Nyári gyakorlat beszámoló

Farkas Péter

2019 nyár

Tartalomjegyzék

Első két hét

Kezdetek (első hét hétfő - ~csütörtök)

Az első hét első napjai főként az előzőlegesen e-mail-ben kiküldött segédanyagok olvasgatásából állt. Bár elérhető volt már a trial licenszes SAP platform, erőforrások tekintetében a teremnyi használat önmagában közel használhatatlanná tette (ezen ügyben kaptunk egy saját szerverre ígéretet), emellett feladat híján nem sok ötletem volt, mit lehetne próbálgatni benne, így viszonylag hamar háttérbe szorult. Személy szerint engem főként a PAL és az adatbányászat érdekelt, azonban mivel nem tűnt úgy, hogy ezeket gyorsan módomban állna kipróbálni, és konkrét feladat se volt még, az elmélyedés helyett inkább több téren próbáltam felületesebb ismereteket szerezni, és az adattárházak fogalmaival/nodeJS-sel ismerkedtem, utóbbiból létrehoztam egy lokális projektet és – az előzőleges JS ismeretek alapján – próbáltam megérteni a működését.

Adatbázis (első hét ~csütörtök – második hét kedd)

Amikor végül megkaptuk a társaságok bejegyzéseit, és fel lett vázolva nagy vonalakban a feladat, először az adatok importálását és normalizálását vettem célul. Rövid tanulmányozás után létrehoztam egy relációsémát, amire szerintem a .csv-ben tárolt adatok illeszkedni fognak. Ezen a ponton sokaknál (köztük nálam is) akadtak nehézségek, ugyanis az ingyenes HANA SAP nehezen bírta a teremnyi hallgató build request-jeit, emellett problémát okoztak még a nem egyedi sémaelnevezések, illetve a legújabb database modul verzióval se tudtuk a sémát buildelni, és akiknek valahogyan mégis sikerült, a csv adat import nem volt elérhető. Ezek miatt nem túl sok eredménnyel telt el a csütörtök nagy része. A nehézségek miatt az eredeti csv-ből inkább egy json exportot csináltam, amit utána JS-sel dolgoztam volna fel, azonban ekörül vége lett az aznapi időnek.

Pénteken a hamarosan elérhető saját szerver ígérete miatt inkább az akkortájt 2. adag tutorial anyagot olvasgattam, mert ez a további haladás feltételének tűnt, azonban végül a nap végéig a szerver nem lett elérhető, úgyhogy sok produktuma nem volt a napnak.

Az elkövetkező héten hétfőn már elkészült a lokális szerver, így megtörténhetett a séma létrehozása és az adatok importálása. Ehhez több körben újra kellett csinálni a sémát, mert egyes társaságok nevei 500 karakternél is hosszabbak voltak. A mezőhosszok igazítása után is maradt egy sor, amely hibás volt, így az simán nem került felvitelre. Ezek után elkezdtem megtervezni, hogy hogyan kéne a táblákat szétszedni. A legelső, triviális felbontásban csak a név-kód mező párok lettek volna kivéve, az eredeti táblában csak a kódokat meghagyva, azonban az adatok tanulmányozása után úgy gondoltam, hogy az ország-régió-megye-település és nemgazd_ag-nemgazd_agazat-nemgazd_szakagazat mezőket hierarchikusan egymásra lehet rakni. Ekkor megkezdtem az alap adathalmazból az kód-név mezők külön táblákba exportálását `CREATE TABLE TBL AS SELECT DISTINCT `KÓD`, `NÉV` FROM `TÁBLA` ORDER BY `KÓD`; ALTER TABLE TBL ADD PRIMARY KEY(`KÓD`);` struktúrájú lekérdezésekkel. A

