

Dokumentáció

07. 01.

Kick-off esemény végighallgatása, érdeklődési körök felvázolása, kezdetleges csoportfelosztás, ismerkedés az SAP Cloud Foundry-val.

07. 02.

Önálló munka kezdete, szükséges ismeretek elsajátításának megkezdése.

developers.sap.com oldal böngészése, ez alapján próbálkozás Node.js modul létrehozásával a jövőbeni front end kialakításához. Akadályba ütközés bizonyos funkciók hiánya miatt az általunk alkalmazott platformon. Ugyanerre vezetett egy HTML5 modul létrehozására való kísérlet.

Félidőben témavezetői segítséggel első táblázat létrehozása SAP HANA-ban. Próbálkozás „Hello World” alkalmazás létrehozásával, szerverproblémák miatt sikertelenül.

Maradék időben informálódás Node.js-ről.

07. 03.

JavaScript nyelv tanulása további instrukciók megadásáig.

9:40 – mail érkezése elsajátítandó anyagokról.

9:52 – tanári tájékoztató a legfontosabb témákról, a gyakorlat időbeosztásáról.

A nap hátralévő részében az ajánlott linkek olvasása és jegyzetelése. Ezek a következők:

- XSA dokumentáció:
<https://help.sap.com/viewer/4505d0bdaf4948449b7f7379d24d0f0d/2.0.03/en-US/d8226e641a124b629b0e8f7c111cd1ae.html>
- CDS dokumentáció:
<https://help.sap.com/viewer/09b6623836854766b682356393c6c416/2.0.02/en-US/0b1eb07d74ec4f91947ff4cc4f557429.html>
- PAL dokumentáció:
<https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/2.0.03/en-US/c9eed704f3f4ec39441434db8a874ad.html>

07. 04.

Dokumentációk feldolgozásának folytatása.

11:14 – tanári oldalra csv fájl felkerülése, információk meghallgatása a teendőkkel kapcsolatban. Szerverhiba miatt a feltöltés nem működik az SAP HANA platformon, így az adatok tisztítását kezdjük meg a csv fájlban.

07. 05.

Szerverhibák okainak felmérése témavezetői segítséggel, a megoldásig informálódás az OData-ról és a front end lehetőségeiről.

12:25 – saját szerver elkészül, ezen már működik az adatbázis, elkezdődhet a táblázat létrehozása és a csv fájl importálása. Amint megvan, a maradék időben további adattisztítással foglalkozunk.

Az adatok feltöltésének lépései:

- Workspace-be „New Project from Template” (nálam a *nyarigyak* nevet kapja)
- *nyarigyak*-ba „New SAP HANA Database Module” (*nyaridb* nevet kapja)
- *nyaridb src* mappájába „New HDB CDS Artifact” (*tablazat.hdbcds*)
- a kontextus nevét a vizuális szerkesztőben „*tablazat*”-ra állítottam, majd dupla kattintással belementem
- a kontextuson belül létrehoztam a „*database*” entitást, ez fogja tárolni az összes adatot
- dupla kattintás a „*database*”-re megnyitja az adattagok hozzáadásának lehetőségét; itt manuálisan felvettem a csv fájl fejléceiben megadott mezőket, beállítottam a mezőtípusokat
- a *nyarigyak*, majd a *nyaridb* sikeres buildelése után átmentem a Database Explorerbe
- itt rákattintottam az „Add a database to the Database Explorer” (plusz ikonnal jelölt) lehetőségre, és a felugró listából kiválasztottam a *nyarigyak*-ot
- ezt követően a bal menüsávban megjelent az imént létrehozott adatbázis, a „Tables” fülben megtaláltam a „*database*”-t
- a „*database*”-re jobb gombbal kattintva megjelent az „Import Data” lehetőség, amire kattintva egy felugró menüben megadtam a csv fájl forrásnak
- néhány tulajdonságot az SAP-s táblázat és a csv között manuálisan kellett beállítani, ezek után az importálás sikeres volt
- további adattisztítás szükségessége miatt a Database Explorer konzolablakát megnyitva, SQL parancsok segítségével készítettem egy *MODOSITOTT* nevű táblát, melybe átmásoltam a *database* adatait, majd ezeket update-eltem (2 sor törlésre került hibás, illetve teljesen hiányzó adatok miatt)

Az első hét összefoglalása: serverhibák miatt az első hét javarészt az elméleti alapok elsajátításáról szólt; péntek délutánra a saját serveren mindenki elkészítette a saját táblázatát, importálta az adatokat.

