Utoljára megtekintve: 2022. 05. 11.

[12] ASP.NET keretrendszer
(http://nyelvek.inf.elte.hu/leirasok/ASP.NET/index.php?chapter=1),
Utoljára megtekintve: 2022. 05. 11.

[13] Kourosh - MVC Design Pattern
(https://www.duplicatetransaction.com/mvc-design-pattern-with-a-php-example/),
Utoljára megtekintve: 2022. 05. 11.

[14] Data Mart vs. Data Warehouse
(https://panoply.io/data-warehouse-guide/data-mart-vs-data-warehouse/),
Utoljára megtekintve: 2022. 05. 11.

[15] Döntéstámogató rendszerek általános jellemzése
(https://szaszak.krudy-szeged.hu/static/kruszam/szakmn/14ker/IR_DontestamogatoRendszerek.pdf),
Utoljára megtekintve: 2022. 05. 11.

[16] Sidló Csaba, ELTE – Összefoglaló az adattáházak témaköréről
(http://scs.web.elte.hu/Work/DW/adattarhazak.htm#3),
Utoljára megtekintve: 2022. 05. 11.

[17] What is ETL?
(https://www.informatica.com/se/resources/articles/what-is-etl.html),
Utoljára megtekintve: 2022. 05. 11.

[18] David Taylor – Dimensional Modeling
(https://www.guru99.com/dimensional-model-data-warehouse.html),
Utoljára megtekintve: 2022. 05. 11.

[19] What is data visualization?
(https://powerbi.microsoft.com/hu-hu/data-visualization/),
Utoljára megtekintve: 2022. 05. 11.

[20] Dashboard definíciója
(https://www.hrportal.hu/jelentese/dashboard.html), Utoljára megtekintve: 2022. 05. 11.

[21] Mi az a Bootstrap?
(https://digikiad.gitbook.io/digitalis-kiadvanyok/bootstrap/mi-az-a-bootstrap),
Utoljára megtekintve: 2022. 05. 11.

[22] Óbudai Egyetem - Korszerű adatbázisok jegyzet 2020

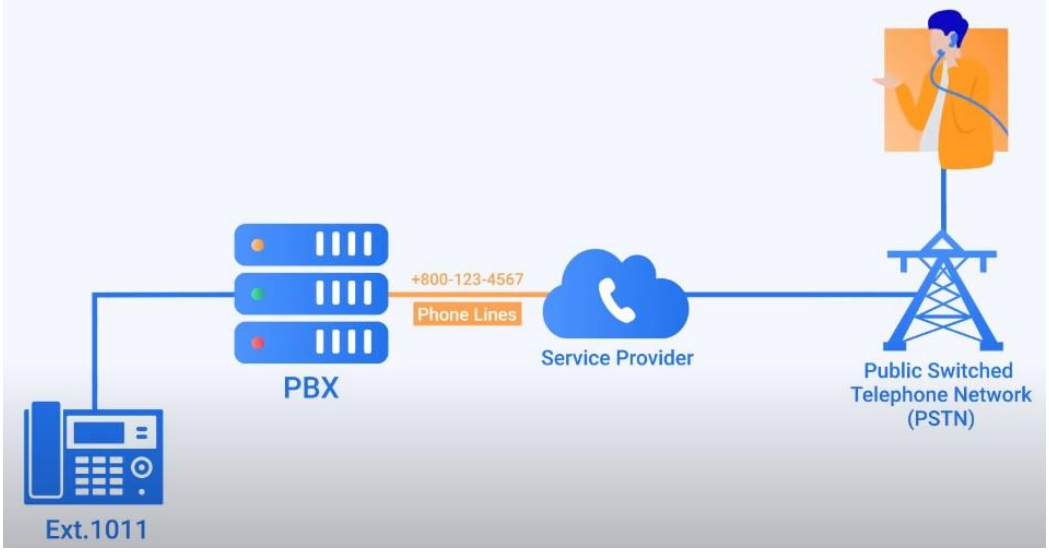
[23] Rusznák Attila: Adattárház architektúra és adatmodellezés jegyzet 2021

[24] Rusznák Attila: Adattárházak és Üzleti intelligencia jegyzet 2021

[25] Rusznák Attila: Riportkészítés jegyzet 2021

[26] SpryMedia Ltd – DataTables plug-in
(https://datatables.net), Utoljára megtekintve: 2022. 05. 11.