legtöbbnél a kulcsok kiosztásakor hibába botlottam, mert több esetben is egy kódhoz több név is tartozott (rendszerint 1 karakter elgépeléséből). Ezeket `UPDATE` lekérdezésekkel kiküszöböltem, így az egyszerűbb kód-név táblákat sikerült kiemelni. Az előbb említett általam hierarchikusnak gondolt mezőkhöz érve az elképzelés az volt, hogy a legfelső szint (ország/nemgazd_ag) csak egy kód-név párból áll, kód kulccsal, míg az alatta levők kód-név-eggyel_fentebbi_szint_kód hármasokból így az eredeti táblában a legalsó szint kulcsából el lehetne jutni a legfelső szintig. Itt problémába ütköztem, ugyanis az ország-régió-megye-település tábláknál a régiók és megyék között is szerepelt 'egyéb', amely egyebek viszont több országhoz is tartoztak, így, amely települések több országhoz/régióhoz is tartoztak, nem voltak egyértelműen országig felvezethetőek csak az egyel feljebbi szint kódjának hozzáadásával. Ekkor gondolkodtam, hogy újratervezem a hierarchiát, esetleg egy 'HELY' tábla bevezetésével, amely ország-régió-megye-település kódnégyesekből állna, aminek a település már valid kulcsa, azonban ebben a megoldásban sem voltam teljesen biztos, hogy helyes lenne, és inkább elnapoltam a normalizálást, és a Bing Maps API használatát próbálgattam egy egyszerű html oldallal, amin a táblából szúrópróba-szerűen vett címekkel tesztelgettem. A címekből az API geokódoló paraméterezhető URL-jének query-ként megadva először long-lat koordinátákat szerezem, majd utána egy térképen pushpineket helyeztem el az így megszerzett pozíciókra. Ezek mellett a nap folyamán Béleczki tanár úr útmutatása mellett a szerveren levő homokozón létre lett hozva egy odata service, egyelőre csak a nyers adatokat közvetítve.

Kedden visszatértem a normalizáláshoz. Kezdetnek kaptam János csoporttársamtól egy valamennyi duplikátumtól/hibás bejegyzéstől mentesített .JSON file-t, ezt importáltam, majd a hierarchiát egyelőre elhagyva simán kód-név táblákra szedtem szét az adatokat a már korábban említett struktúrájú sql utasításokkal. A visszamaradt fő kapcsolótáblából első körben a teljesen megegyező sorokat küszöböltem ki egy `SELECT DISTINCT *…..` lekérdezéssel, kulcsának végül összetetten a `TARS_HOSZ_NEV`, `TARS_ROV_NEV`, `ASZ_EVE` mezőket adtam meg, ez alapján kezdetben ~16 megegyező kulcsú pár volt, ezeket egyesével, célzott lekérdezésekkel küszöböltem ki (a probléma az összesnél az volt, hogy a címeknél a budapesti irányítószám mellett még római számmal a kerület is meg volt jelenítve, a kerületet is tartalmazó címekhez tartozó sorokat töröltem). Ezen kívül még korábbi lekérdezések során szemet szúrt, hogy pár Budapesti teljes címmel rendelkező társaságnál településnek nem Budapest szerepel, ezeket a sorokat szintén töröltem, bár ez nem volt túl következetes részemről, mert más településekkel hasonló jellegű problémákat nem kerestem. A gyakorlatra kijelölt idő lejártával a nap hátralevő részében internetes segédanyagok felhasználásával próbáltam az általam megfelelőnek tartott táblákból egy calculation view-t készíteni. Kezdetben hibákba ütköztem, ugyanis az importált táblákat a rendszer build közben nem találta meg, mert a database artifact-ba ezek sémái nem lettek felvéve, hanem csak lekérdezésekkel lettek létrehozva. Minden relációhoz létrehoztam egy-egy entity-t, megadtam az egymáshoz való kapcsolatukat, a saját és foreign kulcsokat, majd az így kapott táblákba átmásoltam a korábbiak tartalmát. Így már az alábbi (https://data-

flair.training/blogs/calculation-view-in-sap-hana/) tutorial alapján sikeresen összeraktam a calculation view-t, majd utána odata-ként elérhetővé tettem.

A hét hátralevő részében nem tudtam megjelenni a gyakorlaton, és a feladattal sem tudtam haladni.