07. 08.

Az első órában az adatok megjelenítésének csoportos tervezése. Általános megegyezés az adatok szűrésére év és/vagy megye/régió alapján.

Következő lépés az OData létrehozása; ehhez először anyagok olvasása (developers.sap.com, help.sap.com). Hiábavaló próbálkozás az olvasottak reprodukálására.

14:12 – tanári megbeszélés során kiderült, hogy az OData jogosultsági problémák miatt nem működik. Ezt követően az előző heti munkát foglaltuk össze szóban.

07. 09.

Adatok további tisztítása.

10:23 – témavezetői segítséggel sikerül létrehozni az ODatát. Ennek lépései:

- *nyarigyak*-ba „New Node.js Module” (fontos az „Enable XSJS support” kipipálása, a modult *odata*-nak neveztem el)
- *odata lib* mappájába „New File” (*myodata.xsodata*), ebbe a következő kód került:

```
service{
  "MODOSITOTT" as "DATABASE";
}
```
- *server.js* fájl 11. sorában a *redirectUrl*-t átírtam „*/myodata.xsodata*”-ra
- build után kaptunk egy linket (<https://oktnb132.inf.elte.hu:51045/myodata.xsodata>), ennek végére különböző lekérdezéseket illesztve kapjuk meg a kívánt adatokat (pl. [https://oktnb132.inf.elte.hu:51045/myodata.xsodata/\\$metadata](https://oktnb132.inf.elte.hu:51045/myodata.xsodata/$metadata))

11:30 – vissza az önálló munkához. Altáblázatok létrehozása, melyek a kalkulációs nézethez fognak kelleni. Ezek „kereteit” a *nyarigyak/nyaridb/src/tablazat.hdbcds* fájl vizuális nézetében csinálom meg, csakúgy, mint egykor a *database*-t. Mikor végeztem a létrehozással, a Database Explorer konzolablakában, SQL utasításokkal töltöttem fel az altáblázatokat a megfelelő adatokkal.

A következő altáblázatokat hoztam létre:

- TARSKODOK(TARS_ROV_NEV, TARS_HOSZ_NEV, CIM_EGYBEN, TARS_TIPUS_KOD, ADOSZAM, GAZD_FORM_KOD, CEGALL_KOD, NEMGAZD_AG_KOD, NEMGAZD_AGAZAT_KOD, NEMGAZD_SZAKAGAZAT_KOD, JEGYZ_TOKE_ERT_HUF, ORSZAG_KOD, REGIO_KOD, MEGYE_KOD, TELEPULES_KOD, ASZ_EVE)
- TARSTIPUS(TARS_TIPUS_MEGNEV, TARS_TIPUS_KOD)
- GAZD(GAZD_FORM, GAZD_FORM_KOD)
- CEGALL(CEGALL, CEGALL_KOD)
- NEMAG(NEMGAZD_AG_MEGNEV, NEMGAZD_AG_KOD)
- NEMAGAZAT(NEMGAZD_AGAZAT_MEGNEV, NEMGAZD_AGAZAT_KOD)
- NEMSZAKAGAZAT(NEMGAZD_SZAKAGAZAT_MEGNEV, NEMGAZD_SZAKAGAZAT_KOD)
- ORSZAG(ORSZAG, ORSZAG_KOD)
- REGIO(REGIO, REGIO_KOD)
- MEGYE(MEGYE, MEGYE_KOD)
- TELEPULES(TELEPULES, TELEPULES_KOD)

07. 10.

Olvasás OData felhasználási lehetőségeiről.

10:00 – témavezető segítségével Angular telepítése, de nem működik a gépeken.

A következő órák azzal telnek, hogy a témavezető és az egyik hallgató kolléga működőképpé tegye az Angulart, közben mindenki más önálló munkát végez.

A nap végére az Angular-project sikerrel járt, kör-emailben kaptunk egy tömörített mappát, amely keretül szolgálhat saját Angularos kódunk megírására.

07. 11.

