SZAKDOLGOZAT

Szakács Kristóf

2024

Magyar Agrár- és Élettudományi Egyetem

Szent István Campus

Műszaki Intézet

Adattechnológus-adatelemző szakmérnök szakirányú továbbképzési szak

HR Adattárház Fejlesztése Chatbot és Bérszámfejtő Rendszer Integrációjával

Belső konzulens:	Orova Lászlóné dr. egyetemi docens
Belső konzulens intézete/tanszéke:	Műszaki Intézet, Mérnökinformatikai Tanszék
Külső konzulens:	Gál Andor Senior tanácsadó
Készítette:	**Szakács Kristóf (GLYDC0)**

Gödöllő

2024

Kivonat

A WHC Kft. egy kelet-közép-európai régióban működő jelentős HR szolgáltató, amely a munkaerő-kölcsönzés, közvetítés és bérszámfejtés területén nyújt széleskörű szolgáltatásokat. A vállalat a kommunikációs és kapcsolattartási folyamatok egyszerűsítésére egy többnyelvű Chatbot rendszert alkalmaz, amelyet külső szolgáltató fejleszt és üzemeltet. Azonban a rendszer használatával kapcsolatos részletes statisztikák és a historikus adattárolás képességének hiánya komoly kihívást jelent a vállalat számára.

Az adattárházak fontosságát az adatvezérelt döntéshozatal támogatásában kiemelve, a dolgozat bemutatja, hogy az adattárházak hogyan képesek nagy mennyiségű adat hatékony kezelésére, a vállalatok gyors reagálására a változó piaci környezetre, valamint a jogi és szabályozási követelmények teljesítésére. Az adattárházak segítségével a vállalatok jobban megérthetik ügyfeleiket, hatékonyabban tudnak versenyezni a piacon, és javíthatják az üzleti folyamataikat.

A szakdolgozat a WHC Kft. szolgáltatásainak bemutatása mellett részletesen ismerteti az adattárház projektet, beleértve a célkitűzéseket, a forrásrendszereket és az adattárház felépítését.

A szakdolgozat célja egy olyan adattárház létrehozása volt, amely integrálja és centralizálja a WHC Kft. Chatbot rendszeréből és Abacus bérszámfejtő rendszeréből származó adatokat részletes statisztikák elérése, a historikus adatok megőrzése és az integrált adatelemzés lehetőségének megteremtése érdekében.

A projekt során kifejlesztett adattárház a Python nyelven írt ETL (Extract, Transform, Load) eljárások segítségével teszi lehetővé az adatintegrációt. Az adatok két különböző forrásból történő összegyűjtését és egységesítését követően, a Dagster adatfeldolgozási keretrendszer segítségével az adatok ütemezetten kerültek betöltésre egy SQL Server adatbáziskezelő rendszerbe. Az így létrejött adattárház lehetőséget nyújt részletes statisztikák készítésére, a historikus adatok hosszú távú megőrzésére és az integrált adatelemzésre. A Power BI alkalmazásával készített dashboard szemléletesen támogatják a döntéshozókat, így elősegíti a vállalati hatékonyság növelését.

Tartalomjegyzék

1. Bevezetés

1.1. Problémafelvetés és célkitűzés

1.1.1. Problémafelvetés

A WHC Kft. egy jelentős HR szolgáltatásokat nyújtó cég, melynek fő tevékenységi körei közé tartozik a munkaerő kölcsönzés, közvetítés és bérszámfejtés. A vállalat a kölcsönzött munkaerővel való kommunikáció és kapcsolattartás megkönnyítése érdekében használ egy többnyelvű Chatbot rendszert, amit egy külső szolgáltató fejleszt és üzemeltet. Jelenleg azonban a vállalat vezetése számára nehézséget okoz az, hogy nehezen vagy egyáltalán nem áll rendelkezésre kielégítő és részletes statisztikai adat a Chatbot rendszer használatával kapcsolatban. Kritikus információk, mint például a rendszerbe bejelentkezett és nem bejelentkezett felhasználók aránya, nyelvi preferenciák, és egyéb releváns adatok, nem érhetőek el. További komplikációt jelent, hogy a Chatbot rendszer nem rendelkezik historikus adattárolási képességgel, vagyis, ha egy felhasználó jogviszonya megszűnik, az illető adatai fizikailag törlődnek a rendszerből, ezzel megakadályozva a hosszú távú adatelemzést és trendek azonosítását.

1.1.2. Célkitűzés

A célom egy olyan adattárház létrehozása, amely integrálja és centralizálja a Chatbot rendszerből származó adatokat és az Abacus bérszámfejtő rendszerben megtalálható kölcsönzötti állományra vonatkozó további részletes adatokat. Az adattárház kialakítása lehetővé tenné a WHC Kft. számára, hogy részletes összehasonlító elemzéseket végezzen a munkavállalókkal kapcsolatos preferenciákról, a rendszer használat hatékonyságáról és egyéb releváns adatokból. A projekt elsődleges célja a következők biztosítása:

- **Részletes statisztikák elérése**: Az adattárház révén a felső és középvezetés képes legyen elemezni a Chatbot rendszer használatával kapcsolatos statisztikai adatokat, mint a felhasználók nyelvi megoszlása, bejelentkezési szokásaik és egyéb releváns tevékenységeik.

- **Historikus adatok megőrzése**: Az adattárház biztosítson lehetőséget a Chatbot rendszerből származó adatok historikus tárolására is, még azok esetében is, akik már nem állnak jogviszonyban a céggel. Ezáltal hosszú távú trendek és mintázatok azonosíthatók lesznek.
- **Integrált adatelemzés**: Az adattárház létrehozásával nyíljon lehetőség a Chatbot rendszer adatainak kiegészítésére az Abacus bérszámfejtő rendszer adataival, hogy teljesebb képet kapjunk a munkavállalók profiljáról, magatartásáról és igényeiről.

A projekt sikerességének kulcsa olyan belső adatfeldolgozási és elemzési eljárások kifejlesztése, amelyek hozzájárulnak a vállalati hatékonyság növeléséhez.

1.2. Cég bemutatása: WHC Kft.

A WHC Kft. a kelet-közép-európai régió meghatározó HR szolgáltatójaként tevékenykedik. Széleskörű szolgáltatásokat nyújtunk, amelyek közé tartozik a fizikai és szellemi munkaerő kölcsönzése és közvetítése, valamint a szövetkezeti foglalkoztatás (beleértve a diákokat, nyugdíjasokat és kismamákat), bérszámfejtés és nemzetközi toborzás.

Nemzetközi jelenlétünk kiterjed Magyarországon túl Ausztriára, Szerbiára, Szlovákiára, Ukrajnára, Bosznia-Hercegovinára, Észak-Macedóniára, Horvátországra, Montenegróra és Szlovéniára is. Professzionális segítséget nyújtunk partnereinknek és munkavállalóinknak ezen országokban is.

Fontos kiemelni, hogy minősített foglalkoztatóként akár harmadik országbeli munkavállalók szerződtetésével is ki tudjuk elégíteni partnereink igényeit.

Fő toborzási területeink közé tartoznak a Fülöp-szigetek, Mongólia és Indonézia, de az adódó igényeknek megfelelően további országok bevonása is lehetséges. A WHC Kft. elkötelezett a hatékony és megbízható munkaerőpiaci megoldások nyújtása iránt, és folyamatosan fejleszti szolgáltatásait a partnerek és munkavállalók elégedettségének érdekében.

1. ábra
WHC főbb adatok

Partnereik, munkavállalóik és az álláskeresők elégedettsége a szakmai munkájának alapja. Gyors, pontos és testre szabott szolgáltatást nyújt, legyen szó rövid vagy hosszú távú munkaerőigényről, vagy bérszámfejtésről. (1. ábra)

WHC Kft. által biztosított megoldások:

- Fizikai munkaerő-kölcsönzés és -közvetítés
- Szellemi munkaerő-kölcsönzés és -közvetítés
- Diákmunka
- Nyugdíjas-szövetkezet
- Kismama-szövetkezet
- Bérszámfejtés és auditálás

(WHC, 2024)

2. Irodalom kutatás

2.1. Forrás rendszerek ismertetése

A HR adattárházam fejlesztése során két különböző forrásrendszerből gyűjtöm az adatokat. Az első forrásrendszer az Abacus Bér, ami a piacon jól ismert bérszámfejtő rendszer. Ez a rendszer szolgálja az alapját annak, hogy a munkavállalókkal kapcsolatos összes személyes és foglalkoztatási jogviszonyra, valamint bérre vonatkozó adatot központilag kezeljük. Az Abacus Bér rendszer a WHC Kft. esetében különös jelentőséggel bír, mivel itt a kölcsönzött munkaerő állományához köthető adatokat tekintjük az adatok fő forrásának. Ez a megközelítés lehetővé teszi számunkra, hogy a bérszámfejtési folyamatokat hatékonyan és átláthatóan kezeljük, miközben biztosítjuk az adatok integritását és megbízhatóságát a HR adattárház építése során.

A HR adattárházam további forrás rendszere a Chatbot Team által fejlesztett és üzemeltetett chatbot rendszer, amely a WHC Kft. kölcsönzött munkaerő állományának kommunikációját és ügyintézési folyamatait hivatott segíteni. Ez a rendszer kiemelkedően fontos szerepet tölt be abban, hogy a munkavállalók gyorsan és hatékonyan kaphassanak választ kérdéseikre, valamint, hogy zökkenőmentesen intézhessék az őket érintő ügyeket. Az említett chatbot rendszer öt különböző nyelven érhető el: magyar, angol, szerb, ukrán, orosz és indonéz. Ez a nyelvi sokszínűség különösen előnyös a külföldi munkavállalók számára, mivel így saját anyanyelvükön is képesek kommunikálni, ami jelentősen megkönnyíti számukra a vállalatnál való tájékozódást és ügyintézést.

2.1.1. Abacus Bér

Az VT-Soft által fejlesztett Abacus Bér egy korszerű és rugalmas bérszámfejtő rendszer, amely széles körű funkciókkal rendelkezik a hatékony bérügyvitelhez. A rendszer előnyei között szerepel a gyors és pontos bérszámfejtés, az aktuális jogszabályoknak való teljes mértékű megfelelés, valamint az integrációs képesség más ügyviteli rendszerekkel. Az Abacus Bér lehetővé teszi a felhasználók számára, hogy rugalmasan kezeljék a bérszámfejtési folyamatokat, beleértve a munkavállalói adatok kezelését, a bérjegyzékek összeállítását, valamint az adatszolgáltatásokat és riportokat. A rendszer az adatokat historikusan kezeli, így

a korábbi időszakok adatszámfejtései is könnyen elérhetőek és módosíthatóak. Emellett, a bérszámfejtés mellett, a VT-Soft számos más HR és munkaügyi modullal is rendelkezik, mint például TB nyilvántartás, munkaidő-nyilvántartás, és cafeteria rendszer, melyek integráltan használhatóak az Abacus Bérrel.

Az Abacus Bér jellemzői közé tartozik a felhasználóbarát felület, amely támogatja a folyamatos használatot és hozzáférést az összes funkcióhoz egyetlen modulból. A rendszer támogatja a tömeges adatimportálást és -exportálást, lehetővé teszi rugalmas riportolást és elemzést, valamint automatikus korrekciókat biztosít a bérszámfejtési folyamat során felmerülő eltérések kezelésére

Az Abacus Bér egy moduláris felépítésű, egyszerűen kezelhető és korszerű bérszámfejtő program, amit a VT-SOFT több évtizedes szakmai tapasztalattal rendelkező magyar

Néhány fontos tulajdonsága:

- Teljes mértékben alkalmazkodik az aktuális jogszabályokhoz.
- Könnyen integrálható kapcsolható beléptető-rendszerekhez, bér- illetve munkaügyi és vállalatirányítási programokhoz.
- Moduláris felépítésű, így többi termékükkel, például a TB, HR, Cafeteria és Munkaidő szoftverekkel is együttműködik.
- Rugalmas és felhasználóbarát, valamint bármely ügyviteli rendszerhez könnyedén kapcsolható.

Az Abacus Bér rendszerét a munkaerő kölcsönzés szempontjából kiinduló és elsődleges adatforrásnak tekintjük. Ennek oka, hogy a rendszerben megtalálható minden kölcsönzöttre és saját állományhoz tartozó személyes, foglalkoztatási és bérszámfejtési információ. Ezáltal az Abacus Bérprogram lehetővé teszi a teljes körű és pontos bérszámfejtést, valamint a munkaerő kölcsönzéshez kapcsolódó adatok átlátható kezelését. (VT-Soft, 2024)

2.1.2. Chatbot

A Chatboss Team egy HR chatbot vállalat, amely a toborzást, az HR adminisztrációt és a belső kommunikációt támogatja chatbotokkal. Az általuk kifejlesztett mesterséges intelligencia alapú technológia lehetővé teszi az automatizált beszélgetéseket és hatékony kommunikációs platformot biztosít a jövő számára. Ezenkívül a Chatboss Team célja a teljes belső HR munkafolyamat optimalizálása, a toborzó csapat hatékonyságának növelése és a vállalaton

belüli kommunikációs folyamatok javítása. A konklúzió részben pedig javaslatokat tesznek a fejlődésre, hogy milyen problémákat lehet javítani vagy elkerülni a kutatásukkal.

Megoldások:

- Toborzás támogató Chatbot
- HR Adminisztrációs Chatbot
- Ügyfélszolgálati Chatbot

Ezek a chatbotok nem csak az ügyfelek, hanem a vállalatok számára is előnyöket kínálnak, lehetővé téve számukra, hogy növeljék a felhasználói elkötelezettséget, javítsák az arculattervezést, és a kulcsszavakra alapozott válaszadás, valamint az élő chat funkciók révén egy intelligens FAQ rendszert valósítsanak meg. Az ilyen típusú technológiák alkalmazása lehetővé teszi a vállalatok számára, hogy javítsák ügyfélszolgálati márkájukat és erősítsék piaci pozíciójukat azáltal, hogy hatékonyabb és költségtakarékosabb ügyfélkiszolgálást biztosítanak.

AI-alapú chatbotok használata jelentős előnyökkel jár a vállalati működés számos területén, beleértve az új munkavállalók integrációját, a munkavállalói visszajelzések kezelését, képzési anyagok biztosítását, közlekedési információk megosztását, panaszok kezelését és a belső kommunikáció támogatását. Ezek a chatbotok automatizálják az adatmegosztást és a feladatdelegálást, csökkentve az emberi hibák lehetőségét, miközben pontos és konzisztens válaszokat biztosítanak. A munkavállalói felmérésektől kezdve az ötletládán át az interaktív onboardingig és offboardingig terjedő széleskörű támogatásnak köszönhetően javítják a vállalati munkakörnyezetet, növelik az alkalmazottak elégedettségét, és hozzájárulnak a vállalati célok hatékonyabb eléréséhez.