Frontend (harmadik hét hétfő - kedd)

Egy olyan felületet terveztem elkészíteni, ami táblázatként pár oszlopnyi adatot ad meg soronként, illetve az adott sorhoz tartozik még egy gomb, amivel egy térképre a társasághoz tartozó címre elhelyez egy pint. Ehhez először rendelkezésemre állt még előző hét hétfőről a Bing Maps API tesztelős html oldal. Ebbe akartam importálni egy XMLHttpRequest-tel az odata-ból elérhető JSON-t, de ekkor problémába ütköztem, ugyanis nem volt az elérhető más hostról (CORS request denied). Pár órányi keresés után se sikerült ezt megoldani, úgyhogy a már adat JSON forrásként megnevezett Jánoshoz fordultam, mivel előző hét kedden ő nodeJS-ből küzdött hasonló problémával. Neki volt egy megoldása nodeJS-re és Angular keretrendszerre is, előbbi (saját felfogásom szerint) biztonsági funkciók kikapcsolásával, utóbbi proxy-val. Végül az utóbbi mellett döntöttem. E-mail-ben keringett egy kiindulási alapnak megfelelő mintaprojekt, ezt próbáltam először működésre bírni, azonban értés nélkül kaotikus volt, indításra a lokális szerver hibákat dobott, és végül inkább az Angular dokumentációjának Getting Started részével nulláról kezdtem egy saját projektet. Így, hogy valamennyi fogalmam már volt a keretrendszer struktúrájáról beimportáltam a korábbi mintaprojektből az odata service-t és a proxy-t. Lekérdeztem az első 100 bejegyzést, mindegyik után generáltam egy táblázatbeli sort, bennük a TARS_ROV_NEV, CÍM_EGYBEN, ASZ_EVE sorokat megjelenítve, illetve soronként egy extra gombbal egy térképre való pin hozzáadáshoz.

Kedden miután ezek működtek, próbáltam implementálni egy Bing Map-et. Ezzel a vártnál sokkal több nehézség volt, ugyanis míg html felületen ez kb 2 script és a hozzájuk tartozó div elhelyezéséből áll, Angularban fogalmam se volt, hogy mit hol kell elhelyezni, hogy utólag a mapról legyen referenciám a későbbi pinek felhelyezéséhez. Végül találtam egy megoldást, ami ígéretesnek tűnt, implementálva meg is jelenítette a térképet, egy furcsa viselkedést váltott ki. Először is, az oldal első indítására build failed volt, az oldal nem jelent meg, azonban (az `ng serve` paranccsal futtatott szerver érzékeli a forrásfájlok változásait, ekkor újrabuildeli a projektet) mindenféle változtatás nélkül (pl. szóköz hozzáírás majd törlés, mentés) minden továbbira a hibaüzenetek megmaradtak, azonban az oldal megjelent, és látszólag rendesen működött is, erre a jelenségre nem sikerült magyarázatot találni. Odáig nem jutottam el, hogy megpróbáljak pint elhelyezni rá, nem akartam a látszólag hibás megoldásra építeni, ezért inkább a Bing Maps API-ból elérhető statikus, képként megjelenített térképre hagyatkoztam, ami scriptek helyett csak egy <img> tag src attribútuma. Létrehoztam GeocoderService néven egy service-t, ami az egyik függvényével query-ből long-lat koordinátákat állít elő, illetve a másikkal long-lat koordinátákból a koordináták középpontú, ott egy pinnel rendelkező API által előállított kép url-jét adja vissza. Ekkor abba a hibába ütköztem, hogy a long-lat koordinátás résznél a HttpRequest eredménye undefined, de csak első híváskor. Ezt

végül a függvény aszinkron megvalósításával sikerült kiküszöbölni (előbb próbáltunk olvasni az objektumból, semminthogy az betöltődött volna, ami az itt csak egyszer, init-en lekért odata-nál nem volt problémás, de a gombnyomásra lekért adat első hívásánál igen). Végül az 'Ismeretlen' egybecímmel rendelkező társaságok miatt bekerült egy ellenőrzés, hogy kezeljük az eseteket, amikor nem kapunk vissza koordinátákat.

Folytatás

2019. 07. 18. Csütörtök

Mivel az adatmodellezés részre jelentkeztem, és konkrét feladat nem volt kiosztva, a társaságok adatait tartalmazó adatbázisokhoz tértem vissza. Eddig adattisztítást nem igazán végeztem, inkább a relációsémákat igyekeztem logikusabban/hatékonyabban kialakítani. Először is a sima név-kód pár táblák egy része helyett az ország-régió-megye-települést és nemgazd_ag-nemgazd_agazat-nemgazd_szakagazat részeket külön vettem egy HELY és egy NEMGAZD táblába, aminek a legalsó szint (település, szakágazat) lett a kulcsa, utána a fő táblába is csak ez marad benne. Ebből frissítettem az eddigi kalkulációs nézetet.