Calculation View létrehozása egy hallgató kolléga segítségével. Lépései:

- *nyarigyak src* mappájának *.hdbconfig* fájljában *plugin_version* átírása 2.0.30.0-ra (csak ezzel működik a kalkulációs nézet)
- szintén *src* mappába „New Calculation View” (*Calculation.hdbcalculationview*)
- Create Join-nal behúzok egy joint a szerkesztőbe, amit „data”-ra nevezek át
- „Add Data Source” funkcióra kattintás után megadom az altáblázatokat forrásként
- ezután dupla kattintás a „data”-ra – Join Definition fülben az inner joinok meghatározása az adattagok összekötésével
- ezt követően a Mapping fülbe az adattagok behúzása (mindegyiket csak egyszer, tehát mivel a MEGYE-ben szerepel a MEGYE és a MEGYE_KOD, a TARSKODOK-ból már nem húzom be a MEGYE_KOD-ot még egyszer)
- szerkesztő bezárása, dupla kattintás az Aggregationre
- Mapping fülben itt is minden adattagot behúzunk
- build után működik a Data Preview

A maradék időben az előző nap kapott Angularos kód szerkesztése úgy, hogy az saját OData-val működjön; a munkaidő végéig sikertelen.

07. 12.

További munkák az Angularos kódon, félidőben sikerült a keretet saját OData-ra futtatni. A maradék idő a megjelenítés szerkesztésével telt (cím átírása, felesleges szövegek törlése, több szűrő megvalósítása).

Alább felsorolom a „keretkód” módosításait.

odata.service.ts:

configUrl-t át kellett írni saját odatára: configUrl = '/hana/myodata.xsodata/';

proxy.conf.js:

Ebben a fájlban adjuk meg, hogy az előző configUrl elején mi az a „hana”. Ehhez a targetet kellett módosítani: target: "https://oktnb132.inf.elte.hu:51045",

proxy.conf.json:

Itt is át kellett írni a targetet sajátára: target: "https://oktnb132.inf.elte.hu:51045",

app.component.html:

Ebben a fájlban írjuk le a megjelenítendő oldal kinézetét. Egy üdvözlő címsor után az oldalon kirajzolódik egy diagram, melyen a cégek bevételei láthatóak. A megjelenített adatokat évek és régiók szerint szűrhetjük (a két szűrő egyszerre érvényesül). A szűréshez a listaelemeket odata segítségével kérdezzük le, melyet a következő fájlban állítunk be.

app.component.ts:

A megjelenítés háttérlogikája. Az ngOnInit függvény beállítja a szűrők kezdőadatait, lekérdezi az odatából a listaelemeket. A show_graph függvény kirajzolja a grafikont, az onChange pedig a grafikonon megjelenő adatokat módosítja a beállított szűrők függvényében. Ebben a fájlban szükség volt a kalkulációs nézet lekérdezéseire is, így az SAP Web IDE-ben lévő myodata.xsodata fájlt ki kellett egészíteni:

```
service{  
  "MODOSITOTT" as "DATABASE";  
  "Calculation" as "kalkulacio";  
}
```

A második hét összefoglalása: a második hét végére rendelkezésre áll egy, a csv fájlból importált adatbázis SAP HANA-ban, ehhez tartozó altáblázatok, kalkulációs nézet és OData, illetve az ODatát felhasználó, még kezdetlegesnek mondható megjelenítés. A hétvége feladata a dokumentáció elkészítése, illetve a továbbhaladás átgondolása, célkitűzés a következő 4 hétre.

Megjegyzés: az elkészült „miniprojekt” az src mappában, a webide.zip és angular.zip állományokban található.

07. 15, 07. 16.

Első két hét miniprojektjeinek és dokumentációjának véglegesítése, ezek feltöltése Gitre. A végleges csoportok kialakítása.

07. 17.

Mivel az adatbányászat témakört választottam, elkezdtem olvasni és jegyzetelni az SAP PAL (Predictive Analysis Library) dokumentációt.

(<https://help.sap.com/viewer/2cfbc5cf2bc14f028cfbe2a2bba60a50/1.0.12/en-US/c9eed704f3f4ec39441434db8a874ad.html>)

11 óra tájékán tanári felügyelet alatt, kivetítveadtunk beszámolót az addigi munkánkról. A tanároktól utána visszajelzést és tippeket kaptunk, hogyan fejleszthetnénk tovább a kezdetleges projekteket. (Pl. a legnagyobb céget ki kéne venni a megjelenítésből, mert mellette a többi túl kicsi; az 1000-es odata limit oka ismeretlen; téérkép fontos lenne.)

07. 18, 07.19.

PAL dokumentáció jegyzetelése. (A jegyzetről képet terveztem beszúrni, de végül arra jutottam, az nem e dokumentáció célját szolgálná.)