A WHC-nél mindhárom típusú chatbotot alkalmazzuk, azonban az adattárház szempontjából a HR adminisztrációs és ügyfélszolgálati chatbot a mérvadó. Ezeket a megoldásokat kombinálva használjuk a kölcsönzött állományunkkal való ügyintézés és kommunikáció megkönnyítésére.

HR Adminisztrációs Chatbot:

A HR Adminisztrációs Chatbot bevezetése jelentős előnyöket kínál a vállalatok számára, amelyek hozzájárulnak az alkalmazottak elégedettségének, elkötelezettségének javításához, a belső kommunikáció hatékonyságának növeléséhez, a humán tevékenységek során fellépő hibák minimalizálásához, valamint a zökkenőmentes információáramlás biztosításához. Az alkalmazottak gyors és hatékony válaszokat kaphatnak adminisztratív kérdéseikre, ami

csökkenti a hibalehetőségeket és javítja a munkamorált. A chatbotok lehetővé teszik az egyszerű és gyors kommunikációt a HR-osztállyal és az alkalmazottak között, beleértve azokat is, akik nem rendelkeznek céges e-mail címmel.

Ügyfélszolgálati Chatbot:

Az Ügyfélszolgálati Chatbotok bevezetésével a vállalatok gyorsabb válaszadást, magas szintű ügyfélkiszolgálást, költséghatékonyságot és folyamatos, 24/7 elérhetőséget kínálhatnak ügyfeleik számára. Ezek a digitális asszisztensek előre meghatározott adatok és algoritmusok segítségével azonnal válaszolnak a felhasználói kérdésekre, lehetővé téve a vállalatok számára, hogy minden ügyfél számára gyors és pontos válaszokat biztosítsanak. Az egyedi és személyre szabott interakcióknak köszönhetően a chatbotok javítják az ügyfélkiszolgálási élményt, miközben csökkentik a vállalatok költségeit azáltal, hogy nem igényelnek folyamatos emberi beavatkozást. (Chatbossteam, 2024)

Összegzés:

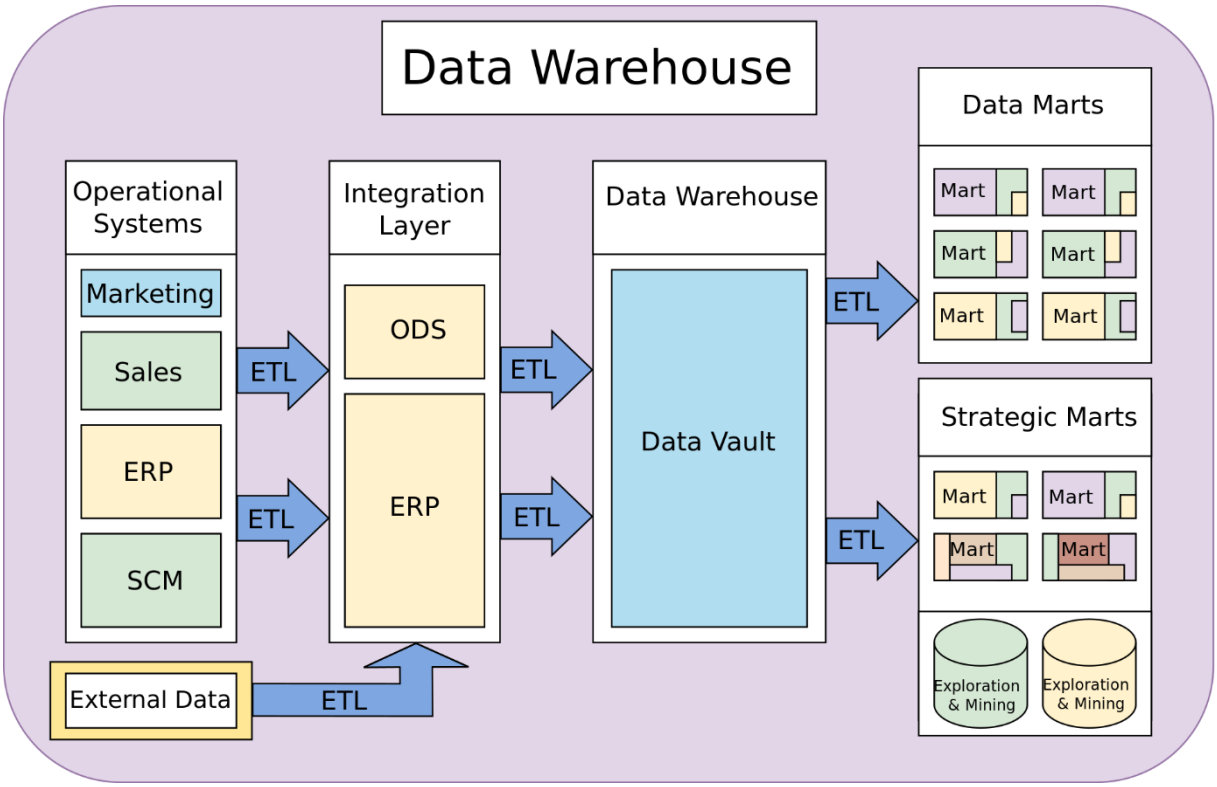
Az adatok, amelyek a chatbot rendszerből származnak a munkavállalókról, kritikus fontosságúak a vállalati kommunikációs stratégiák és a munkavállalói interakciók kezelésében. Ezek az információk magukban foglalják, hogy mely munkavállalók vannak regisztrálva a rendszerben, be vannak-e jelentkezve, milyen nyelvű chatbotokat használnak, mely feliratkozási listákhoz vannak hozzárendelve, valamint további személyes adatokat, amelyek az Abacus bérrendszerben is megtalálhatóak.

2.2. Adattárházak bemutatása

Az adattárház (data warehouse) olyan adatbázisrendszer, ami egy tárgyorientált, integrált és időben változó információkat tárol, amely a vállalatok számára stratégiai jelentőséggel bír.

Az adattárház egy nagy, integrált adatbázis, amely különböző forrásokból származó adatokat tárol. Ezek az adatok lehetnek üzleti tranzakciók, ügyféladatok, értékesítési információk, pénzügyi adatok stb. Az adattárház célja, hogy egységesítse és optimalizálja az adatok tárolását és hozzáférését. (2. ábra)

2. ábra
Adattárház felépítése
(How Data Modeling and ETL designs are important for designing a Data Warehouse?, 2024)



Miért fontos a vállalatok számára?

Döntéshozatal támogatása: Az adattárház lehetővé teszi a vezetők és döntéshozók számára, hogy pontos és időszerű információk alapján hozzanak stratégiai döntéseket. A vállalatoknak gyorsan kell reagálniuk a változó piaci környezetre, és az adattárház segít ebben.

Nagy adatmennyiség kezelése: A vállalatok napról napra rengeteg adatot generálnak. Az adattárház segít ezt a hatalmas mennyiséget rendszerezni és hatékonyan kezelni. Az adatok aggregálása és optimalizálása révén a vállalatok hatékonyabban tudnak működni.

Jogi és szabályozási követelmények: Az adattárház lehetővé teszi a vállalatok számára, hogy megfeleljenek a jogi és szabályozási előírásoknak az adatok tárolása és kezelése terén. Az adattárházban tárolt adatok biztonságosan kezelhetők, és a vállalatoknak könnyebb a szabályozásoknak megfelelniük.

Versenyelőny: Azok a vállalatok, amelyek hatékonyan használják az adattárházat, jobban megérthetik ügyfeleiket, és hatékonyabban tudnak versenyezni a piacon. Az adattárház segítségével a vállalatok jobban megismerhetik a fogyasztói szokásokat, a piaci trendeket és az

üzleti lehetőségeket. (Kimball & Ross, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2002) (Sidló, 2004) (Inmon, 2005)

2.2.1. Adattárház komponensei

Adatforrások:

Az adattárházba kerülő adatok forrásai lehetnek tranzakciós rendszerek, külső adatforrások, vagy akár más adattárházak is. Fontos megérteni, hogy az adatforrások milyen típusú adatokat szolgáltatnak és milyen formátumban. Például:

- Tranzakciós rendszerek: Ezek azok a rendszerek, amelyek az üzleti tranzakciókat kezelik, például az értékesítési rendszerek, az ügyfélkezelő rendszerek stb. Ezekből az adatokból gyakran naprakész információkat szerezhetünk.
- Külső adatforrások: Ezek az adatok olyan forrásokból származhatnak, mint például webes API-k, harmadik fél által biztosított adatok, vagy akár adatfájlok, például CSV, Excel stb. Ezek gyakran különböző formátumokban és struktúrákban érkeznek.

Adattárolás:

Az adattárolás az adattárházban már előkészített és átalakított adatok tárolását foglalja magában. A tárolás módja és helye változhat attól függően, hogy az adattárház milyen típusú adatbázisrendszerekben tárolja az adatokat. Például:

- Relációs adatbázisok: Ezek a hagyományos SQL alapú adatbázisrendszerek, amelyek táblákban tárolják az adatokat. Ezeket gyakran használják az adattárházak tárolására, mivel jól támogatják az SQL lekérdezéseket és a tranzakciókezelést.
- Oszlopos adatbázisok: Ezek az adatbázisrendszerek az adatok oszloponként történő tárolására specializálódtak, ami hatékonyabb adatkompressziót és lekérdezési teljesítményt eredményezhet nagy adatmennyiségek esetén.

Az adattárolás során fontos figyelembe venni az adatok méretét, az adatelérés gyakoriságát és az analitikai igényeket annak érdekében, hogy megfelelő adatbázisrendszert válasszunk az adattároláshoz.

Ezek a komponensek együttesen biztosítják az adattárház hatékony működését, lehetővé téve az üzleti döntéshozatalt és az adatvezérelt stratégiák kialakítását. (Inmon, 2005) (Kimball & Ross, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2002)

2.2.2. Adattárház Architektúrák

Kimondott (Monolitikus) Adattárház:

A kimondott, vagy monolitikus adattárház egy hagyományos, központosított architektúra, amely egyetlen nagy adattároló egységet alkot. Ez az adattárház típus általában a következő jellemzőkkel rendelkezik:

- Központosított adattárolás: Az összes adat egyetlen helyen van tárolva, ami egyszerűsíti az adatelérést és az adatkezelést.
- Gyakori adatelérések: Mivel az összes adat egy helyen található, könnyen elérhető és használható az analitikai és riportálási célokra.
- Egységes adatmodell: Az adattárházban található adatokat gyakran egyetlen, centralizált adatmodell szerint tárolják, ami segíti az egységes adatértelmezést és az egyszerűbb adatfeldolgozást.

Adatpiacok (Data Mart):

A Data Martok kisebb, specifikus területekre fókuszáló adattároló egységek, amelyek az adattárház részei lehetnek, vagy önállóan is működhetnek. A Data Martokra jellemzők:

- Szegmentált adatok: A Data Martok specifikus üzleti területekre vagy funkcionális egységekre fókuszálnak, így csak az adott területre vonatkozó adatokat tartalmazzák.
- Optimalizált teljesítmény: Mivel kisebb méretűek és specifikus területekre szakosodottak, a Data Martok gyakran gyorsabb adatelérést és jobb teljesítményt biztosítanak az adott üzleti területek számára.
- Könnyebb kezelés és skálázhatóság: A Data Martok könnyebben kezelhetők és skálázhatók, mint a hagyományos adattárházak, mivel kisebb adatmennyiséget és specifikus üzleti igényeket kezelnek.

Logikai Adattárház:

A logikai adattárház olyan adattárház architektúra, amely logikai rétegeket vagy nézeteket alkalmaz az adatok strukturálására és az üzleti logika elválasztására. A logikai adattárházak jellemzői:

- Absztrakció és elválasztás: A logikai rétegek lehetővé teszik az adatok absztrakcióját és az üzleti logika elválasztását az adattárolástól, ami rugalmasabb adatfeldolgozást és könnyebb karbantarthatóságot eredményez.
- Központosított adatmodellezés: A logikai adattárházak gyakran központosított adatmodellt használnak, amely egységes nézetet biztosít az üzleti adatokról az egész szervezet számára.
- Könnyebb integráció és felhasználás: A logikai adattárházak lehetővé teszik az adatok könnyebb integrációját és felhasználását különböző üzleti alkalmazásokban és analitikai eszközökben.

(Inmon, 2005) (Kimball & Ross, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2002) (Kimball, The Data Warehouse Lifecycle Toolkit, 1998)

2.2.3. Adattárház Felhasználók és Felhasználási Módok

Döntéshozók és Analitikusok:

A döntéshozók és analitikusok az adattárházakat gyakran stratégiai és operatív döntéseik támogatására használják. Néhány jellemző módszer, ahogy az adattárházakat ezen felhasználók csoportja használja:

- Stratégiai döntések támogatása: Az adattárházak az üzleti adatok gyűjtésével és elemzésével segítik a vezetőket a hosszú távú stratégiai döntések meghozatalában. Például, az értékesítési trendek, a piaci részesedés elemzése segíthet a termékstratégiák kidolgozásában.
- Adatvezérelt döntéshozatal: Az adattárházak által biztosított adatok és riportok segítenek az adatvezérelt döntéshozatalban. A vezetők és analitikusok a legfrissebb adatokat használják fel a döntések meghozatalához, és a korábbi adatok elemzésével kvalitatív és mennyiségi előrejelzéseket készítenek.

- Komplex elemzések és riportok: Az adattárházakban tárolt adatok lehetővé teszik a komplex analitikai műveleteket és riportok készítését. Az OLAP (Online Analytical Processing) kockák és az ad-hoc riportok segítségével a felhasználók részletes elemzéseket végezhetnek az üzleti adatokról.

Operatív Felhasználók:

Az operatív felhasználók az adattárházakat gyakran a mindennapi munkájuk során használják, például operatív riportok készítésére és ad-hoc lekérdezések végzésére. Néhány jellemző módszer, ahogy az adattárházakat ezek a felhasználók csoportja használja:

- Operatív riportok: Az adattárházak rendszeresen generálnak és szállítanak operatív riportokat az üzleti folyamatok nyomon követéséhez és az operatív döntések támogatásához. Ezek a riportok gyakran a naprakész adatokat és a kulcsfontosságú mutatókat tartalmazzák.
- Ad-hoc lekérdezések: Az operatív felhasználók lehetőséget kapnak arra, hogy saját ad-hoc lekérdezéseket hajtsanak végre az adattárházban tárolt adatokra, így azonnali válaszokat kapnak az aktuális üzleti kérdéseikre.
- Trendelemzések és teljesítménymérések: Az operatív felhasználók az adattárházban tárolt adatok alapján végzik el a trendelemzéseket és a teljesítménymméréseket a mindennapi munkájuk során, például az értékesítési teljesítmény vagy a termelés hatékonyságának nyomon követése céljából.