Következőnek a táblához akartam szervezni a címekhez tartozó long-lat koordinátákat. Ehhez külön táblába szedtem a címeket (ez egyelőre kulcs), mellé felvéve egy LONG és egy LAT default null mezőt. A sémák alakítása közben feltűnt, hogy a korábbi NEMGAZD és HELY táblákban nem specifikáltam, hogy a főtábla ezekhez való kötésekor az előbbieknek melyik mezőjét vegye kulcsnak. Ezt megadtam, viszont ez elrontotta az eddigi kalkulációs nézeteket (azt is, amelyikben a módosított tábla egyáltalán nem is szerepelt). Ez után ezeknek a helyreállításán dolgoztam, majd ezzel végezve visszatértem a címek kiszervezéséhez. Az elképzelés az volt, hogy a címek long-lat átalakításával az egyébként elég inkonzisztens formátumú címek szűrése is megkönnyebbül, ugyanis visszafelé a megegyező koordinátás címek már feltételezhetően egyszerűen rezolválhatóak. Az egyedi címeket külön táblába véve a következő feladat ezek long-lat koordinátákká kódolása, odata-n keresztül a rekordok frissítésével.

2019. 07. 19. Péntek

Odata-n keresztüli geokódolás folytatása. Korábban létre lett hozva a CÍM_EGYBEN tábla CIM, LONG, LAT mezőkkel (utóbbi kettő az összes rekordra egyelőre null), ennek kéne az összes elemét kiegészíteni a koordinátákkal. Ehhez végig kéne menni az összes rekordon (~2300), viszont egyelőre probléma volt, hogy az odata service max. 1000 rekordot enged lekérdezni egyszerre. Először úgy akartam ezt megkerülni, hogy mindig 1000 null LONG, LAT értékkel rendelkező rekordot kérdezek le, amíg a visszaadott JSON kevesebb, mint 1000 bejegyzést nem tartalmaz. Itt problémába ütköztem, ugyanis alapértelmezésben az xsodata service nem engedélyezi a null érték használatát. Utánanézve megtaláltam, hogy a .xsodata settings részében kell ezt beállítani, ugyanakkor ugyanitt lehet megadni a visszaadott rekordok számának felső korlátját, úgyhogy a probléma mindkét

szempontból megoldódott. Következőnek az odata service-n keresztüli rekordfrissítésnek kell utánanézni.

Ennek JS-ben (vagy arra épülő keretrendszerben) való megvalósításához találtam az o.js library-t, először erre átírtam a már korábban létrehozott angularos miniprojektet (az eddigi httpclient alapú service helyett), majd csináltam egy odata patch-et, a null LONG, LAT értékű címekhez (amennyiben van találat hozzá). Itt cross-origin-resource-sharing problémát kaptam a bing API-tól, annak köszönhetően, hogy az egyik cím tartalmazott kettőspontot, ami úgy látszik, még a keresési query-ben sincs megengedve. Ennek kiküszöbölésére az adatbázisban az összes kettőspontos címben a kettőspontot szóközre cseréltem, utána viszont amikor próbáltam az adatokat felvinni status 405 „Method not allowed" http választ kaptam.

2019. 07. 22. Hétfő

Cél a method not allowed válasz kiküszöbölése. Ezt látszólag sikerült megoldani egy delete majd post használatával, a frissített rekord az adatbázisban is jól szerepel. Az összes rekord frissítése után úgy gondoltam, hogy visszafelé a koordinátákból a címet egységes formázásban lehetne visszakapni, így tovább lennének tisztítva az adatok. Ennek tesztelgetése közben nyilvánvalóvá vált, hogy ezek az átalakítások nem bijektívek, pl. volt olyan cím, ami az eredetihez képest ~130 háznyit odébb került az utcán, sőt, a visszakapott házhoz tartozó koordináta különbözött az eredetileg megadottól. Ezen felül már az elején tudni lehetett, hogy házszámnál részletesebb adatok, mint pl. emelet/ajtó majdnem biztosan el fognak veszni, ráadásul több koordinátához nem talált az API címet, ezeknél címnek a „HIBA" értéket adtam meg. Ezek miatt élesben aligha lenne használható a megoldás, ezzel együtt azért végigviszem a feladatot, egyrészt, mert tanulni lehet belőle, másrészt nincs kiosztott alternatíva.