A harmadik hét összefoglalása: A hét első fele az előző két hét munkájának egységbe fogásával és lezárásával telt el, második felében a PAL-ról informálódtam.

07. 22.

A dokumentációban olvasottak gyakorlatba való átültetésével való próbálkozás.

Először a következő SQL-lekérdezésekkel ellenőriztem, telepítve van-e a PAL:

```
SELECT * FROM "SYS"."AFL_AREAS" WHERE AREA_NAME = 'AFLPAL';  
SELECT * FROM "SYS"."AFL_PACKAGES" WHERE AREA_NAME = 'AFLPAL';  
SELECT * FROM "SYS"."AFL_FUNCTIONS" WHERE AREA_NAME = 'AFLPAL';
```

Mivel mindent rendben találtam, tovább haladtam a könyvtár első procedúrájára, az „Affinity Propagation” (röviden „AP”) nevű klaszterezési eljárásra.

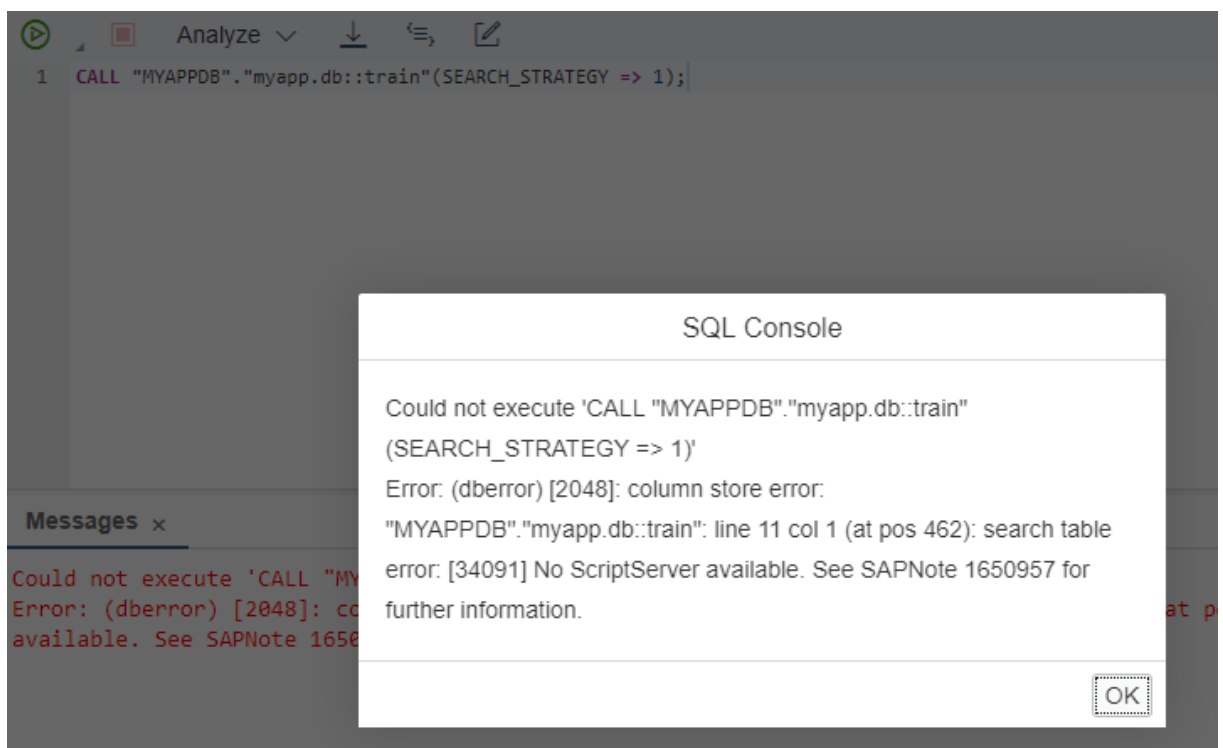
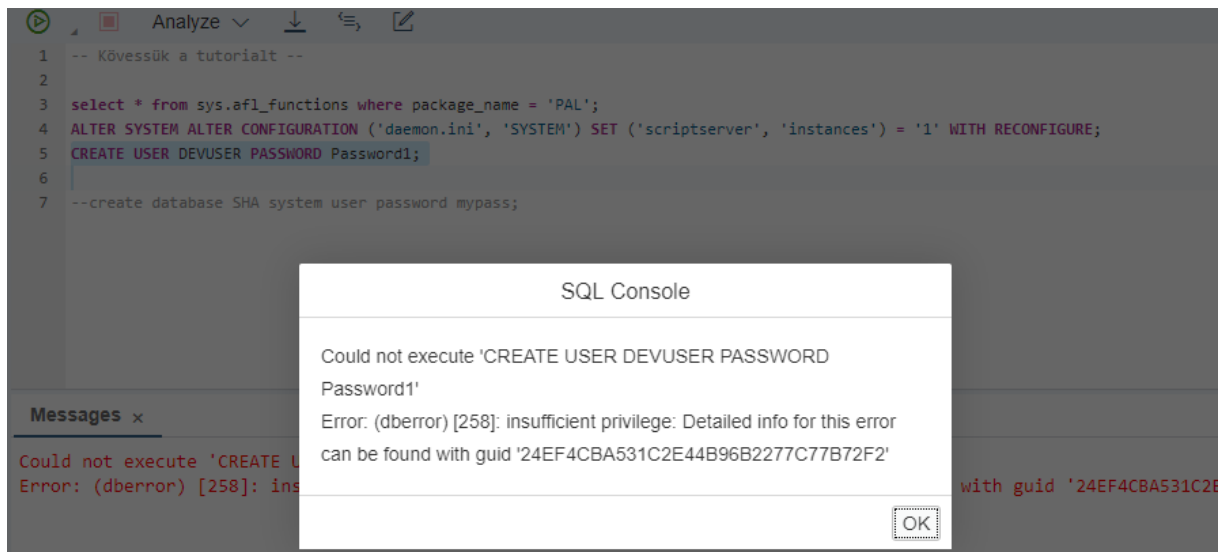
A procedúra létrehozásánál (a `CALL SYS.AFLLANG_WRAPPER_PROCEDURE_CREATE ('AFLPAL', 'AP', '<schema_name>', '<procedure_name>', <signature_table>);` parancs futtatásával való próbálkozáskor) azonban jogosultsági problémák adódtak. Ezt először annak tudtam be, hogy talán rossz táblatípusokat vagy sémanevet írtam, amelyekre nem lehet meghívni az adott eljárást. Emiatt ismételten az elméleti tudásom igyekeztem tovább mélyíteni.