(Inmon, 2005) (Kimball & Ross, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2002)

2.3. Adattárház modellezés

Az adattárház modelljének sikeressége érdekében kritikus fontosságú, hogy milyen módszert választunk az adatmodell létrehozására. Két fő módszer létezik: az entitás-kapcsolat (ER) modell és a dimenziós modell. Ezek kombinációját javasolják, ahol a dimenziós modell alkalmazása javasolt a felhasználói felületen, az ER modell pedig a háttérben történő adatkiaknázásra és adattárolásra szolgál.

2.3.1. Dimenziók és Mérők

Dimenziók: A dimenziók az adattárházban azokat a jellemzőket írják le, amelyek mentén az adatokat elemzési célokra kategorizáljuk és szegmentáljuk. Például az idő, a földrajzi hely, vagy a termékek dimenziói. Ezek a dimenziók lehetővé teszik az adatok multidimenzionális nézetben történő szervezését, így támogatva a döntéstámogató lekérdezéseket és analíziseket.

Mérők: A mérők azok az adatelemek, amelyek kvantitatív információt tartalmaznak, és amelyeken matematikai műveleteket (pl. összegzés, átlagolás) végezhetünk. A mérők tükrözik az adattárházban tárolt teljesítményindikátorokat, mint például értékesítési összegek vagy tranzakciók száma.

Fact (tény) táblák

A fact táblák az adattárház modell központi elemei, amelyek az üzleti folyamatok kvantitatív mérőit tartalmazzák. Ezek a táblák tartalmazzák a mérőket (más néven mértékegységeket), amelyek az adattárház legkritikusabb elemzési célú adatelemei. A fact táblák kapcsolódnak a dimenziós táblákhoz, amelyek kontextust és dimenziót adnak az adatoknak, lehetővé téve ezzel a felhasználók számára, hogy különböző szempontokból elemezzék az adatokat. A fact táblák jellemzően nagy adatmennyiséget tárolnak, ami miatt gyakran ők a legnagyobb táblák egy adattárházban.

- **Granularitás:** Meghatározza a fact táblán belüli adatok részletességét. A granularitás lehet nagyon finom, például tranzakciószintű adatok, vagy nagyobb szemcsézettségű, mint például összesített értékesítési adatok egy adott időszakra.
- **Mérők:** Az üzleti teljesítmény mérésére szolgáló kvantitatív adatok, mint például értékesítési összegek, költségek, és tranzakciók száma. Ezek a mérők lehetővé teszik a matematikai műveletek, mint az összegzés, átlagolás, maximum, és minimum végrehajtását.
- **Surrogate Kulcsok:** Egyedi azonosítók, amelyek kapcsolják a fact táblákat a dimenziós táblákhoz. Ezek a kulcsok lehetővé teszik az adatok hatékony lekérdezését és az integritás megőrzését az adattárházban.

A fact táblák kritikus fontosságúak az adattárházakban, mivel ezek tartalmazzák azokat az adatokat, amelyeken az üzleti döntések alapulnak. Az adatok granularitása és a megfelelő

dimenziókhoz való kapcsolódás biztosítja, hogy a felhasználók képesek legyenek komplex lekérdezéseket végezni, előrejelzéseket készíteni, és mélyreható elemzéseket végezni az üzleti folyamatok és teljesítmény megértése érdekében. (Kimball & Ross, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2002) (Inmon, 2005)

2.3.2. Dimenziós Adattárház Modell

Csillag Séma:

A csillag séma az adattárház modellezés egyik legnépszerűbb és legelterjedtebb formája, amelyet kifejezetten az adatok olvasására, lekérdezésére és elemzésére optimalizáltak. A modell nevét onnan kapta, hogy a vizuális struktúrája egy csillagra hasonlít, a középpontban egy nagy faktortáblával és körülötte helyezkednek el a dimenziótáblák, mint a csillag sugarai. A faktortábla az adatok mérőértékeit tartalmazza, mint például értékesítési összegek vagy darabszámok, míg a dimenziótáblák azokat a dimenziókat reprezentálják, amelyek mentén az adatokat analizálni és szegmentálni kívánjuk, mint például idő, földrajz, ügyfelek, termékek.

A csillag séma egyszerűsége és hatékonysága miatt vált az adattárházakban a preferált modellé. Különösen alkalmas olyan környezetekben, ahol a lekérdezés teljesítménye és a felhasználói élmény kiemelt fontosságú. Az egyszerű struktúra megkönnyíti az adatmodellezést, a rendszergazdák és a végfelhasználók számára egyaránt, lehetővé téve gyors és intuitív lekérdezések készítését.

Hópehely Séma:

A hópehely séma a csillag séma egy továbbfejlesztett változata. A fő különbség a csillag séma és a hópehely séma között a dimenziótáblák normalizálásának mértékében rejlik. Míg a csillag séma esetében a dimenziótáblák denormalizáltak, addig a hópehely sémában a dimenziótáblák további normalizálásra kerülnek, ami azt jelenti, hogy a kapcsolódó információkat kisebb, egymással összekapcsolt táblákban tárolják. Ezáltal a hópehely séma több táblát és bonyolultabb kapcsolatokat generál, ami pontosabb adatszervezést tesz lehetővé, de növelheti a lekérdezések bonyolultságát.

A hópehely séma előnyei közé tartozik a tárolási hely optimalizálása, mivel a redundáns adatokat eltávolítják, valamint a dimenzióadatok jobb szervezettsége, ami elősegítheti az

adatok integritásának megőrzését. Ugyanakkor a bonyolultabb lekérdezési logika és az összetettebb kapcsolati struktúrák miatt a hópehely sémát használó rendszerek gyakran lassabbak lehetnek lekérdezések végrehajtása során, mint a csillag séma esetében. (Kimball & Ross, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2002)

2.3.3. Adattárház Modell Implementációja

Az adattárház modell implementációja során többféle adatbázisrendszer közül választhatunk, attól függően, hogy melyik rendszer illeszkedik legjobban az adott üzleti és technikai igényekhez. A választás befolyásolhatja az adattárház teljesítményét, skálázhatóságát, rugalmasságát és kezelhetőségét.

Oszlopos Adatbázisok:

Oszlopos adatbázis kezelő rendszerek, mint például Apache Cassandra, Google Bigtable, és Amazon Redshift, az adatokat oszlopok szerint tárolják, nem sorok szerint. Ez a tárolási módszer optimalizálja az olvasási és írási műveleteket, különösen nagy adathalmazok esetében, és javítja az adattömörítést, valamint az adatelemzés sebességét.

Relációs Adatbázisok:

Relációs adatbázis kezelő rendszerek, mint például az Oracle, PostgreSQL, és Microsoft SQL Server, táblákban tárolják az adatokat, amelyeket kulcsok kötnek össze. Ezek a rendszerek kifejezetten alkalmasak komplex lekérdezések és tranzakciók kezelésére, támogatják az ACID (Atomicity, Consistency, Isolation, Durability) tulajdonságokat, és jól működnek strukturált adatok esetén.

Microsoft SQL Server

A Microsoft SQL Server egy relációs adatbázis-kezelő rendszer, amelyet kifejezetten vállalati környezetek nagy adatmennyiségének kezelésére fejlesztettek ki. Ez a platform támogatja a strukturált és a strukturálatlan adatok kezelését is, lehetővé téve a felhasználók számára, hogy hatékonyan elemezzék és kezeljék az információikat.

Előnyök:

- Integráció: Az SQL Server szorosan integrálható más Microsoft termékekkel és szolgáltatásokkal, mint például az Azure adattárház szolgáltatásokkal, Power BI-val az adatelemzéshez és jelentéskészítéshez, valamint az SQL Server Integration Services (SSIS) segítségével az adatintegráció és -transzformáció terén.
- Teljesítmény: Az SQL Server fejlett adattárolási és lekérdezési optimalizálási képességeket kínál, mint például az adatindexelés és partícionálás, amelyek javítják a lekérdezések teljesítményét.
- Biztonság: Az SQL Server magas szintű adatbiztonsági funkciókat kínál, beleértve az adattitkosítást, hozzáférés-vezérlést és auditálási képességeket.
- Felhasználóbarát: Az SQL Server rendelkezik egy integrált fejlesztői környezettel, az SQL Server Management Studio-val (SSMS), ami megkönnyíti az adatbázis-kezelést és fejlesztést.

(Microsoft SQL documentation, 2024)

2.4. Adattárház betöltése

Az adattárház betöltése egy kritikus lépés az adattárház építési folyamatában, amely magában foglalja az adatok kinyerését, tisztítását, átalakítását és betöltését az operatív rendszerekből az adattárházba. Ez a folyamat biztosítja, hogy az adatok konzisztensek, megbízhatók és az elemzésekhez készen álljanak.

Az adattárházak adatbetöltési eljárásai alapvetően két nagy csoportra oszthatók: teljes adattöltésre (full load) és részleges vagy inkrementális adattöltésre (incremental load).

Teljes Adattöltés (Full Load)

A teljes adattöltés során az adattárház teljes tartalmát egyetlen művelet során töltik be vagy frissítik. Ez azt jelenti, hogy az adatokat először törlik az adattárházból, majd a teljes adatkészletet újra betöltik. Ez hasznos lehet például akkor, ha az adattárház adatszerkezetében jelentős változások történtek, vagy ha az adatok minőségével kapcsolatos aggályok miatt szükséges az adatok teljes újratöltése.

Részleges vagy Inkrementális Adattöltés (Incremental Load)

Az inkrementális adattöltés során csak azok az adatok kerülnek betöltésre, amelyek az előző adattöltés óta változtak vagy lettek hozzáadva. Ez lehetővé teszi az adattárház naprakészen tartását anélkül, hogy az összes adatot újra kellene betölteni, így jelentősen csökkenthető az adattöltéshez szükséges idő és erőforrás. Az inkrementális betöltés gyakran támaszkodik a módosítás időbélyegeire, sorszámokra vagy log fájlokra az új, vagy módosított adatok azonosításához.

2.4.1. Adatcsővezetékek (datapipeline)

A datapipeline, vagy adatcsővezeték, egy adatok áramlását lehetővé tevő technológiai folyamat, amely összekapcsolja az adatforrásokat az adatfogyasztási célpontokkal, mint például adatbázisok, adattárházak, vagy analitikai eszközök. Egy adatcsővezeték célja, hogy automatizálja az adatok átalakítását, mozgatását és tömörítését különböző rendszerek között, miközben biztosítja az adatok integritását és minőségét.

A datapipeline-ok tipikusan négy fő fázisból állnak:

- Adatgyűjtés: Az adatok összegyűjtése különböző forrásokból, mint például online tranzakciók, naplófájlok, szenzor adatok, stb.
- Adattisztítás és -átalakítás: Az összegyűjtött adatok megtisztítása (pl. duplikációk eltávolítása, hiányzó értékek kezelése) és átalakítása az adattárház vagy analitikai eszközök által szükséges formátumra.
- Adattárolás: Az átalakított adatok tárolása adatbázisokban, adattárházakban vagy a felhőben.

- Adatelemzés és -megjelenítés: Az adatok elemzése és megjelenítése üzleti intelligencia (BI) eszközök és irányítópultok segítségével, hogy értékes betekintéseket nyújtson az üzleti döntéshozóknak.

Az adatcsővezetékek kialakítása során fontos figyelembe venni az adatok biztonságát és adatvédelmet, valamint az adatfolyamatok megbízhatóságát és skálázhatóságát. A különböző eszközök és platformok, mint például Apache Kafka, Apache Nifi, és a felhő alapú szolgáltatások (pl. AWS Data Pipeline, Google Cloud Dataflow, és Azure Data Factory) lehetővé teszik a vállalatok számára, hogy rugalmasan és hatékonyan hozzanak létre adatcsővezetékeket a különféle adatkezelési igények kiszolgálására.

Betöltési folyamatok

- **Valós idejű:** A streaming betöltési folyamatok lehetőséget adnak az adatok folyamatos feldolgozására és analízisére valós időben. Ezek a folyamatok különösen hasznosak olyan alkalmazásokban, ahol az azonnali adatelemzés kritikus, mint például a pénzügyi tranzakciók nyomon követése, online ügyfélinterakciók vagy érzékelőadatok elemzése. Az adatokat közvetlenül az adatforrásból olvassák és azonnal feldolgozzák, minimalizálva az adatkésleltetést.
- **Ütemezett:** A batch betöltési folyamatok, másrészt, nagy mennyiségű adatot gyűjtenek össze és egy adott időpontban vagy időintervallumon belül dolgozzák fel. Ez a módszer ideális nagy, strukturálatlan adatkészletek kezelésére, ahol az adatfeldolgozás nem igényel azonnali választ. A batch folyamatok lehetővé teszik az adatok hatékonyabb kezelését, csökkentve a számítási erőforrásokra nehezedő terhet, de általában nagyobb késleltetéssel járnak az adatok elemzésében és elérhetőségében. Adattárházak betöltésénél ez a legjellemzőbb módszer.

Az adatcsővezetékek kulcsfontosságúak a modern adatvezérelt vállalatok számára, lehetővé téve számukra, hogy gyorsan és hatékonyan reagáljanak az üzleti kihívásokra, optimalizálják működésüket, és javítsák ügyfélélményüket az adatokból nyert betekintések révén. (Apache Kafka, 2024) (Apache Nifi Documentation, 2024) (Google Dataflow, 2024)

2.4.2. Datapipeline Eszközök

A piac számos olyan szoftveres megoldást kínál, amelyek lehetővé teszik adatvezetékeink ütemezését és karbantartását, biztosítva ezzel az adatfolyamatok zökkenőmentes és hatékony működését.

Apache Airflow:

Az Apache Airflow egy népszerű nyílt forráskódú munkafolyamat-kezelő eszköz, amely különösen alkalmas adatpipeline-ok és ETL folyamatok kezelésére. Előnyei közé tartozik a bonyolult függőségek hatékony kezelése, időalapú ütemezés, széleskörű operátorok választéka, felhasználóbarát kezelőfelület, és az adatok immutabilitásán (változatlanságán) és idempotenciáján (azonos bemenetre mindig azonos kimenetet ad) alapuló tervezés. Ezek a tulajdonságok lehetővé teszik az adatfolyamatok átlátható kezelését és az adatok könnyű átalakítását különböző feladatok között. Emellett az Airflow egy aktív közösséggel rendelkező nyílt forráskódú projekt, ami elősegíti a felhasználói problémák gyors megoldását.

Ugyanakkor az Airflow használata bizonyos kihívásokkal is járhat. Kezdők számára az adatfeldolgozási modell nem mindig intuitív, és a tesztesetek írása nehézségeket okozhat. A folyamatok ütemezésének megváltoztatása a DAG (Directed Acyclic Graph) átnevezését igényli, ami bonyolultsághoz vezethet. A CI/CD folyamatok implementálása, különösen Docker használata esetén, trükkös lehet, mivel az újraindítások befolyásolhatják a folyamatban lévő munkákat. Továbbá, nincs natív Windows támogatás, bár ez Docker használatával kezelhető.