Ezek után a „HIBA" értékű bejegyzésekhez visszanéztem az eredeti címet, több cím esetén a többit töröltem, majd a hibát átírtam az eredeti címre. Ezek után létrehoztam egy új főtáblát, amiben már ezek az új címek szerepelnek.

2019. 07. 23. Kedd

A tegnap létrehozott táblából csináltam egy kalkulációs nézetet, utána az eddig meglévő angularos projectet mögé ezt raktam be. Az oldalon megnézve több cím is tartalmazott 'undefined' tagot (az utca/házszám részben). Az adatbázisban visszanézve 577 társaság címében szerepelte az undefined valamelyik résznél, illetve 124 egyedi címben szerepelt összesen. A legegyszerűbb megoldás ezekben az esetekben visszaírni az eredeti címet. Végül pedig a címekhez tartozó térképmegjelenítéshez hozzáadtam, hogy ha az adott címnek vannak eltárolva long-lat koordinátái direktben azokból töltse be a térképet a cím helyett.

Ezzel ezt a minifeladatot egyelőre lezártnak gondoltam. Az első két héti állapot óta főként a long-lat kódolással, az adatokhoz való hozzáfűzésével dolgoztam, esetleg a címek egységesítésével. Ezt szisztematikusan akartam végezni, a bing api felhasználásával,

azonban az eredeti-új címeket összehasonlítva elég magas a hibaszázalék, nem igazán mondanám sikeresnek.

2019. 07. 24. Szerda

Először grafikus megjelenítéseket próbálgattam (vc++/openGL, html canvas), kipróbáltam pár online mintaprojektet, dokumentációkat olvasgattam. Amennyiben továbbra se nagyon történik semmi, valószínűleg visszatérek az előző projekthez és próbálok extra funkcionalitást hozzáadni, vagy további adattisztítást végezni.

2019. 07. 25. Csütörtök

Újra visszanéztem az előző projekthez. Eddig 2 külön kalkulációs nézet volt, az egyikben simán egybe-címekkel, a másikban a cím kulcs a koordinátákhoz (és az új, oda-vissza geokódolt, többnyire eltérő címek szerepelnek bennük). Ezek helyett csináltam egyet, amiben a cím egyfelől kulcsa a koordinátatáblának, másrészt egy másik táblában kapcsolódik hozzá az új cím is.

Korábban a táblában szerepeltek 'Ismeretlen' címek is. Ezt akkor lecseréltem `null` értékekre. Most feltűnt, hogy ezzel eltűntek az ilyen címmel rendelkező rekordok a kalkulációs nézetből. Ezzel több időt elölve, a kalkulációs nézetet többször újracsinálva végül rájöttem, hogy a főtábla – koordinátatábla csatolásánál inner join volt alkalmazva, így az eredménybe nem került be olyan eredmény, amiben bal oldalt null szerepelt. Ezt (és mivel az a felbontást tekintve helyesebbnek tűnt, még ha ott nincsenek is null értékek, ami ténylegesen megindokolná, az összes többit is) kicseréltem left outer join-ra, így visszajöttek ezek is.

A kalkulációs nézet átalakítása során több táblát is töröltem, egyrészt az alternatív főtáblát, amiben az új címek szerepeltek, másrészt az alternatív koordinátatáblát, amiben szintén ezek szerepeltek. Ezek használva voltak az angularos applikációban, ezért következőnek ott kell a kapcsolódó részeket átalakítani.

Mivel a koordinátáról visszakódolás nem volt sikeres, ezért a hozzá tartozó gombot és eseménykezelőt teljes egészében kivettem. A triviális névátírások után még annyit csináltam, hogy a címek térképen való megjelenítésénél, ha az adott bejegyzésnél null a LAT vagy LONG koordináták értéke, akkor a címből előállítunk egy címet, amit aztán a koordinátatáblába rögtön fel is töltünk a címhez, illetve az eddigi mindig címből való kódolás helyett rögtön a tárolt LAT LONG koordináták alapján jelenítjük meg a térképet.