07. 23.

Mivel a kibővített tudásommal sem sikerült létrehoznom a kívánt eljárásokat (kísérleteztem a „K-Means” eljárással is, ismételten hiába) jogosultsági problémák miatt, tanári segítséget kértem. A nap végén érkezett válasz egy YouTube lejátszási lista formájában: <https://www.youtube.com/watch?v=xzeG3kByKEs&list=PLkzo92owKnVwL3AWaWVbFVrfErKkMY02a&index=68>.

07. 24.

Elkezdtem a lejátsszási lista utasításait követni. Munkám hamarosan elakadt jogosultsági problémák és script szerver hiánya miatt. A pontos hibaüzenetek:



Ez ügyben ismét tanári segítséget kértem. A válasz megérkezéséig a videókból látottakat jegyzeteltem és értelmeztem.

07. 25.

Tanári segítséggel létrejött a script szerver. Így már sikerült a lejátsszási listában látottakat reprodukálni. Ez azt jelentette, hogy létrehoztam egy új MTA-t, azon belül egy StockPrices nevű adatbázist. Ezek után a PAL eljárásokhoz szükséges table type-ok és entity-k lekódolása következett. Ezekre épülhetett egy

train és egy predict procedúra megírása és meghívása (a tutorial egy Auto ARIMA eljárást mutatott be). A project probléma nélkül felépült, a procedúrák meghívása is sikeres volt. A videóban látható eredménytől azonban valamennyire eltért az enyém, ennek okát nem ismerem.

[illegible]

07. 26.

Mivel a hivatalos dokumentáció szerinti `CALL SYS.AFLLANG_WRAPPER_PROCEDURE_CREATE (...)` formátumú parancsok még mindig privilege error-t dobtak, megpróbáltuk a tutorialban látható Auto ARIMA eljárást a saját adatbázisunkra alkalmazni. Ennek első akadályá az volt, hogy az adatbázis nem rendelkezik egyedi azonosító oszloppal. A nap hátra lévő része azzal telt el, hogy módszert kerestünk egy ID oszlop beszúrására, sajnos sikertelenül.