(Start Data Engineering Apache Airflow, 2024)

Dagster:

Dagster egy nyílt forráskódú adatirányítási platform, amely az adatvezetékek (data pipelines) fejlesztését, üzemeltetését és karbantartását célozza meg. A Dagster egyedülállóan ötvözi az adatfolyamatok definícióját, végrehajtását és monitorozását, egy integrált fejlesztői környezetben, ami kifejezetten az adatmérnökség és adattudomány területén dolgozó szakemberek igényeire szabott. A platform keretrendszere lehetővé teszi a felhasználók számára, hogy rugalmasan kezeljék az adatfolyamatokat, támogatva ezzel a komplex adattranszformációkat és az adattovábbítási munkafolyamatokat.

A Dagster kiemelkedő jellemzője a „DAG” (Directed Acyclic Graph) alapú megközelítés, amely strukturált és könnyen követhető módon ábrázolja az adatfolyamatokat és azok függőségeit. Ez a megközelítés segíti a felhasználókat az adatvezetékek pontos tervezésében és megvalósításában, valamint javítja a hibakeresés hatékonyságát és a folyamatok újrafelhasználhatóságát.

Dagster támogatja az adattípus-ellenőrzést és a tesztelést, ami hozzájárul az adatminőség javításához és az adatvezetékek megbízhatóságának növeléséhez. A platform lehetővé teszi a felhasználók számára, hogy részletesen megadjanak adattípusokat és ellenőrzéseket, amelyek biztosítják, hogy az adatfolyamat minden lépése a megfelelő adatformátumokat és struktúrákat használja.

Dagster integrálódik számos adattárolóval, adatfeldolgozási keretrendszerrel és üzenetküldő rendszerrel, így biztosítva a felhasználók számára a széles körű alkalmazhatóságot és rugalmasságot. A platform kiterjeszthető architektúrája lehetővé teszi a felhasználók számára, hogy saját bővítményeket és integrációkat fejlesszenek, amelyek tovább bővítik a Dagster alkalmazási területeit. (Dagster Docs, 2024)

2.4.3. ETL (Extract, Transform, Load)

Az ETL (Extract, Transform, Load) folyamat az adatcsővezetékek kulcsfontosságú része, amely lehetővé teszi az adatok különböző forrásokból történő gyűjtését, üzleti szabályok szerinti átalakítását, majd cél adattárolóba való betöltését.

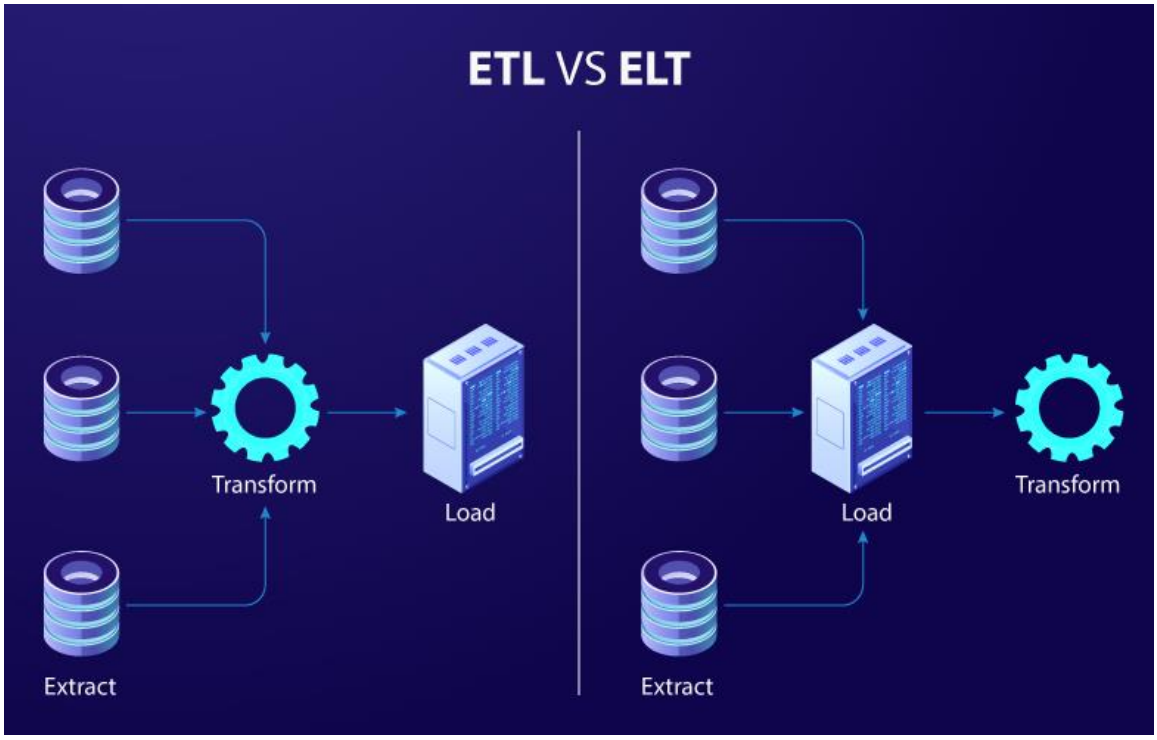
ETL Folyamat:

1. Extract (Kinyerés): Az adatok kinyerése az adatforrásokból. Ez magában foglalja az adatok összegyűjtését és átvitelét az adattárházba. Ez lehet teljes adatkinyerés vagy inkrementális (csak az új vagy megváltozott adatok kinyerése).
2. Transform (Átalakítás): Az adatok átalakítása az adattárházba való betöltés előtt. Ide tartozik az adatok tisztítása, normalizálása, aggregálása vagy akár új számított mezők létrehozása. Ez a lépés gyakran időigényes és erőforrásigényes, mivel az adatokat átalakítani kell az adattárházban történő tároláshoz és elemzéshez.
3. Load (Betöltés): Az átalakított adatok betöltése az adattárházba. Ez lehet teljes vagy inkrementális betöltés, attól függően, az adatok mennyiségétől és gyakoriságától.

ELT Folyamat:

- Extract (Kinyerés): Az adatok kinyerése az eredeti forrásrendszerből, ugyanúgy, mint az ETL esetén.
- Load (Betöltés): Az adatok betöltése az adattárházba még az átalakítás előtt. Az adatokat általában egy nyers formában vagy minimális átalakítással töltik be az adattárházba. Ez lehetővé teszi az adatok gyors betöltését és a tárolást anélkül, hogy azokat előzetesen nagyobb mértékben átalakítanák.
- Transform (Átalakítás): Az adatok átalakítása az adattárházban, miután már be vannak töltve. Az adatokat az adattárházban történő tárolás után alakítják át és tisztítják megfelelő formátumra és szerkezetre. Ez lehetővé teszi az adatok azonnali elérhetőségét a betöltés után, valamint nagyobb rugalmasságot biztosít az adatfeldolgozási folyamatokban.

3. ábra
ETL vs ELT
(ETL vs ELT – understanding the key differences, 2024)



Összefoglalás:

Az ETL és az ELT megközelítések közötti fő különbség a Transform (Átalakítás) lépés helyzetében van. Az ETL esetében az átalakítás a kinyerés és a betöltés között történik, míg az ELT esetében

az átalakítás a betöltés után következik be. Az ETL hagyományosan használatos a strukturált adatokhoz és a klasszikus adattárházakhoz, míg az ELT gyakran alkalmazható a nagy adatmennyiségekkel és a rugalmasabb adatfeldolgozási igényekkel rendelkező környezetekben. Mindkét megközelítésnek vannak előnyei és hátrányai. (3. ábra)

A modern felhő alapú adattároló platformok, mint például a Snowflake, kiemelkedő előnyt nyújtanak az ELT megközelítésben, mivel rendkívül nagy számítási kapacitással és dinamikusan skálázható infrastruktúrával rendelkeznek. Ennek eredményeként az adatbetöltés és az átalakítás együttesen, azaz az adatok betöltése után történő átalakítás, hatékonyan végezhető el a platform által biztosított erőforrások optimalizált felhasználásával. Ezáltal lehetővé válik a gyors adatfeldolgozás és az analitikai műveletek végzése a nagy volumenű és változatos adatokon, ami kiemelkedő fontossággal bír az üzleti döntéshozatal és az innováció szempontjából. (Kimball & Ross, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2002) (Apache Nifi Documentation, 2024)

2.5. ETL Eszközök

Az ETL (Extract, Transform, Load) eszközök kulcsfontosságúak az adatok integrálásában, lehetővé téve vállalatok számára, hogy több forrásból származó adatokat egyetlen központi adattárba összesítsenek. Az ETL folyamat segít javítani az adatminőséget, egységesíti az adatokat az elemzés megkönnyítése érdekében, és gyorsítja a döntéshozatali folyamatokat.

2.5.1. Pentaho Data Integration (PDI)

A Pentaho Data Integration (PDI), mely Java alapú rendszer, kiválóan alkalmazható számos ETL folyamat megvalósítására. Ennek ellenére néhány bonyolultabb folyamat implementálása kihívást jelenthet a platformon belül, különösen azok számára, akik mélyreható programozási logikát igénylő transzformációkat kívánnak végrehajtani. A PDI ingyenes Community Edition változata széles körben elérhető, azonban az ehhez kapcsolódó dokumentáció hiányos lehet, ami megnehezítheti az új felhasználók számára a rendszer mélyebb megértését és kiaknázását. Emellett a Pentaho Data Integration fizetős, Enterprise Edition változata magas árcímkével rendelkezik, amely költségvetési korlátok miatt nem minden szervezet számára elérhető. (Datatheta Getting Started with Pentaho Data Integration, 2024)

2.5.2. Microsoft SSIS

A Microsoft SSIS (SQL Server Integration Services) egy kiemelkedően erőteljes ETL (Extract, Transform, Load) eszköz, amely adatintegrációs és adattranszformációs célokra jött létre. Mint a Microsoft SQL Server egyik összetevője, az SSIS támogatja a széles körű adatforrások integrációját és képes komplex adattranszformációs, valamint adattisztítási műveletek elvégzésére.

Előnyök:

- Többféle adatforrás kezelése: Az SSIS alkalmazkodik a heterogén adatforrásokhoz, támogatva olyan forrásokat, mint az FTP, HTTP, MSMQ, és az Analysis Services.
- Komplex transzformációs képességek: Széleskörű eszközöket kínál az adatok átalakításához, beleértve az aggregálást, egyesítést és módosítást.
- Szoros integráció: Közvetlen kapcsolatot biztosít a Microsoft Visual Studio és az SQL Serverrel, megkönnyítve a projektkezelést és konfigurációt.
- Testreszabhatóság és kiterjeszthetőség: Lehetővé teszi a szkriptek újrafelhasználását és egyedi komponensek létrehozását a fejlesztők számára.

Hátrányok:

- Jelentéskészítési korlátok: A csomagok végrehajtási jelentéseinek megtekintése az SQL Server Management Studio használatát igényli, korlátozva a jelentések elérhetőségét.
- Magas memóriahasználat: Több párhuzamosan futtatott csomag memóriaigénye konfliktusokhoz vezethet az SSIS és az SQL Server között.
- CPU-allokáció: A párhuzamos csomagvégrehajtások során figyelemmel kell lenni a processzorforrások megfelelő elosztására, hogy elkerüljük az SSIS teljesítménybeli csökkenését.
- Korlátozott harmadik féltől származó támogatás: Az SSIS kompatibilis eszközök és bővítmények megtalálása nehézségekbe ütközhet, ami korlátozhatja a felhasználhatóságát bizonyos szituációkban.

Összegzésben az SSIS egy rendkívül hatékony és rugalmas eszköz adatintegrációs és transzformációs feladatokhoz, amelynek használata jelentős előnyökkel járhat a megfelelő alkalmazási környezetben, annak ellenére, hogy bizonyos hátrányai is vannak, különösen a

rendszerforrás-igény és a jelentéskészítési korlátok terén. (Sarjen SSIS Advantages-Disadvantages, 2024)

2.5.3. Python ETL (custom ETL development)

A Python egy magas szintű, általános célú programozási nyelv, amely a könnyű olvashatóságot és a hatékony programkód írását helyezi előtérbe. Guido van Rossum hozta létre 1991-ben. A nyelv egyszerű szintaxisa lehetővé teszi a programozók számára, hogy kevesebb kódsorral több funkciót valósítsanak meg, ami Python-t kiváló választássá teszi kezdők és haladók számára egyaránt. Széles körben használják webfejlesztés, adatbányászat, tudományos számítások és mesterséges intelligencia területén. (Python, 2024) (GeeksforGeeks - History of Python, 2024)

A Python nyelven manuálisan implementált ETL folyamatok kidolgozása lehetőséget nyújt az adatmérnökök és adattudósok számára, hogy teljes mértékben testre szabhassak és optimalizálhassák az adatkezelési folyamatokat az egyedi üzleti igényeknek megfelelően. Ennek során Python kód segítségével végezhetik el az adatok kinyerését, átalakítását és betöltését különböző adatforrásokból és adatcélpontokba.

Előnyök

- Testreszabhatóság: A Pythonban megírt ETL folyamatok teljes mértékben testreszabhatóak, lehetővé téve, hogy pontosan megfeleljenek az adott adatkezelési és üzleti igényeknek.
- Rugalmas adatkezelés: Python segítségével könnyen kezelhetőek a különböző adatformátumok, és a különböző adatforrásokból származó adatok könnyen integrálhatóak.
- Közösségi támogatás: A Python rendkívül népszerű programozási nyelv, így széles körű közösségi támogatást és rengeteg könyvtárat kínál az adatkezeléshez.
- Költséghatékony megoldás: Nem igényel drága licencdíjakat vagy harmadik féltől származó eszközök beszerzését, így költséghatékony megoldást nyújthat.
- Kód újra felhasználhatósága: Egyszer megírt Python ETL folyamatok vagy scriptek könnyen adaptálhatóak más projektekhez, ami időt és erőforrásokat takarít meg.

Hátrányok

- Fejlesztési idő: Az ETL folyamatok nulláról történő megírása több időt és erőforrást igényelhet, mint a kész eszközök használata.
- Karbantartási kihívások: A testreszabott kód karbantartása és frissítése időigényes lehet, különösen, ha az adatstruktúrák vagy üzleti igények változnak.
- Szakértelem szükségessége: A hatékony és optimalizált ETL folyamatok Pythonban történő megírása mélyreható programozási és adatkezelési ismereteket igényel.
- Teljesítmény kérdések: Bár a Python rugalmas és hatékony adatkezelést tesz lehetővé, bizonyos esetekben a teljesítménye nem éri el a speciálisan ETL feladatokra tervezett eszközökét, különösen nagy adatkészletek esetén.