11. MELLÉKLETEK



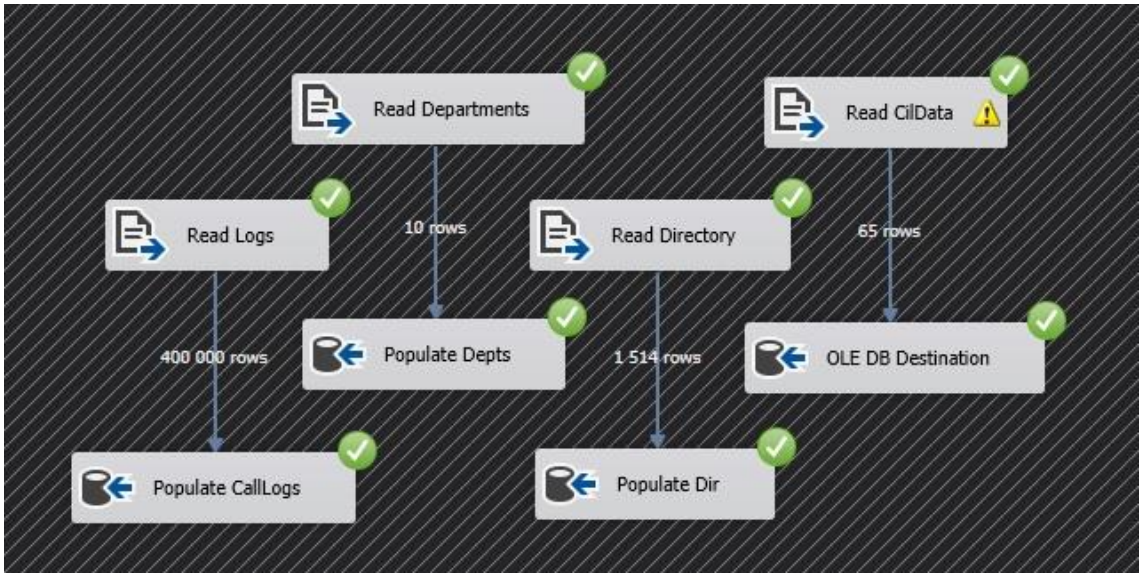
11.1. Ábra: A telefonközpont egyszerűsített működési elve
Forrás: https://www.youtube.com/watch?v=KviuXiNr_7w

Egy karakteres kódok	
( )	Outgoing call
(A)	Call handled by a PABX operator
(B)	Calls to a busy party
(C)	Abandoned calls
(D)	Extremely long call duration
(E)	Group call pickup calls
(F)	Group Hunting calls
(G)	Call has been connected with alternative route selection
(H)	Recall to route
(I)	Incoming call or tandem call
(J)	Internal call
(K)	Calls to vacant numbers
(L)	Conference call
(M)	Least cost routing
(N)	Dialled party not equal to answering party
(P)	Short Message Service (SMS) Call
(Q)	Malicious Call Tracing
(R)	Intrusion
(S)	Dynamic Route Allocation
(T)	Transferred call