2019. 07. 29. Hétfő

A reggeli áramszünet után a kijelölt közös git repository-t próbáltam lokálisan klónozni, azonban korábban elfelejtettem usert küldeni emailben, és nincs rá jogosultságom, úgyhogy egyrészt arra várok, másrészt a csoportból többünknek nem világos, hogy pontosan mit is kéne feltölteni. Megvolt ugyanis a felosztás (saját esetben „adatmodellezés"), azonban ehhez feladatot nem kaptunk, és jobb híján vegyesen csináltam frontendet/backendet, amiból viszont kérdéses mennyit lenne érdemes

feltölteni modellezés címszó alatt. Elméletben ma délután valamikor be fog nézni Vincellér tanár úr, valószínűleg őt kéne megkeresni a kérdéssel. Különben ha már webes technológiákba merültem valamennyire bele amúgy is, most épp az ASP-t tanulmányozgattam.

2019. 07. 30. Kedd

A tegnapi email-re válaszul kaptam a repository-hoz egy meghívót, így már le tudtam lokálisan klónozni, így már csak az a probléma maradt, hogy mit kéne feltölteni.

2019. 07. 31. Szerda

Sok haladás sokáig most se történt, azonban délután bejött Vincellér tanár úr, felmérésre került, hogy kinek van elképzelése, mit csinálna, illetve akiknek nincs, fel lett vázolva egy feladat, ami általános szenzor mérési adatainak feldolgozásról/aggregálásáról, majd ezekből valami szolgáltatás nyújtásáról szólna. En személy szerint ezen a vonalon haladok tovább.

2019. 08. 01. Csütörtök

A szenzoros téma megadása után próbáltam tájékozódni általános kimeneti formátumokról, hogy néz ki és milyen felépítésű, ez alatt jutottam el az IoT (Internet of Things)-hez. Ezzel kapcsolatos dolgokat olvastam a nap nagy részében, ezen kívül a jövőbeli bemeneti első körbéli adatfeldolgozáshoz/aggregáláshoz tanulmányoztam az sqlscriptet, triggereket, annak kapcsán, hogy a felvitelnél az adatokat még feltöltés előtt, pl. egy webes alkalmazással, vagy pedig triggerrel feltöltés közben dolgozzuk-e fel.

2019. 08. 01. Péntek

További online anyagokat kerestem, esetleg valami feldolgozható bemeneti adathalmazt, emellett más, ugyanezen témájúakkal kicsit konzultáltunk, egy esetleges tárolási formátumot (~ id, idő, long, lat, adat | esetleg gondolva rá, hogy a szenzor mozog/álló), de nagyon további haladás nem történt.

2019. 08. 05. Hétfő

A nap közepe felé volt egy hosszabb megbeszélés, ez után kaptunk a szenzoros feladathoz egy adatszettet. Ezt valamennyire tanulmányoztuk, felbontás ügyében úgy gondoltam, idő-szenzor-adatok felbontásban kéne szétszedni, ahol az idő-szenzor összetett kulcs.

2019. 08. 06. Kedd

Létrehoztam a kitalált felbontásra egy hdbcds artifact-ot, illetve hozzá egy odata servicet. Ezek után az adatok odata-n keresztüli feltöltésére, esetleges előfeldolgozására kerestem megoldást .NET-ben vagy c++-ban, mivel a kapott adatok tömbösítve voltak csak időpont szerint, a saját elképzelés szerint viszont időpont és szenzor alapján lennének bontva. Időközben azonban inkább úgy gondoltam, a mostani bemeneti file-

okat elég egyszerűen parse-olni egy konzol applikációval, hiszen ha lesz köztes feltöltőfelület, az úgyse tömbösített adatokat fog kapni, hanem egyesével a szenzorokét stream-ben, a mostani adatokat pedig file importként elég feltölteni.

2019. 08. 07-08. Szerda-Csütörtök

Az adatok átalakítása egy program c++ segítségével megtörtént, majd az import is. Az adatokat tesztelendő/valami az adatokra épülő szolgáltatás létrehozásához csináltam egy angular projektet, ebben első körben csak egy táblázatban ellenőrizni az adatokat. Ekkor feltűnt, hogy az odata-n keresztül elérhető idő az adatbázis által tárolt Integer formában van, ezért csináltam rá egy kalkulációs nézetes dimenziót, ahol számolt oszlopként hozzáraktam az adatokhoz egy stringgé castolt időt is. Ekkor feltűnt, hogy meg lehet adni az oszlopoknak kimeneti/bementi konverziós függvényt, ezért csináltam egy tárolt függvényt, ami egy SECONDDATE paraméterből visszaad egy stringet. Ezt hozzáadtam a db modulhoz, azonban amikor alkalmazni próbáltam a dimenzióban, fordítási hibát kaptam, időközben be lett jelentve egy újabb konzultáció, ezért végül maradtam a számolt oszlopnál.