(ActiveBatch - ETL Automation with Python, 2024) (Stich - Using Python for ETL, 2024)

Leggyakrabban használt Python modulok:

SQLAlchemy**:**

Az SQLAlchemy egy átfogó Python SQL eszközkészlet és objektum-relációs leképező (ORM), amely lehetővé teszi az alkalmazásfejlesztők számára, hogy teljes mértékben kihasználják az SQL nyújtotta lehetőségeket és rugalmasságot. Hatékony, nagy teljesítményű adatbázis-hozzáférést biztosít, bemutatva a jól ismert vállalati szintű tartósági mintákat egy egyszerű, Python alapú domain nyelven. Ezáltal ideális választás a fejlesztők számára, akik intuitívabb és Python-barát módon szeretnének interakcióba lépni az adatbázisokkal.

Az SQL lehetővé teszi a felhasználók számára, hogy lekérdezéseket hajtsanak végre, adatokat szúrjanak be, frissítsenek vagy töröljenek, valamint szerkezeti változtatásokat végezzenek az adatbázison, mint például táblák létrehozása és módosítása. Az SQL nyelv kifejezetten az adatok kezelésére lett tervezve, így rendkívül hatékony eszközt nyújt az adatok szűrésére, rendezésére és összetett lekérdezések végrehajtására. (SQLAlchemy Docs, 2024)

Pandas**:**

A Pandas egy rendkívül hasznos eszköz az ETL (Extract, Transform, Load) folyamatok építésében, különösen az adatok ideiglenes tárolására és manipulálására. Ez a Python könyvtár lehetővé teszi adatstruktúrák, mint a DataFrame-ek kezelését, amelyekkel hatékonyan végezhetők el az adatok kinyerése, átalakítása és betöltése különböző forrásokból. A Pandas

funkciói magukban foglalják az adatok importálását, tisztítását, előfeldolgozását, és analízisét, így kulcsfontosságú szerepet játszik az adatok előkészítésében és elemzésében az adattudományi projektek során. (Pandas Docs, 2024)

2.6. Adatvizualizáció

Az adatvizualizáció az adatelemzés és üzleti intelligencia kulcsfontosságú eszköze, amely lehetővé teszi összetett adatkészletek vizuális formában való ábrázolását. A hatékony vizualizáció segíti az adatok gyorsabb értelmezését és megértését, elősegítve az információs minták, trendek és összefüggések felfedezését. A grafikonok, diagramok, térképek és interaktív dashboardok használata növeli az adatok hozzáférhetőségét és érthetőségét, segítve a döntéshozókat az informáltabb döntések meghozatalában. Az adatvizualizáció tehát nem csak esztétikai eszköz, hanem egy alapvető analitikai technika, amely elősegíti az adatokból származó ismeretek kommunikálását.

Az adatvizualizáció részletes megközelítése során érdemes különbséget tenni a statikus és interaktív vizualizációk között. A statikus vizualizációk, mint a diagramok és grafikonok, egyszerű, könnyen érthető ábrázolásokat kínálnak, amelyek segítségével az adatok fő jellemzőit gyorsan fel lehet dolgozni. Az interaktív dashboardok és térképek viszont lehetővé teszik a felhasználó számára, hogy mélyebbre ásson az adatokban, szűrőket és választásokat alkalmazva felfedezze az adatok rejtett összefüggéseit. Az adatvizualizáció így áthidalja a szakértői tudás és a döntéshozatali folyamatok közötti szakadékot, lehetővé téve minden szintű felhasználó számára, hogy teljes mértékben kiaknázzák az adatokban rejlő értékeket.

2.6.1. Adatvizualizációs eszközök

Az adatvizualizációs eszközök, mint például a Tableau, Power BI, és Google Data Studio, lehetővé teszik a felhasználók számára, hogy összetett adatokat vizuálisan ábrázoljanak, segítve az információk könnyebb megértését és prezentálását. Ezek az eszközök támogatják az interaktív dashboardok létrehozását, amelyek dinamikus elemzést és adatfelfedezést tesznek lehetővé, így az üzleti felhasználók és döntéshozók azonnali betekintést nyerhetnek az adatokba. Az adatvizualizációs szoftverek széleskörű integrációt kínálnak adatforrásokkal, és

rugalmas formázási lehetőségeket biztosítanak a felhasználói igények széles skálájának kielégítésére.

Tableau

A Tableau egy vezető adatvizualizációs eszköz, amelyet kifejezetten az adatok vizuális elemzésére és megosztására terveztek. Segítségével a felhasználók könnyen hozhatnak létre komplex adatvizualizációkat, interaktív dashboardokat, és adattérképeket anélkül, hogy mély programozási ismeretekkel rendelkeznének. A Tableau képes kezelni nagy mennyiségű adatot, és támogatja a különböző adatforrásokból származó adatok integrálását, lehetővé téve a felhasználók számára, hogy gyorsan és intuitív módon nyerjenek betekintést az adatokba, felfedezzék azok rejtett összefüggéseit, és hatékony döntéstámogatási információkat állítsanak elő. (Tableau, 2024)

Microsoft Power BI

A Power BI egy Microsoft által fejlesztett adatvizualizációs eszköz, amely lehetővé teszi a felhasználók számára, hogy adatokat gyűjtsenek, elemzéseket végezzenek és megosztásra kész, interaktív jelentéseket készítsenek. Az eszköz kiemelkedik integrációs képességeivel, mivel zökkenőmentesen kapcsolódik számos adatforráshoz, és felhasználóbarát interfészt kínál az adatok felfedezéséhez és vizualizálásához. A Power BI erősítette pozícióját az üzleti intelligencia piacon azáltal, hogy demokratizálja az adatelemzést, lehetővé téve a nem technikai háttérrel rendelkező felhasználók számára is, hogy értékes betekintéseket nyerjenek az adatokból. (Power BI, 2024)

3. Eszközök és módszerek

Ebben a szekcióban kitérek arra, hogy milyen szoftvereket, keretrendszereket és programozási nyelveket választottam a projekt során, valamint ismertetem azokat a módszereket, amelyek segítségével az adatokat feldolgoztam, analizáltam és vizualizáltam.

3.1. Adatbázis-kezelő rendszer

Microsoft SQL Server

Az adattárházam számára az SQL Servert választottam ki, elsősorban annak számos előnye, amit az (2.3) fejezetben kifejtettem és a vállalati irányelvünk miatt, amely előtérbe helyezi a Microsoft termékeinek használatát és különösen a Power BI integráció miatt.

3.2. ETL eszköz

Python ETL (custom ETL development)

Az (2.5) fejezetben felsorolt és kifejtett lehetőségek és alkalmazások áttekintése után arra a következtetésre jutottam, hogy a Python nyelven manuálisan implementált ETL folyamatok a legmegfelelőbbek az igényeimhez. Ez a döntés több tényezőn alapul, beleértve a Python nyelv általános elérhetőségét, a széles körű közösségi támogatást és a programozási nyelvnek a testre szabható adatkezelési folyamatok terén mutatott erősségeit. Ennek eredményeként a fejlesztés során tervezek integrálni egy folyamatütemezőt is, amely segítségével hatékonyan kezelhetem és automatizálhatom az ETL munkafolyamatok végrehajtását.

3.3. Datapipeline Eszközök

Dagster

Az ETL folyamataim kezelésére és ütemezésére a Dagster-t választottam a (2.4) fejezeteben kifejtett számos előny miatt. A Dagster egy erőteljes és rugalmas eszköz az adatmérnökök és

adattudósok számára, amely támogatja a komplex adatvezetékek hatékony tervezését, végrehajtását és monitorozását. A platform elősegíti az adatvezérelt projektek sikeres megvalósítását, optimalizálva ezzel az adatfolyamatok kezelését és az adatok értékének maximalizálását. Az Apache Airflow szintén jó választás lett volna, azonban az implementációs korlátok miatt különösen a Windows Server környezetbe való telepítés nehézségei miatt végül el kellett vetnem ezt a lehetőséget.

3.4. Adatvizualizációs eszköz

Microsoft Power BI

A Microsoft Power BI alkalmazása a WHC Kft.-nél természetes választás volt, tekintettel arra, hogy adatvizualizációs igényeinket kizárólag ezzel a rendszerrel elégítjük ki. Ennek fényében, az adatbázis-kezelő rendszerünk kiválasztását is részben befolyásolta a Power BI-al való integrációs képesség, amely mellett számos további előnnyel is rendelkezik, amiket részletesen tárgyalok az (2.6) fejezetben.

4. Eredmények és értékelésük

4.1. HR Adattárház modell

Az (2.3) fejezetben kifejtettem, hogy milyen módszerek és gyakorlatok vannak az adattárházak modellezésére, ezeket alkalmaztam a saját modellem kialakítása során.

4.1.1. Dimenzió táblák

Dimenziók tervezése:

Ezek a táblák tartalmazzák a faktortábla adatainak dimenzióit, amelyek mentén elemzést végezhetünk. A dimenziótáblák denormalizáltak, vagyis minden lehetséges attribútumot egyetlen táblában tárolnak a gyorsabb lekérdezések érdekében. Az én esetemben 8 ilyen adatbázis tábla került kialakításra, amit a lenitekben részletesen kifejtek.

Dim_Person: Ez a dimenzió tábla (1. táblázat), amely tartalmazza a munkavállalók személyes adatokkal kapcsolatos részletes információit, mely lehetővé teszi a felhasználók pontos azonosítását, kategorizálását és az adatok alapján történő részletes elemzést.

1. táblázat
Dim_Person – Statikus modell

Dim_Person			
Oszlop neve	Adattípus	NULL Engedélyezése	Jellemzés
persid (PK)	int	Nem	Személy rendszer egyedi azonosítója (Elsődleges kulcs)
birthdate	date	Igen	Születési dátum
name	varchar(128)	Nem	Név
birthcountry	varchar(128)	Igen	Születési ország
birthplace	varchar(128)	Igen	Születési hely
perscode	varchar(32)	Igen	Törzsszám
taxcode	varchar(12)	Nem	Adóazonosító
scn	varchar(12)	Nem	Társadalombiztosítási azonosító
addrtypeactual	varchar(32)	Nem	Cím típusa
gender	varchar(1)	Igen	Nem
country	varchar(128)	Igen	Állampolgárság
country2	varchar(128)	Igen	Állampolgárság (kettős állampolgár)
birthnamefull	varchar(128)	Igen	Születési név
mothernamefull	varchar(128)	Igen	Anyja teljes neve
load_date_time	datetime	Igen	Adatok betöltés ideje

Dim_Accommodation: Ez a dimenzió tábla tartalmazza azon munkavállalók szállás adatait, akik elszállásolásban részesülnek.

Dim_Employment: Ez a dimenzió tábla, amely a kölcsönzötti állomány munkaviszonyával kapcsolatos információkat tárolja, mely lehetővé teszi a jogviszony alapján történő részletes elemzést.

Dim_Ktgh: Költséghely kódokhoz tartozó megnevezéseket tartalmazza, ami alapján megállapítható egy adott személy melyik irodából lett kikölcsönözve.

Dim_Project: Project kódokhoz tartozó megnevezéseket tartalmazza, ami alapján megállapítható egy adott személy melyik partnerhez lett kikölcsönözve.

Dim_Application: Különböző nyelvű chatbotokat használunk a cégnél és ezzel a táblával tudjuk meghatározni, hogy adott személy melyik nyelvű chatbot rendszerhez van hozzárendelve.

Dim_Login: Ezzel a dimenziótáblával tudjuk nyomon követni a felhasználók státuszát, hogy egy adott munkavállaló mikor került be a rendszerbe, bejelentkezett-e, mikor jelentkezett be először vagy mikor került törlésre a rendszerből.

Dim_Date: Lehetővé teszi az adatok időbeli analízisét. A dátum dimenzió tábla tartalmaz minden lehetséges dátumot egy előre meghatározott időintervallumban (2023-től 2031-ig), minden egyes dátumhoz kapcsolódó attribútumokkal

4.1.2. Fact tábla

Az én esetemben nem hagyományos Fact tábla struktúrát alkalmaztam, amikor egy adattárház nem aggregált mérőket (pl. összegeket, átlagokat), hanem tranzakciós vagy eseményszintű részleteket tartalmaz, ez a struktúra kevésbé tipikus, de a projektem esetében ez bizonyult a legmegfelelőbb választásnak mert nagyon hasznos lehet bizonyos analitikai feladatokhoz. Ebben az esetben a faktortábla a dimenziótáblákhoz kapcsolódó dimenziós kulcsokat és a tranzakciók vagy események egyedi azonosítóit tartalmazza, mint a felhasználók be- és kijelentkezéseit egy rendszerben. Ez lehetővé teszi az események pontos nyomon követését és elemzését az idő függvényében. (2. táblázat)

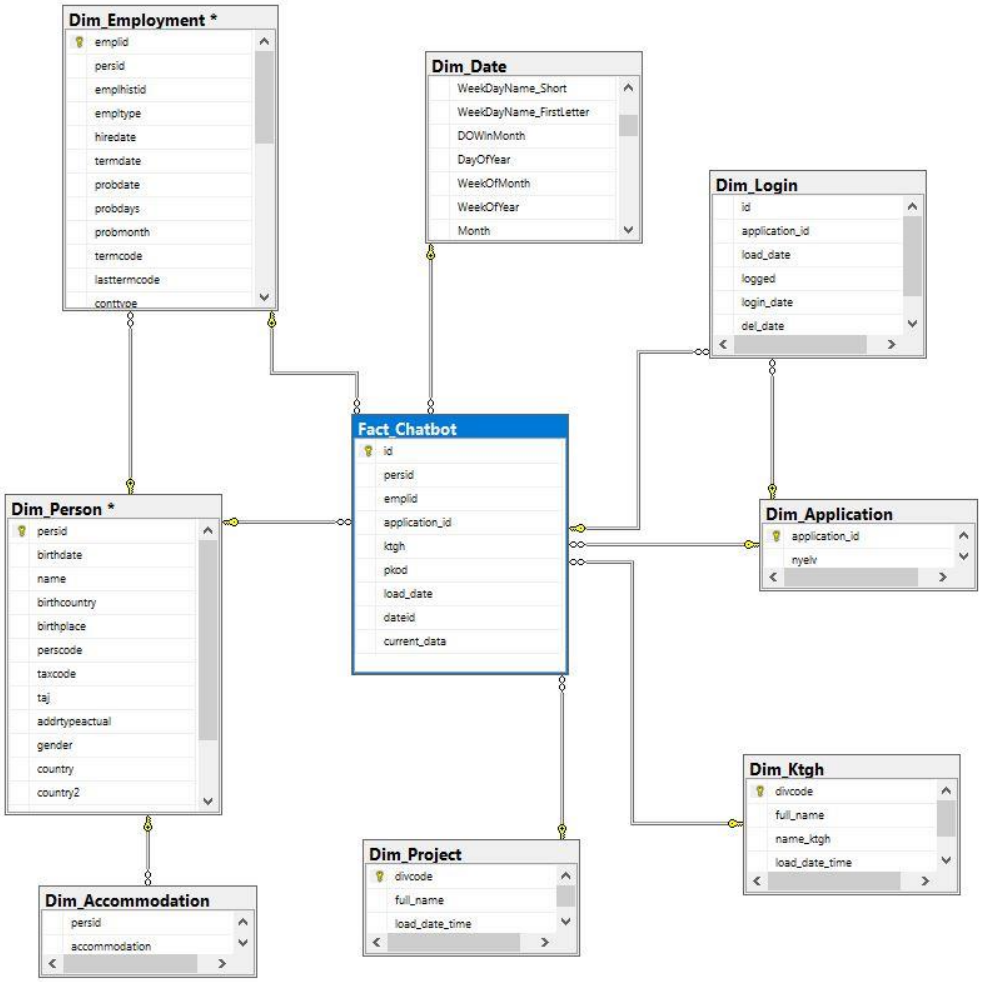
2. táblázat
Fact_Chatbot – Statikus modell