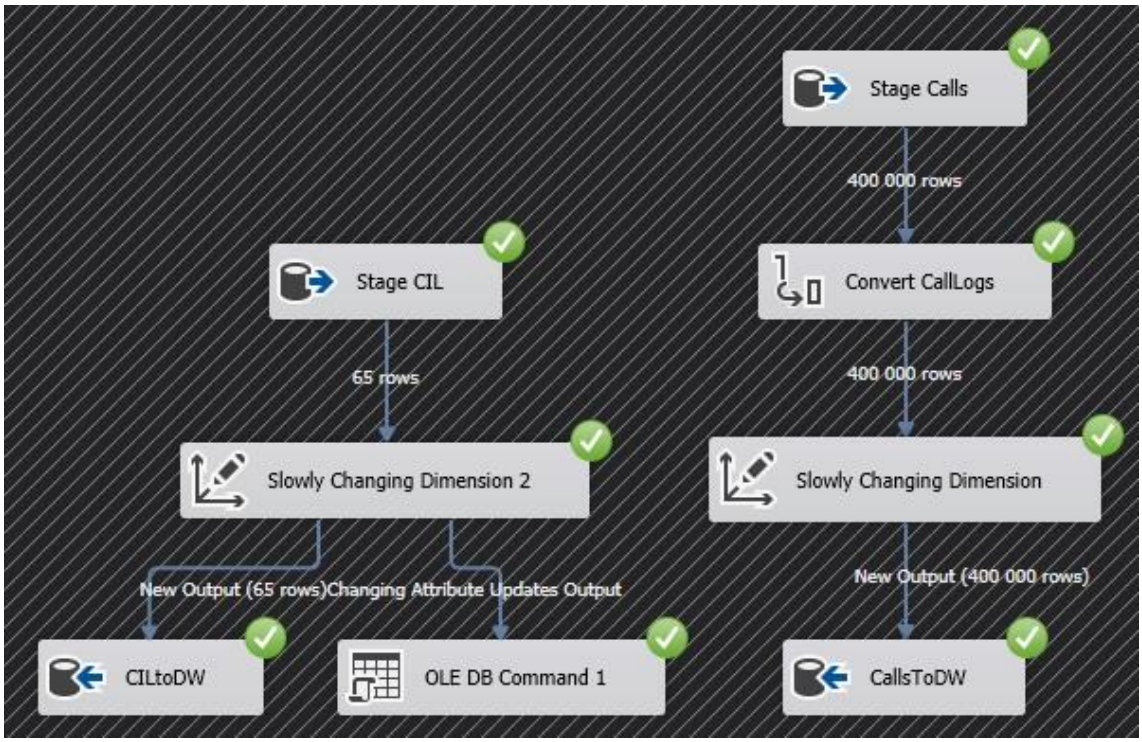
(V)	Data call
(W)	Call terminated due to route optimization
(X)	External follow me
(Y)	Call established due to route optimization
(Z)	DISA call
Két karakteres kódok	
(VJ)	Internal data call
(VI)	Incoming data call
(VO)	Outgoing data call
(DO)	Long outgoing call /Direct Divert /Diversion on Busy /Diversion on No Answer outgoing call
(DA)	Long PABX operator extended call /Direct Diverted /Diversion on Busy /Diversion on No Reply calls Extended by operator
(DI)	Long incoming call /Direct Diverted /Diversion on Busy /Diversion on No Reply incoming calls
(DJ)	Long internal call /Direct Diverted /Diversion on Busy /Diversion on No Answer internal calls
(DV)	Long data call
(DX)	Long external follow me call
(NJ)	Dialled party is not the answering party on an internal call
(NI)	Incoming DID call when the answering party was not the dialled party
(NT)	Dialled party is not the answering party on a transferred call
(CI)	Abandoned non-DISA call for Segment 0
(CJ)	Abandoned internal call for Segment 0
(CO)	Abandoned outgoing call in a private network for Segment 0
(CZ)	Abandoned DISA call for Segment 0
(FC)	Abandoned Group Hunting outgoing /Incoming /Internal calls for Segment 3 and Segment 7
(DC)	Abandoned outgoing /Incoming /Internal call due to Direct Diversion /Diversion on Busy /Diversion on No Reply in a private network
Három karakteres kódok	
(NCI)	Abandoned non-DISA call, dialled party is not the answering party.
(NCJ)	Abandoned internal call, dialled party is not the answering party.
(NCO)	Abandoned outgoing private network call, dialled party is not the answering party.
(NCZ)	Abandoned DISA call, dialled party is not the answering party.
(FCI)	Abandoned non-DISA incoming Group Hunting Call.
(FCJ)	Abandoned internal Group Hunting Call.
(FCZ)	Abandoned DISA call.
(FCO)	Abandoned outgoing Group Hunting Call.