A konzultálásnál azt mondták célszerű lenne egy idő dimenziót létrehozni a dologból, ami dátum/idő szerinti hiearchiát is tartalmaz, rövid utánajárás alapján ehhez az általam éppen nem látható, a rendszer részét képező táblák kellenének, úgyhogy ez egyelőre halasztva lett.

Amit még továbbá mondtak, hogy az adatbányász részleggel beszéljünk össze, hogy milyen formátumú táblák kellenek, ennek eredményeként lett egy 2. tábla az első adataiból, hozzáfűzve egy egyszerű Integer kulccsal.

2019. 08. 09. Péntek

Mivel a time dimension-höz nem férek hozzá a _SYS_BI sémához, ezért az adatstreamen keresztüli feltöltéshez tértem vissza. Az egyszerű elképzelés szerint az angular app url-ben megkap egy rekordot, amit aztán odata-n keresztül feltölt azt az első, összetett kulcsos táblába. Úgy gondoltam, csinálok egy triggert, ami ekkor az adatot feltölti a 2., egyszerű kulcsos táblába maxkulcs+1 kulccsal, azonban először rosszul hoztam létre a triggert (nem történt meg a kulcshoz a +1 hozzáadása), viszont utólag már nem volt engedélyem törölni/inaktívvá tenni a triggert, úgyhogy egyelőre ezen a fronton se tudok sokat kezdeni.

Home office

Adatbázis

Az itthoni munka alatt először is rendbe raktam az adatbázist. Mivel a korábbi triggert nem tudtam törölni, ezért az adatok exportálása után a cockpit-ből töröltem az adatbázis instance-t, majd újrageneráltam azt. Az adatok beillesztésénél több bejegyzés feltöltésénél is hiba keletkezett, ezért, és mert elegánsabb megoldásnak gondoltam,

módosítottam a korábbi c++-ban írt parsert, hogy egyetlen filestream helyett filestream-ek kollekciójából állítsa elő a kimenetét. Az így kapott file feltöltése előtt még hozzáadtam Hana-ban a projekthez egy triggert, ami az alapformátumú (azaz az egyszerű kulcsot nem tartalmazó, parser által előállított formátumra illeszkedő) rekordok feltöltésénél beszúr egy egyszerű integer kulcsos bejegyzést a másik adattáblába, ezután a parse-olt adatok importjánál mindkét tábla fel lett töltve.

Frontend

Ez után a Hana odata service-t felhasználva csináltam egy feltöltési lehetőséget az angular-os frontenden keresztül, ami ha GET paraméterként kap legalább 'Time' és 'Sensor' paramétereket, feltölti az adatokat egy átmeneti „Upload" táblába. Ennek mezői megegyeznek az összetett kulcsos adattábla mezőivel, azonban itt a Time mező String. Ennek oka, hogy az eredetileg dátumként tárolt mezőt az OData a háttérben ténylegesen tárolt Integerként jelenítette meg, ezért a táblára épülő kalkulációs nézetben az eredeti mező helyett egy számolt, Stringgé konvertált mező kerül megjelenítésre, amin keresztül a visszatöltés közvetlenül nem lehetséges. A többi paraméter az adatbázis séma alapján Temperature, Humidity, Pressure, Pm1, Pm25, Pm10 néven adható meg, azonban ezek mivel lehetnek null tartalmúak, megadásuk nem szükséges. Az adatok Upload táblából való adatbázisba való hozzáadására egy újabb triggert hoztam létre, amely Upload-ba való beszúrás után az eredményt az összetett kulcsos táblába is feltölti (ahonnan aztán a másik trigger az egyszerű kulcsosba is beszúrja azt).

Utoljára még hozzáadtam az oldalhoz egy szenzorazonosító szerinti szűrőmezőt, ami az adatbázisban tárolt szenzorazonosítókból ad választási lehetőséget.