Fact_Chatbot			
Oszlop neve	Adattípus	NULL Engedélyezése	Jellemzés
id (PK)	varchar(256)	Nem	Fact tábla egyedi azonosítója (Elsődleges kulcs)
persid (FK)	int	Nem	Dim_Person tábla egyedi azonosítója (Idegen kulcs)
emplid (FK)	int	Nem	Dim_Employment tábla egyedi azonosítója (Idegen kulcs)
application_id (FK)	int	Nem	Dim_Application tábla egyedi azonosítója (Idegen kulcs)
ktgh (FK)	varchar(128)	Nem	Dim_Ktgh tábla egyedi azonosítója (Idegen kulcs)
pcode (FK)	varchar(128)	Nem	Dim_Project tábla egyedi azonosítója (Idegen kulcs)
load_date	date	Nem	Adatok betöltés ideje
dateid (FK)	int	Nem	Dim_Date tábla egyedi azonosítója (Idegen kulcs)
current_data	int	Nem	Aktuális adatok jelölésé

4.1.3. Adattárház modell

Az adattárházam kialakításánál a csillag sémát használtam mert ez illet leginkább a projekt követelményeihez és kifejezetten az adatok olvasására, lekérdezésére és elemzésére optimalizáltak. A (2.3.2) fejezetben részletesen kifejtettem a sémákat. (4. ábra)

4. ábra
Adattárház diagram



4.2. ETL folyamatok kialakítása

Ebben a részben mutatom be a Python nyelven létrehozott ETL folyamataimat. Az adatok kinyerésére és betöltésére a Pythonban SQLAlchemy adatbázis-eszözkészletet használtam és az adatok ideiglenesen a Pandas modul által biztosított DataFrame-ben tároltam a folyamataim között, ezekről a Python modulokról a (2.5.3) részben esett szó.

4.2.1. Extract (Kinyerés)

Az Abacus Bér által használt bérinformációk az Oracle Database 19c adatbáziskezelő rendszerben helyezkednek el, melyek a cégünk belső hálózatán belüli szervereken találhatóak

meg. A Chatbot rendszer adatbázisa nem nálunk található meg, de rendelkezem az adatbázishoz távóli hozzáféréssel és olvasási jogosultsággal, a rendszer fejlesztő-üzemeltető PostgreSQL adatbázist használ.

4.2.2. Adatbázis csatlakozás

Az SQLAlchemy segítségével létrehoztam egy „Engine”-t (5. ábra), ami a Python alkalmazás és az adatbázis közötti kommunikációt hivatott megkönnyíteni. Az Engine a SQLAlchemy magját képezi, lényegében egy központi adatbázis kapcsolódási pontot biztosít, ahol az adatbázis-specifikus paraméterek, mint például a cím (IP), a hitelesítési adatok, és a kapcsolatkezelési stratégiák konfigurálhatók. Az Engine létrehozása az adatbázis felé történő első lépés, ami lehetővé teszi, hogy az alkalmazás SQL utasításokat hajtson végre az adatbázison keresztül, anélkül, hogy közvetlenül kezelnünk kellene az alacsony szintű kapcsolatkezelést vagy a különböző adatbázisok közötti kisebb eltéréseket.

5. ábra
Oracle adatbázis kapcsolat

```
def oracle_engine():

    engine = None
    MYDB='(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=255.255.255.255)
    (PORT=1521))(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=oracle.db)))'
    user=os.environ['ORAUID']
    password=os.environ['ORAPASS']

    try:
        engine=sqlalchemy.create_engine(
            f'oracle+oracledb://{user}:{password}@{MYDB}')
    except exc.OperationalError as ex:
        print(f"OperationalError a OracleDB csatlakozáskor: {ex}")
    except exc.ProgrammingError as ex:
        print(f"ProgrammingError a OracleDB csatlakozáskor: {ex}")
    except Exception as ex:
        print(f"Hiba a OracleDB csatlakozáskor: {ex}")
    return engine
```

Az újra felhasználhatóságot szem előtt tartva a Python Engine-ket funkcióként implementáltam, amely lehetővé teszi azok hatékony újra felhasználását különböző lekérdezések esetén.

4.2.3. Adatok lekérdezése:

Az ETL folyamatom Extract szakaszában a SQL lekérdezéseimet próbáltam úgy kialakítani, hogy azok nem csupán az adatok kinyerését végzik, hanem egyfajta előfeldolgozást és tisztítást is magukban foglalnak. Ennek keretében kizárólag a szükséges oszlopokat választom ki az adatbázistáblákból, valamint az adatok előszűrését is elvégzem, ezzel biztosítva, hogy csak a releváns és tiszta adatok kerüljenek további feldolgozásra. Ez a megközelítés javítja az adatok minőségét és hatékonyságot kínál a későbbi szakaszokban.

6. ábra
Abacus Bér oldali lekérdezés

```
def abperson():


    engine = oracle_engine()

    if engine is not None:

        try:
            sql_query = pd.read_sql_query('''
                                select PERSID, BIRTHDATE, NAME,
                                (select name from ABDICDAT where DICCODE =
                                ABPERSON.BIRTHCOUNTRY) BIRTHCOUNTRY,
                                BIRTHPLACE,PERSCODE,TAXCODE,TAJ,ADDRTYPEACTUAL,
                                (select name from ABDICDAT where DICCODE =
                                ABPERSON.GENDER) GENDER,
                                (select name from ABDICDAT where DICCODE =
                                ABPERSON.COUNTRY) COUNTRY,
                                (select name from ABDICDAT where DICCODE =
                                ABPERSON.COUNTRY2) COUNTRY2,
                                BIRTHNAMEFULL, MOTHERNAMEFULL
                                from ABPERSON
                                where deleted = '0'
                                and persid in
                                (select distinct persid from ABEMPL where
                                deleted = '0' and termdate >= '2022.01.01')
                                ''', engine)

            print(sql_query)
            return sql_query

        except exc.OperationalError as ex:
            print(f"OperationalError a lekérdezés során: {ex}")
        except exc.ProgrammingError as ex:
            print(f"ProgrammingError a lekérdezés során: {ex}")
        except Exception as ex:
            print(f"Hiba a lekérdezés során: {ex}")

    else:
        print(f"Hiba a OracleDB csatlakozáskor.")
```

A bemutatott kódrészlet (6. ábra) jól szemlélteti, hogy az SQLAlchemy és a Pandas könyvtárak felhasználásával könnyedén sikerült a lekérdezett adatokat egy DataFrame objektumban tárolnom. Ez a módszer lehetővé tette számomra, hogy az adatmanipulációs és -analízis folyamatokat egyszerűen és strukturáltan hajthassam végre.

4.2.4. Staging környezet

Az ETL folyamat Extract szakaszában kinyert adatokat egy ideiglenes, úgynevezett 'staging' adatbázisban tárolom el. Ez a módszer egy köztes lépést jelent az adatok kinyerése és a végleges transzformáció elvégzése között. A staging adatbázis használata növeli a folyamat rugalmasságát és biztonságát, mivel lehetővé teszi az adatok további előkészítését, tisztítását és validálását. Ez a gyakorlat biztosítja, hogy csak előzetesen ellenőrzött és megfelelően előkészített adatok kerüljenek a transzformációs lépésbe, javítva ezzel az adatok minőségét és az egész ETL folyamat hatékonyságát.

A staging adatbázis létrehozását SQL Server környezetben végeztem el, amely azonos platformon található, ahol később az adattárházam is helyet kap. Ez a megközelítés számos előnnyel jár. Azonos infrastruktúrán való működés optimalizálja az adatátvitel sebességét és csökkenti a kompatibilitási problémák kockázatát, mivel az adatok nem különböző rendszerek között mozognak.

7. ábra
Adatok betöltése a staging környezetbe

```
import pandas as pd
def create_table(df,tb_name,source,engine):

# Tábla dinamikus létrehozása
    metadata = MetaData()

    if source is not None:

        df['source_id'] = source
        df['load_date_time'] = dt.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        df['load_status'] = 'Loaded'
        df['processing_flag'] = 0

    if not inspect(engine).has_table(tb_name):

        # DataFrame alapján tábla létrehozása, üres érték esetén az
         alapértelmezett típust használjuk
        columns = [Column(col, String) if dtype == 'object' else
                Column(col, Integer) if dtype == 'int64' else
                Column(col, Float) if dtype == 'float64' else
                Column(col, Date) if dtype == 'datetime64' else
                Column(col, String)  # alapértelmezett típus
                for col, dtype in zip(df.columns, df.dtypes)]

        table = Table(tb_name, metadata, *columns)

        metadata.create_all(engine)

    else:

        metadata.reflect(bind=engine)
        table = metadata.tables[tb_name]
        dele = table.delete()

        with engine.connect() as connection:
            result = connection.execute(dele)
            print(f"{result.rowcount} row(s) deleted")
            connection.commit()

    # Adatok átírása DataFrame-ról az adatbázisba
    df.fillna('').to_sql(tb_name, con=engine, if_exists='append', index=False)
```

Tábla dinamikus létrehozása: A szkript képes a pandas DataFrame alapján dinamikusan létrehozni az adattáblát az adatbázisban. Ehhez először lekérdezi, hogy a tábla létezik-e már. Ha nem, akkor a DataFrame oszlopainak nevei és adattípusai alapján oszlopdefiníciókat hoz létre, figyelembe véve az egyes adattípusoknak megfelelő SQL adattípusokat (pl. String, Integer, Float, Date).

Adatok előkészítése és betöltése: A szkript lehetővé teszi további adatmezők, mint például source_id, load_date_time, load_status, és processing_flag hozzáadását a DataFrame-hez,

mielőtt az adatokat az adatbázisba betölti. Ezek az oszlopok segítik az adatok nyomon követését, állapotának jelölését és a feldolgozás menedzselését. Ha a tábla már létezik, a szkript először törli az összes meglévő adatot a táblából, majd az új adatokat betölti, biztosítva ezzel az adatok frissességét és relevanciáját.

Adatbetöltés: Végül a DataFrame adatokat az to_sql metódus segítségével betöltjük az adatbázis táblájába. Ez a lépés támogatja az "append" módot, ami azt jelenti, hogy az új adatok hozzáadásra kerülnek a meglévő adatokhoz, amennyiben a tábla már létezik és nem történt adattörlés.

Ez a folyamat (7. ábra) különösen hasznos azokban az ETL folyamatokban, ahol a dinamikus táblakezelés és az adatok gyors frissítése kulcsfontosságú. A szkript moduláris kialakítása lehetővé teszi, hogy könnyen beilleszthető legyen különböző adatintegrációs és -feldolgozási folyamatokban.

4.2.5. Transform (Átalakítás)

A forrásrendszerekből származó adatok sikeres átmozgatása után a Staging adatbázisba, az ETL folyamat "Extract" szakaszát követően, megkezdhetjük az adatok tisztítási és transzformációs lépéseit. Ez a fázis kulcsfontosságú adataink minőségének biztosítása és a célrendszer számára szükséges formátumba való átalakítás szempontjából. A tisztítási folyamat magában foglalhatja a hiányzó vagy érvénytelen adatok kezelését, duplikátumok eltávolítását, és az adatok normalizálását. Ezt követően, az átalakítási lépés során az adatokat átstrukturáljuk, összevonjuk, és átszámítjuk, hogy azok megfeleljenek az adattárház követelményeinek. Ezek a lépések biztosítják, hogy az adatok készen álljanak a "Load" szakaszra, azaz az adattárházba történő betöltésre.

Ennek a szakasz alap kiinduló pontja a Chatbot rendszerből betöltött adatok tisztítása, ami az alábbiakat foglalja magában:

A folyamat egy kritikus lépése a Chatbot rendszerből származó adatok előkészítése, ami alapvető az adattisztítási feladatok foglal magában elsősorban. A folyamat során két fő tevékenységre helyeztem a hangsúlyt:

- **Felhasználók validálása a bérszámfejtési rendszer alapján**: Ez a lépés magában foglalja azon felhasználók azonosítását és kiszűrését, akik valós entitásokként szerepelnek a bérszámfejtési rendszerben. A bérszámfejtő rendszerben a felhasználok pontos azonosítására a PersID-t használjuk és ez a kulcs rendelkezésre áll a chatbot rendszerben is. Célunk itt az, hogy kizárjuk azokat a személyeket, akik már nem aktív munkavállalók például azokat, akiknek munkaviszonya megszűnt vagy akiket a rendszerből töröltek, valamint a teszt célból létrehozott felhasználói profilokat. Ez a lépés biztosítja, hogy az adattárházamban csak érvényes és releváns felhasználói adatok szerepeljenek, így növelve az adathalmaz pontosságát és megbízhatóságát.
- **Duplikátumok eltávolítása**: Az adattisztítási folyamat egy másik lényeges eleme a duplikációk felismerése és kiküszöbölése. Tekintettel arra, hogy egy személy több különböző nyelvű chatbot rendszerben is regisztrálhat, előfordulhat, hogy több felhasználói profilja is létrejön. Ennek megakadályozása érdekében, megkeresem a személy azonosító alapján (PersID) a duplikációkat, majd megnézem, hogy bevan-e jelentkezve valamelyik nyelvű chatbotba a felhasználó, ha igen akkor azt profilt tartom meg, ha nincs bejelentkezve akkor megnézem a személy állampolgárságát és az ezzel megegyező felhasználót tartom csak meg. A duplikációk eltávolításával csökkentem az adathalmaz redundanciáját és javítom annak integritását.

Ezek a lépések elengedhetetlenek ahhoz, hogy a későbbi elemzések során pontos és megbízható adatokra támaszkodhassak, ezzel maximalizálva a Chatbot rendszerünk által nyújtott értéket és hatékonyságot.

Az ETL folyamat Transform szakaszát SQL alapokon nyugvó megközelítéssel valósítottam meg az SQLAlchemy modul segítségével, mely magában foglalja komplex SQL lekérdezések, táblák és nézetek alkalmazását. Ez a módszer lehetővé tette az adatok strukturált átalakítását, normalizálását, miközben biztosítja az adatok integritását és minőségét. Az SQL lekérdezések precíz és hatékony manipulációt tesznek lehetővé, így az adatokat pontosan az adattárház specifikus követelményeinek megfelelően tudtam előkészíteni.