(DCI)	Abandoned incoming Direct Diversion Call.
(DCJ)	Abandoned internal Direct Diversion Call.
(DCZ)	Abandoned Direct Divert DISA Call.
(DCO)	Abandoned outgoing Direct Divert Call.
(DRI)	Incoming call Diverted on No Answer.
(DRJ)	Internal call Diverted on No Answer.
(DRO)	Outgoing call Diverted on No Answer.
(DRL)	Conference call Diverted on No Answer.
(DRG)	Call has been connected with alternative route optimization on Diverted on No Answer.
(DRH)	Recall to route Diverted on No Answer.
(DRV)	Data call Diverted on No Answer.
(DRT)	Transfered call Diverted on No Answer.
(DAT)	Operator Extended call after Direct Diversion to the operator.
(DRA)	Operator Extended call after Diverted on No Answer to operator.

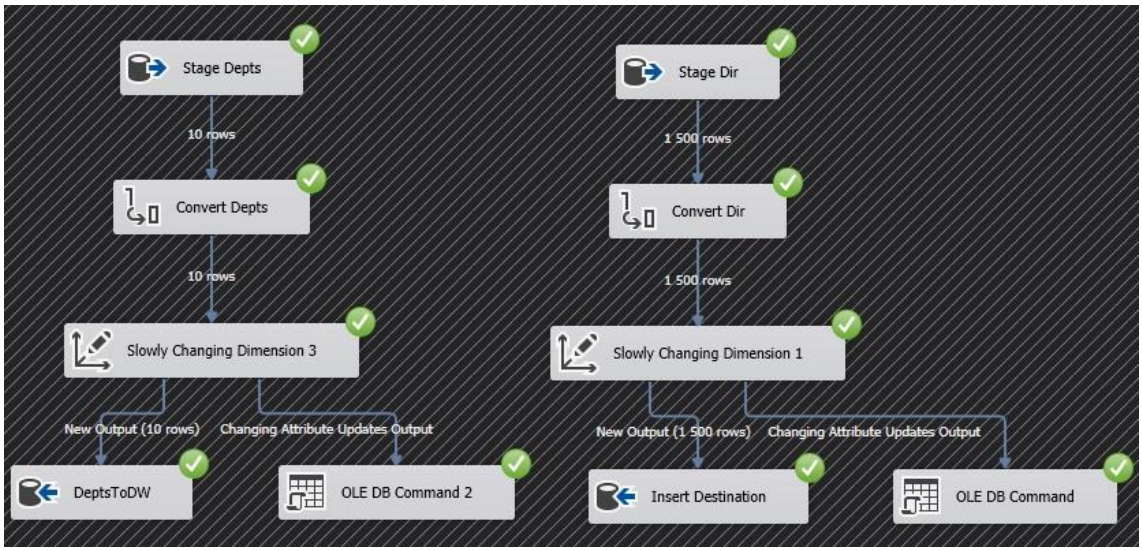
11.2. Táblázat: CIL kódok táblázatos formában
Forrás: ALEX (Active Library Exchange) dokumentáció (MD110 - BC9)



11.3. Ábra: Adatok előkészítése [Saját kép]



11.4. Ábra: Adatok átalakítása 1 [Saját kép]



11.5. Ábra: Adatok átalakítása 2 [Saját kép] [1]