A negyedik hét összefoglalása: PAL dokumentáció olvasása és jegyzetelése a lejátszási lista és a script szerver megérkezéséig. Utána a minta áttöltésére való kísérlet saját adatbázisra.

07. 29.

Megpróbáltam egy klaszter eljárást, a „DBSCAN”-t megírni a lejátszási listából ismert kód átírásával. Sajnos ezt még „buildelni” sem sikerült, ezt követően pedig már az eredeti (a tutorialból átvett, eddig működő) mellékproject sem épült.

Közben kiderült, hogy az ezzel való foglalkozás potenciálisan hiábavaló, ugyanis a Pythonos adatbányász csoportnak sikerült futtatnia egy elemzést SAP-n kívül, ami egyetlen klaszterbe sorolta az összes adatot.

07. 30.

Dokumentáció írása 1-ig. Utána videókat kerestem kifejezetten a klaszterezésről, de csak olyanokat találtam, amelyek SAP HANA Studiót használtak (egy olyan szoftvert, amellyel mi nem rendelkezünk), azon belül is a CALL SYS.AFLLANG WRAPPER PROCEDURE CREATE (...) parancsot.

07. 31.

Mivel a PAL könyvtár eljárásainak mindegyike igényel egy Integer típusú ID oszlopot, a nap nagy része azzal telt, hogy SAP-n beillesszünk egy ilyen adattagot a már meglévő adatbázisunkba. Miután ezt nem tudtuk elérni SQL parancsok segítségével, külön procedúrát írtunk, mely futtatás után sikeresen beillesztette az oszlopot, és feltöltötte azt.

Dél körül tanári megbeszélést tartottunk, ahol azt a feladatot kaptuk, hogy keressünk egy olyan adatbázist, melyen már végeztek elemzéseket, és ezeket hasonlítsuk össze saját PAL-elemzésekkel.

08. 01, 08. 02.

Ezen napok azzal teltek, hogy megfelelő cikkeket keressünk, illetve megpróbáljunk különböző PAL eljárásokat interpretálni a saját szerverünkön. Elsősorban az AP és a K-Means klaszterezési eljárásokat igyekeztünk elsajátítani, a fő problémát az okozta, hogy a paraméter-táblázatot hogyan töltsük ki.

Az ötödik hét összefoglalása: További ismerkedés a PAL-lal, az adatbázis formátumának igazítása a könyvtár eljárásainak igényei szerint.

Mivel a következő hetekben Tihanyi Balázssal végeztem közös munkát, a dokumentáció hátra lévő részét közösen írtuk meg.

PAL dokumentáció

A hatodik hét elején a témavezető tanároktól kaptunk egy adatbázist, mely különböző szenzorok által mért időjárás- és légszennyezés-adatokat tartalmazott (hőmérséklet, nyomás stb). Ezen kellett különböző adatbányász eljárásokat végrehajtanunk, a PAL könyvtár használatával.

Az első problémát az jelentette, hogy az oszlopok száma ellehetetlenítette egy olyan tábla manuális létrehozását, amelybe az összes adatot beolvashatnánk. Ezért az adatbázis első sorát, amely egyben tartalmazta az összes oszlopnevet (vesszővel elválasztva), kimásoltuk egy txt fájlba. Ezt követően írtunk egy programot, amely beolvassa ezt a txt fájlt, és egy másik, kimeneti txt fájlba kiírja az adattagokat soronként.

(Tehát ezt a szöveget:

„UTC_time,3_temperature,3_humidity,3_pressure,3_pm1,3_pm25,3_pm10,140_temperature,140_humidity,140_pressure,140_pm1,140_pm25,140_pm10,142_temperature,142_humidity,142_pressure,142_pm1,142_pm25,142_pm10, (...)” átalakította egy ilyen formátumúvá:

„UTC_time
3_temperature
3_humidity
3_pressure
3_pm1
3_pm25
3_pm10
140_temperature
140_humidity
140_pressure
(...)”.)