Az SQL nyelv segítségével történő adattranszformáció nem csak hogy növelte a folyamatok átláthatóságát és ellenőrizhetőségét, de jelentős mértékben javította az adatkezelési folyamatok hatékonyságát is. Az ilyen típusú megközelítés előnye, hogy közvetlenül az adatbáziskezelő rendszerben is elvégezhető, kihasználva annak teljesítményét és

megbízhatóságát, miközben minimalizálja a külső adatmanipulációs eszközök szükségességét. Az így megirt SQL kódokat (8. ábra) implementáltam a Python folyamataimban.

Példa a bejelentkezetett felhasználok validálására:

8. ábra
Bejelentkezetett felhasználok validálása

```
-- Nincs duplikáció
select *, 1 logged
from employee_employees
where id IN (select distinct employee_id from
employee_employees_platform_profiles)
and emplid in (
    select emplid
    from(
        select emplid, COUNT(emplid) AS count_of_emplid
        from employee_employees
        where ISNUMERIC(persid) = 1
        and emplid IN (select emplid from AbEmpl where cast(termdate as date) >
       cast(getdate() as date))
        and id IN (select distinct employee_id from
       employee_employees_platform_profiles)
        GROUP BY emplid
        HAVING COUNT(emplid) = 1
    ) dp
)

union

-- Duplikációk kezelése
    select id,persid,application_id,emplid,hiredate,taxcode,taj,source_id,
    load_date_time,load_status,processing_flag, 1 logged
    from (
    select ee.*,p.name,p.country, p.country2,
    (select application_id from AbLanguage where name = p.country) valid_app_id
    from [dbo].[employee_employees] ee, AbPerson p
    where ee.persid = p.persid
    and id IN (select distinct employee_id from
    employee_employees_platform_profiles)
    and ee.emplid in(
        select emplid
            from(
            select emplid, COUNT(emplid) AS count_of_emplid
            from employee_employees
            where ISNUMERIC(persid) = 1
            and emplid IN (select emplid from AbEmpl where cast(termdate as date)
>
           cast(getdate() as date))
            and id IN (select distinct employee_id from
           employee_employees_platform_profiles)
            GROUP BY emplid
            HAVING COUNT(emplid) > 1
        ) dp
    )
    ) valid_app
    where application_id = valid_app_id
```

4.2.6. Load (Betöltés)

Az adattárházam betöltési stratégiáját úgy alakítottam ki, hogy az optimálisan kezelje a különböző adatforrásokból származó információkat, ötvözve a teljes és az inkrementális adattöltési módszereket. Ennek keretében, ahol a chatbot rendszer nem nyújtott hozzáférést a historikus adatokhoz, ott az inkrementális adattöltési módszert alkalmaztam. Ez azt jelenti, hogy csak az új vagy módosult adatok kerültek átvezetésre, így biztosítva az adatok naprakészségét anélkül, hogy a teljes adatkészletet minden egyes alkalommal újra betölteném. Ezzel szemben, azokban az esetekben, ahol a forrásrendszer lehetővé tette az összes releváns historikus adat elérését, ott a teljes adattöltési módszert alkalmaztam. Ez a megközelítés azt jelenti, hogy minden egyes adattöltési ciklus során az adattárház teljes releváns adatkészletét frissítettem, függetlenül az adatok előző állapotától. Illetve van olyan kiegészítő adatokat tartalmazó dimenzióm táblám, mint a szállás adatok, amiket nem kell historikusan tárolnom, így ebben az esetben is a teljes adatbetöltési logikát alkalmazom.

Ez a kettős megközelítés lehetővé tette számomra, hogy rugalmasan kezeljem az adattöltési folyamatokat, maximális hatékonyságot és adatminőséget biztosítva. Az inkrementális adattöltés alkalmazása kritikus fontosságú volt a rendszer teljesítményének optimalizálása és a historikus adatok kezelésében az adatfrissítések gyorsaságának és megőrzésének biztosítása szempontjából, míg a teljes adattöltés biztosította az egyszerűséget és adatok teljességét és integritását az adattárházban.

9. ábra
Teljes adatbetöltés

```
def load_dim_accommodation():

    dim_accommodation = '''SELECT persid, szallas accommodation
                    FROM [ChatbotStaging].[dbo].[Dim_Stg_Accommodation]
                    where persid in (select persid from Dim_Person)'''


    delete_to_sql('Dim_Accommodation')
    dim_accommodation_df = select_sql(dim_accommodation)
    dim_accommodation_df.to_sql(
'Dim_Accommodation', mssql_prod_engine(),
if_exists='append', index=False
)
```

A (9. ábra) delete_to_sql függvény segítségével törlöm a dimenzió tábla adatait majd to_sql függvénnyel betöltöm az új adatokat.

10. ábra
Inkrementális adatbetöltés

```
def incermental_load(source,target,key,tb_name):

    load_date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    if 'load_date_time' in source.columns:
        source = source.drop(columns=['load_date_time'])

    if 'load_date_time' in target.columns:
        target = target.drop(columns=['load_date_time'])

    # 1. Változások észlelése
    changes = source[~source.apply(tuple,1).isin(target.apply(tuple,1))]

    # Új rekordok azonosítása
    inserts = changes[~changes[key].isin(target[key])]
    inserts['load_date_time'] = load_date

    # 3. Módosult sorok azonosítása
    modified = changes[changes[key].isin(target[key])]
    modified['load_date_time'] = load_date

    if modified.empty == False:
        update_to_sql(modified, tb_name, key,mssql_staging_engine())
    else:
        print('modified is empty')

    if inserts.empty == False:
        inserts.to_sql(tb_name, mssql_prod_engine(), if_exists='append',
         index=False)
    else:
        print('inserts is empty')
```

A függvény célja (10. ábra), hogy csak az új vagy módosult sorokat töltse át a céladatbázisba, optimalizálva ezzel az adattöltési folyamatot. Az alábbiakban részletezem a függvény működését lépésről lépésre:

Előkészítés: A függvény először frissíti a load_date változót a jelenlegi dátumra és időre. Ezt követően eltávolítja a load_date_time oszlopot mind a forrás-, mind a cél adatkészletekből, amennyiben ezek az oszlopok léteznek. Ez biztosítja, hogy az adatok összehasonlítása ne az időbélyeg alapján történjen.

Változások észlelése: A függvény meghatározza azokat a sorokat, amelyek megtalálhatóak a forrásban, de nincsenek jelen a cél adatkészletben. Ez a lépés azonosítja az új vagy módosult sorokat.

Új rekordok azonosítása: Ebben a lépésben a függvény kiválasztja azokat a sorokat a változások közül, amelyek kulcsa (key) nem szerepel a cél adatkészletben. Ezeket az új rekordokat készíti elő a beszúrásra (inserts), hozzáadva a jelenlegi load_date_time értéket.

Módosult sorok azonosítása: A függvény kiszűri azokat a sorokat a változások közül, amelyek kulcsa megegyezik a cél adatkészlet valamelyik kulcsával. Ezek a sorok tekinthetőek módosultnak (modified), mivel a kulcs megegyezik, de egyéb adatok eltérhetnek. A módosult sorokhoz hozzáadásra kerül a load_date_time.

Módosult sorok frissítése: Ha vannak módosult sorok, a függvény frissíti őket a cél adatbázisban az update_to_sql függvény segítségével.

Új sorok beszúrása: Ha vannak új sorok, a függvény beszúrja őket a cél adatbázisba az inserts.to_sql metódussal.

4.2.7. Adatminőség-ellenőrzés (data quality check)

Az adattárház betöltését követően kulcsfontosságú lépés az adatminőség-ellenőrzés (data quality check) elvégzése, amely során alaposan megvizsgálom az adattárházban tárolt adatokat annak érdekében, hogy azonosítsam azokat az információkat, amelyek már nem tükrözik pontosan a forrásrendszerek jelenlegi állapotát. Az (4.2.3) szakaszban leírt ETL folyamat Extract fázisában alkalmazott előfeldolgozási és tisztítási technikák - mint az adatok előszűrése és a releváns oszlopok kiválasztása - alapvető előkészítő lépéseket jelentenek az adatminőség biztosítása felé vezető úton. Különös figyelmet fordítok a bérszámfejtési rendszerben törölt jogviszonyokra és személyekre, mivel ezek az adatok idővel elavulhatnak, és ezzel pontatlanságokat okozhatnak az adattárházban. Az adatminőség-ellenőrzés folyamata biztosítja, hogy az adattárházban tárolt adatok naprakészek, pontosak és relevánsak maradjanak, így hozzájárulva az adatvezérelt döntéshozatal megbízhatóságának növeléséhez.

Adattárház mentés:

Az adattárházam kezelése során további lépéseket tettem az adatminőség megőrzése és az integritás biztosítása érdekében. Ebből a célból, valamint az adatbetöltési folyamatok során előforduló esetleges hibák minimalizálása érdekében, minden egyes adatbetöltést egy teljes adattárház-mentést csinálok. Ez a megközelítés biztosítja, hogy bármely kritikus hiba vagy

adatintegritási probléma esetén gyorsan és hatékonyan tudjam visszaállítani az adattárházat a legutóbbi stabil állapotára. Az ilyen típusú mentéseket rendszeresen, visszamenőleg egy hetes távlatban tárolom, ami lehetővé teszi számomra, hogy rugalmasan kezeljem az adattárház állapotának helyreállítását.

Ez a megközelítés számos előnnyel jár:

- Adatintegritás védelme: Az adatbetöltési folyamatok során bekövetkező bármilyen hiba azonnali korrekcióját teszi lehetővé, csökkentve az adatvesztés vagy sérülés kockázatát.
- Biztonsági háló: A rendszeres és előre meghatározott időközönkénti mentések egyfajta biztonsági hálót képeznek, amely biztosítja az adattárház folytonosságát és stabil működését.
- Gyors helyreállítási képesség: Kritikus hibák esetén a rendszer gyors visszaállítása a legutóbbi stabil állapotra minimálisra csökkenti a leállás időt és az ezzel járó üzleti következményeket.
- Adatminőség ellenőrzés: Az adatbetöltések előtti mentések lehetőséget biztosítanak az adatminőség folyamatos ellenőrzésére és az adatok pontosságának fenntartására.

Összességében ez a proaktív megközelítés növeli az adattárházak megbízhatóságát és teljesítményét, miközben minimalizálja az adatvesztés vagy -sérülés kockázatát, így hozzájárulva az adatvezérelt döntéshozatali folyamatok hatékonyságához és pontosságához.

4.3. Automatizált adatbetöltés

A Dagster egy kiváló keretrendszer adatfolyamatok, különösen ETL (Extract, Transform, Load) munkafolyamatok tervezésére és végrehajtására.

Dagster alapvető elemei:

- **Op (Operátor):** Egy műveleti egység, amely specifikus feladatokat hajt végre. Az op-ok modularitást biztosítanak, lehetővé téve az egyes munkafolyamat-lépések izolálását és tesztelését.

- **Asset:** Az asset-ek explicit módon definiálják az adatfolyamatokban szereplő adatokat és azok függőségeit. Ezek segítségével strukturálhatóak az adattranszformációk és az adatfolyamatok közötti kapcsolatok.
- **Job:** Összefogja az op-okat és asset-eket, meghatározva a függőségeket és az adatfolyamot közöttük. Egy job reprezentálja az ETL folyamat teljes körét.
- **Schedule:** Az ütemezők segítségével az ETL folyamatokat előre meghatározott időpontokban automatikusan végrehajthatjuk.

Adatfolyamat Definíciója:

Az ETL folyamat Extract szakaszában op-okat definiáltam (11. ábra) az adatbázisokból történő adatok kinyerésére. A Transform szakaszban az adatok átalakítására szolgáló op-okat hoztam létre, míg a Load szakaszban az adatok céltáblákba történő betöltésére szolgáló op-okat definiáltam.

11. ábra
OP (Operátor)

```
@op
def extract_employee_op():
return employee()
```

Az op-ok kimeneteit felhasználva asset-eket definiáltam (12. ábra) az adatfolyamatok közötti függőségek és az adatok áramlásának explicit meghatározására:

12. ábra
Asset

```
@asset(group_name="Extract")
def extract_employee(context: AssetExecutionContext):
    employee_data = extract_employee_op()
    create_table(employee_data, 'employee_employees', 'ChatbotPG')
```

Végül, létrehoztam egy job-ot (13. ábra), amely összefűzi az op-okat és asset-eket, ezzel definiálva az ETL folyamat teljes körét:

13. ábra
Job

```
@job
def etl_pipeline():
    # Hívja meg az extract `op`-okat
    bereloleg_data = extract_bereloleg_op()
    potszabi_data = extract_potszabi_op()
    employee_data = extract_employee_op()
    # További `op`-ok és `asset`-ek hívása az adatfolyamathoz...
    transform_valid_employees_platform_op()
    valid_employee_op()
    load_dim_person_op()
```

Az ETL folyamatok ütemezése (14. ábra) a Dagster segítségével történik, amely lehetővé teszi azok automatikus végrehajtását előre meghatározott időpontokban:

14. ábra
Schedule

```
@schedule(cron_schedule="0 23 * * *", job=etl_pipeline,
execution_timezone="UTC")
def daily_etl_schedule():
    """
    Ez az ütemezés minden nap este 23:00-kor (UTC idő szerint) indítja el
     az ETL folyamatot.
    """
    return {}
```

Ezáltal az ETL folyamat minden nap este 23:00-kor automatikusan végrehajtásra kerül, ami a cron kifejezés segítségével van meghatározva.

A fentiekben bemutatott lépések biztosítják az ETL folyamatok hatékony szervezését és végrehajtását, javítva az adatkezelés átláthatóságát és megbízhatóságát.

4.4. Power Bi riport

A projekt végső célja egy Power BI riport létrehozása volt, amely nem csupán az adatok megjelenítését teszi lehetővé, hanem interaktív, könnyen értelmezhető és mélyreható betekintést is nyújt a WHC Kft. döntéshozói számára. A Power BI mint eszköz kiválasztása nem véletlen: kiváló adatintegrációs, -feldolgozási és -vizualizációs képességeivel kiemelkedik az analitikai és BI (Business Intelligence) eszközök között, így rendkívül hatékony megoldást nyújt

a komplex adatok értelmezésére és prezentálására, ezt az (2.6.1) fejezetben részletesen kifejtettem.