Ezután azzal szembesültünk, hogy SAP-ban az oszlopnevek nem kezdődhetnek számmal, ezért írunk kellett egy új programot, amely megcserélte a karakterek és számok sorrendjét (pl. „3_temperature”-t átírta „temperature_3”-ra). Ezt követően az első program segítségével ismételten szétválasztottuk az egy sorba szedett oszlopneveket, majd egy harmadik program segítségével mögéjük illesztettük az adattípus elnevezését, így végül egy olyan output fájlt kaptunk, amelynek tartalmát SAP-ba beillesztve létre tudtuk hozni a kívánt táblát.

```
(time_UTC : Integer;  
temperature_3 : Integer;  
humidity_3 : Integer;  
pressure_3 : Integer;  
pm1_3 : Integer;  
pm25_3 : Integer;  
pm10_3 : Integer;  
(...))
```

Megjegyzés: a fent leírt kódokat, txt-ket az src mappán belül cpp.zip állomány tartalmazza.

Miután a januári adatokat sikerült hiba nélkül beolvasni (a *bigDB* táblába), el kellett döntenünk, mely eljárásokat kívánjuk futtatni rajtuk. Választásunk az alábbi háromra esett:

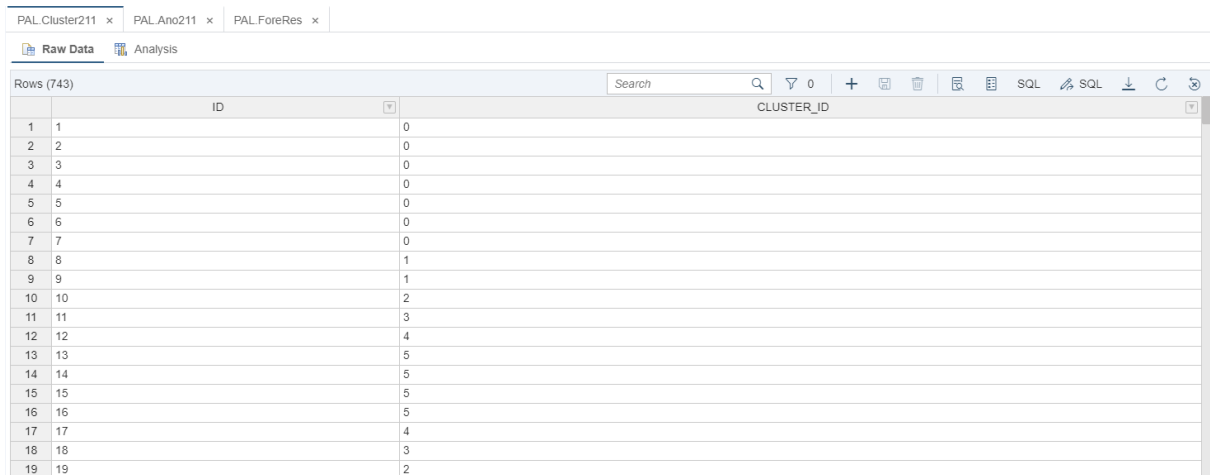
1. Affinity Propagation (AP, klaszterező algoritmus)
2. Anomaly Detection (kiugró adatok kimutatása)
3. ARIMA (az idősor-elemzés egy gyakran használt függvénye).

Megjegyzés: a továbbiakban említett fájlokat az src mappán belül az sap.zip állomány tartalmazza.

A fent említett három eljárás mindegyike megkövetel egy *Integer* típusú *ID* oszlopot, továbbá nem engedi meg, hogy az adatok között *NULL* is szerepeljen. Hogy ezeket a feltételeket teljesítsük, SQL parancsok segítségével kerestünk 5 szenzort (171, 211, 212, 223, 228), melyeknek legfeljebb egy sorában szerepelt ismeretlen adat, majd ezeknek azonos idejű, nem *NULL* adatait kimásoltuk egy új táblába (*fiveID*). Ezt a táblát előre úgy definiáltuk, hogy legyen egy azonosító oszlopa (*time_stamp*); ezt beolvasáskor az egyik szenzor hőmérséklet-adataival töltöttük fel, majd egy eljárás (*identity.hdbprocedure*) segítségével 1-től kezdődő, egyesével növekvő egész számokkal.

Ezek után elkezdhattuk az eljárások megírását. Mivel az AP-hez egy hallgató kollégától is segítséget kaptunk, ezzel kezdtük. Az *apfull* az egész *fiveID*-táblázatot nézi, és az alapján klaszterez, az *aptemp* csak az 5 hőmérséklet-oszlopot veszi figyelembe, míg az *ap211* csak a 211-es szenzor hőmérséklet-adataival számol.