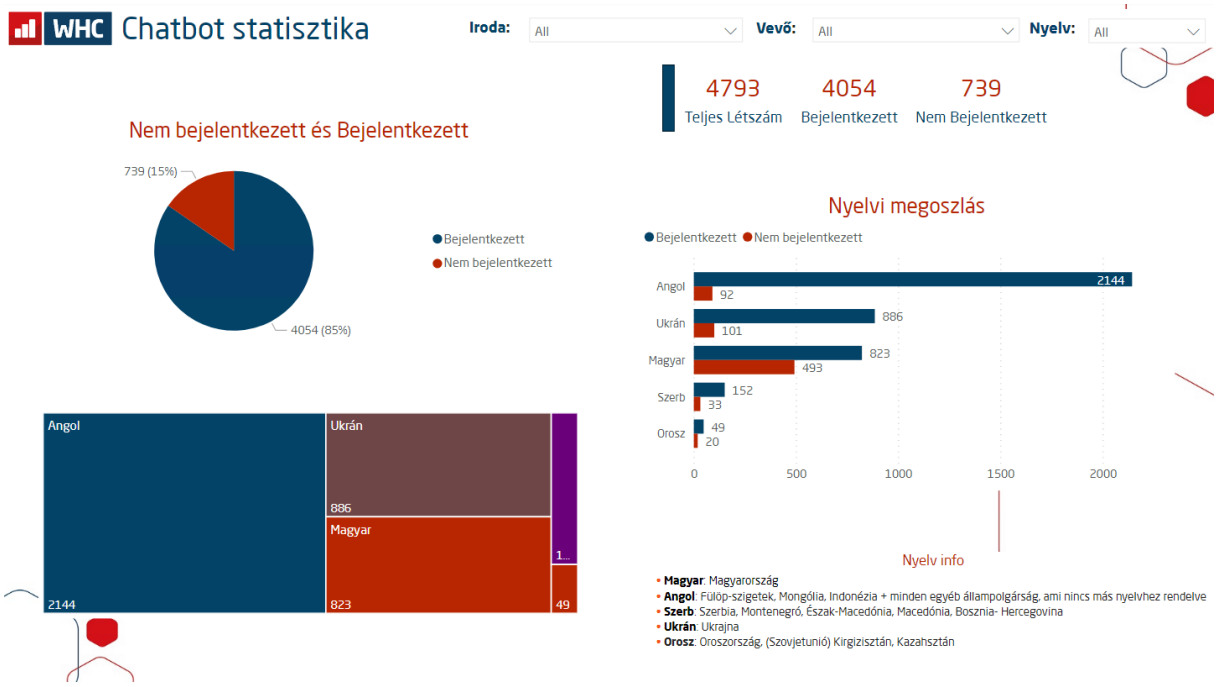
4.4.1. Riport tervezése

A riportot a közép, illetve a felső vezetés használja, a riport tervezésekor különös figyelmet fordítottam arra, hogy az információk struktúrája és az interaktív elemek olyan módon kerüljenek kialakításra, hogy azok maximálisan támogassák a különböző vezetői szintek igényeit és döntéshozatali folyamatait. Az alábbi szempontok kiemelt figyelmet kaptak:

- **Egyszerűség és átláthatóság:** A komplex adatok egyszerű és átlátható módon való prezentálása, hogy a vezetők ne vesszenek el az adatok tengerében, és könnyen megtalálják a számukra fontos információkat.
- **Interaktivitás:** A riport interaktív elemei lehetővé teszik a vezetők számára, hogy mélyebben beleássanak az adatokba, felfedezzék az összefüggéseket, és testre szabhassák a nézeteket saját szükségleteik szerint.
- **Stratégiai fókusz:** A riport kialakításakor külön figyelmet fordítottam arra, hogy a bemutatott adatok és elemzések támogassák a stratégiai döntéshozatalt, kiemelve a vállalati célok eléréséhez kulcsfontosságú metrikákat és trendeket.
- **Időbeli elemzések:** A trendek és időbeli változások vizualizációja lehetővé teszi a vezetés számára, hogy nyomon követhesse a vállalat teljesítményének fejlődését, és azonnal reagálhasson az esetleges változásokra.

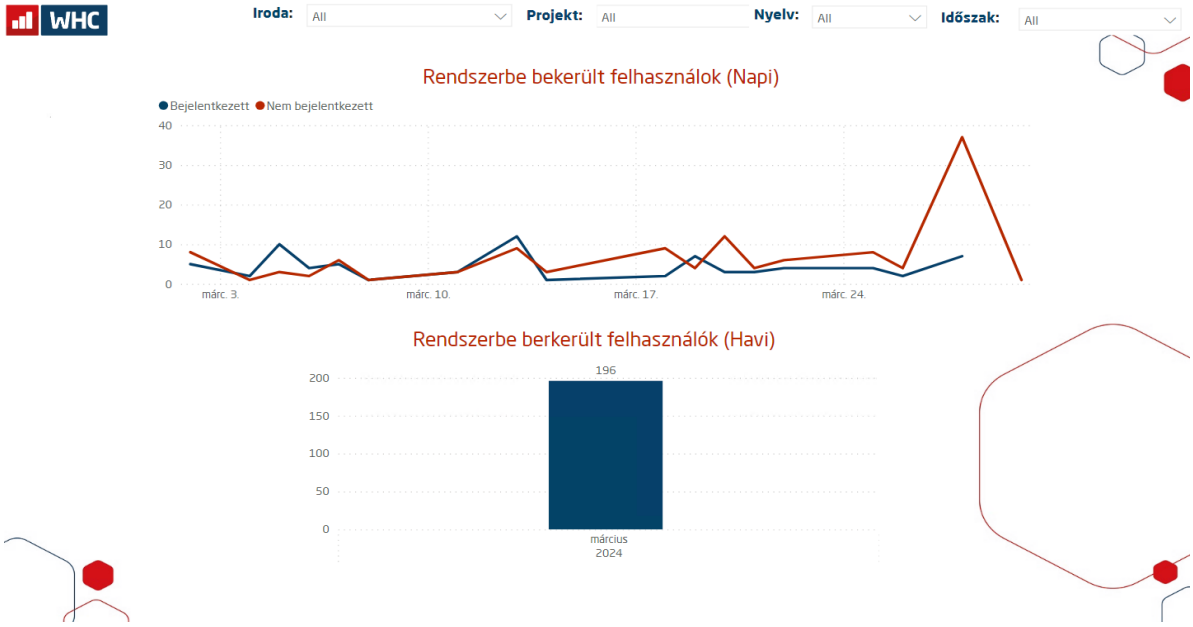
4.4.2. Elkészült riport

15. ábra
Power BI – Dashboard



A fenti (15. ábra) látható a riportom kezdő lapja (Dashboard), amivel a vezetők egy pillantással áttekinthetik az itt találhatók a legfontosabb információk.

16. ábra
Power BI - Historikus adatok



A fenti (16. ábra) láthatóak azok az historikus adatok, ami csak az adattárház létrehozásával álhatott elő, nap és havi bontásban pontosan látható, hogy ki mikor került be a chatbot rendszerbe és mikor jelentkeztek be.

5. Összegzés

A WHC Kft. egy kelet-közép-európai régióban működő jelentős HR szolgáltató, amely a munkaerő-kölcsönzés, közvetítés és bérszámfejtés területén nyújt széleskörű szolgáltatásokat. A vállalat a kommunikációs és kapcsolattartási folyamatok egyszerűsítésére egy többnyelvű Chatbot rendszert alkalmaz, amelyet külső szolgáltató fejleszt és üzemeltet. Azonban a rendszer használatával kapcsolatos részletes statisztikák és a historikus adattárolás képességének hiánya komoly kihívást jelent a vállalat számára.

Az adattárházak fontosságát az adatvezérelt döntéshozatal támogatásában kiemelve, a dolgozat bemutatja, hogy az adattárházak hogyan képesek nagy mennyiségű adat hatékony kezelésére, a vállalatok gyors reagálására a változó piaci környezetre, valamint a jogi és szabályozási követelmények teljesítésére. Az adattárházak segítségével a vállalatok jobban megérthetik ügyfeleiket, hatékonyabban tudnak versenyezni a piacon, és javíthatják az üzleti folyamataikat.

A szakdolgozat a WHC Kft. szolgáltatásainak bemutatása mellett részletesen ismerteti az adattárház projektet, beleértve a célkitűzéseket, a forrásrendszereket és az adattárház felépítését.

A szakdolgozat célja egy olyan adattárház létrehozása volt, amely integrálja és centralizálja a WHC Kft. Chatbot rendszeréből és Abacus bérszámfejtő rendszeréből származó adatokat részletes statisztikák elérése, a historikus adatok megőrzése és az integrált adatelemzés lehetőségének megteremtése érdekében.

A projekt során kifejlesztett adattárház a Python nyelven írt ETL (Extract, Transform, Load) eljárások segítségével teszi lehetővé az adatintegrációt. Az adatok két különböző forrásból történő összegyűjtését és egységesítését követően, a Dagster adatfeldolgozási keretrendszer segítségével az adatok ütemezetten kerültek betöltésre egy SQL Server adatbáziskezelő rendszerbe. Az így létrejött adattárház lehetőséget nyújt részletes statisztikák készítésére, a historikus adatok hosszú távú megőrzésére és az integrált adatelemzésre. A Power BI alkalmazásával készített dashboard szemléletesen támogatják a döntéshozókat, így elősegíti a vállalati hatékonyság növelését.

Irodalomjegyzék

ActiveBatch - ETL Automation with Python. (2024. március 14). Forrás: ActiveBatch: https://www.advsyscon.com/blog/etl-automation-with-python/

Apache Kafka. (2024. március 17.). Forrás: Apache Kafka: https://kafka.apache.org/intro

Apache Nifi Documentation. (2024. március 17.). Forrás: Apache Nifi Documentation: https://nifi.apache.org/documentation/v2/

Chatbossteam. (2024. március 4.). Forrás: Chatbossteam: https://www.chatbossteam.com/

Dagster Docs. (2024. március 19.). Forrás: Dagster: https://docs.dagster.io/getting-started

Datatheta Getting Started with Pentaho Data Integration. (2024. március 19.). Forrás: Datatheta: //www.datatheta.com/getting-started-with-pentaho-data-integration/

ETL vs ELT – understanding the key differences. (2024. március 11). Forrás: Softweb Solution: https://www.softwebsolutions.com/resources/key-differences-etl-vs-elt.html

GeeksforGeeks - History of Python. (2024. március 17.). Forrás: GeeksforGeeks: https://www.geeksforgeeks.org/history-of-python/

Google Dataflow. (2024. március 10.). Forrás: Google Dataflow: https://cloud.google.com/dataflow

How Data Modeling and ETL designs are important for designing a Data Warehouse? (2024. március 11.). Forrás: Medium: https://avisri.medium.com/how-etl-designs-and-data-modeling-are-important-for-designing-a-data-warehouse-2b3cc3514d0e

Inmon, W. (2005). *Building the Data Warehouse, Fourth Edition.* Kanada: Wiley Publishing, Inc.

Kimball, R. (1998). *The Data Warehouse Lifecycle Toolkit.* Kanada: John Wiley & Sons.

Kimball, R., & Ross, M. (2002). *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling.* Kanada: John Wiley and Sons, Inc.

Microsoft SQL documentation. (2024. március 12.). Forrás: Microsoft SQL documentation: https://learn.microsoft.com/hu-hu/sql/?view=sql-server-ver16

Pandas Docs. (2024. március 16.). Forrás: Pandas: https://pandas.pydata.org/

Power BI. (2024. március 20.). Forrás: Power BI: https://www.microsoft.com/en-us/power-platform/products/power-bi

Python. (2024. március 17.). Forrás: Python: https://www.python.org/doc/essays/blurb/

Sarjen SSIS Advantages-Disadvantages. (2024. március 19.). Forrás: Sarjen: https://www.sarjen.com/ssis-advantages-disadvantages/

Sidló, C. (2004). *Adattárház összefoglaló*. Forrás: Óbudai Egyetem: https://tig.kgk.uni-obuda.hu/vir/anyag/Sidlo_adattarhazak.html

SQLAlchemy Docs. (2024. március 20.). Forrás: SQLAlchemy: https://www.sqlalchemy.org/

Start Data Engineering Apache Airflow. (2024. március 17.). Forrás: Start Data Engineering: https://www.startdataengineering.com/post/apache-airflow-review-the-good-the-bad/

Stich - Using Python for ETL. (2024. március 18.). Forrás: Stich: https://www.stitchdata.com/resources/python-etl/

Tableau. (2024. március 21.). Forrás: Tableau: https://www.tableau.com/

VT-Soft. (2024. március 4.). Forrás: VT-Soft: https://vtsoft.hu/

WHC. (2024. március 3.). Forrás: WHC: https://whc.whc.hu/cegeknek

Ábrajegyzék

Táblajegyzék

NYILATKOZAT

a szakdolgozat nyilvános hozzáféréséről és eredetiségéről

A hallgató neve:	Szakács Kristóf
A Hallgató Neptun kódja:	GLYDC0
A dolgozat címe:	HR Adattárház Fejlesztése Chatbot és Bérszámfejtő Rendszer Integrációjával
A megjelenés éve:	2024
A konzulens intézetének neve:	Műszaki Intézet
A konzulens tanszékének a neve:	Mérnökinformatikai Tanszék

Kijelentem, hogy az általam benyújtott szakdolgozat egyéni, eredeti jellegű, saját szellemi alkotásom. Azon részeket, melyeket más szerzők munkájából vettem át, egyértelműen megjelöltem, és az irodalomjegyzékben szerepeltettem.

Ha a fenti nyilatkozattal valótlant állítottam, tudomásul veszem, hogy a záróvizsga-bizottság a záróvizsgából kizár és a záróvizsgát csak új dolgozat készítése után tehetek.

A leadott dolgozat, mely PDF dokumentum, szerkesztését nem, megtekintését és nyomtatását engedélyezem.

Tudomásul veszem, hogy az általam készített dolgozatra, mint szellemi alkotás felhasználására, hasznosítására a Magyar Agrár- és Élettudományi Egyetem mindenkori szellemitulajdon-kezelési szabályzatában megfogalmazottak érvényesek.

Tudomásul veszem, hogy dolgozatom elektronikus változata feltöltésre kerül a Magyar Agrár- és Élettudományi Egyetem MATER Hallgatói Dolgozatok repozitoriumába. Tudomásul veszem, hogy a megvédett és

- nem titkosított dolgozat a védést követően
- titkosításra engedélyezett dolgozat a benyújtásától számított 5 év eltelte után

nyilvánosan elérhető és kereshető lesz az Egyetem MATER Hallgatói Dolgozatok repozitoriumában.

Kelt: Győr, 2024. április. 19.

Szakács Kristóf

Hallgató aláírása

NYILATKOZAT

Szakács Kristóf (név) (hallgató Neptun azonosítója: GLYDC0) konzulenseként nyilatkozom arról, hogy a szakdolgozatot áttekintettem, a hallgatót az irodalmi források korrekt kezelésének követelményeiről, jogi és etikai szabályairól tájékoztattam.

A záródolgozatot/szakdolgozatot/diplomadolgozatot/portfóliót a záróvizsgán történő védésre **javaslom / nem javaslom**[1].

A dolgozat állam- vagy szolgálati titkot tartalmaz: igen nem

Kelt: Gödöllő, 2024. március. 19.

belső konzulens

[1] A megfelelő aláhúzandó.