Az *apfullproc*, amikor nem adtuk meg előre a klaszterek számát, 31 klasztert talált (az eredményt a *PAL.REAL_CLUSTER* táblában találjuk), az *aptemp* 40-et (*PAL.REAL_TEMP_CLUSTER*), az *ap211proc* pedig 24-et (*PAL.Cluster211*). Eredményeink helyadatokkal kiegészülve lennének igazán érdekesek, melyek segítségével megvizsgálhatnánk, az egymáshoz közeli települések tartoznak-e össze, vagy máshol kell keresni az összefüggéseket.



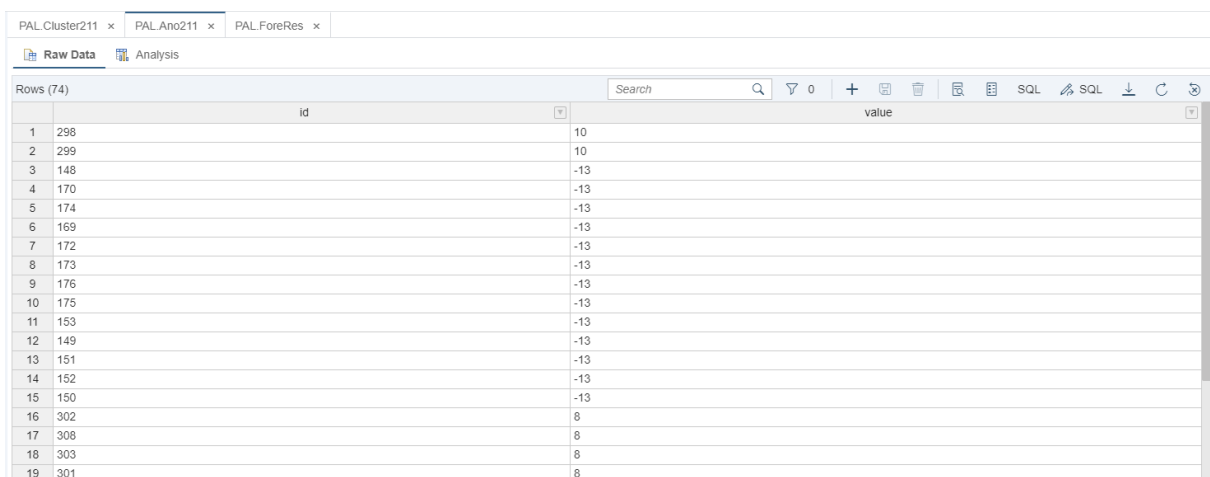
ID	CLUSTER_ID
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	1
9	1
10	2
11	3
12	4
13	5
14	5
15	5
16	5
17	4
18	3
19	2

Ábra: részlet a klaszter-elemzés eredményéből

A továbbiakban azt a döntést hoztuk meg, hogy egy szenzor (a 211-es) hőmérséklet-adataira fogunk koncentrálni, noha később az *ARIMÁ*-t futtattuk az egész *fiveID*-táblára is.

Először azonban az Anomaly Detection eljárást írtuk meg (*ano211.hdbafllangprocedure* és *ano211proc.hdbprocedure*). Ez 74 kiugró adatot talált a 743-ból (*PAL.Ano211*), vagyis nagyjából az adatok tizede mutatott a normálisnál nagyobb ingadozást. Az eljárás működését a futtatása után értettük meg igazán, amikor is világossá vált számunkra, hogy nem egy átlaghoz képest állapítja meg a függvény az outliereket, hanem a szomszédoktól való távolság alapján dönt.

A szenzoros adatok között találtunk egy null adatot, amit tetszőlegesen kitöltöttünk a szomszédok alapján. Ha a szomszédokhoz képest 1 fokkal nagyobb vagy kisebb hőmérsékletet adtunk meg, akkor nem került az outlierek közé, ha 2 vagy nagyobb volt a különbség, akkor az eljárás megtalálta és belerakta az eredménytáblába. Megfigyeltük azt is, hogy az eljárás akár teljes csoportokat is outlierek közé rakott, ha a csoport ugyan azokat az értékeket tartalmazta, és a csoport szomszédai 2 értékkel kisebbek vagy nagyobbak voltak.



id	value
1	298
2	299
3	148
4	170
5	174
6	169
7	172
8	173
9	176
10	175
11	153
12	149
13	151
14	152
15	150
16	302
17	308
18	303
19	301

Ábra: részlet az Anomaly Detection eredményéből

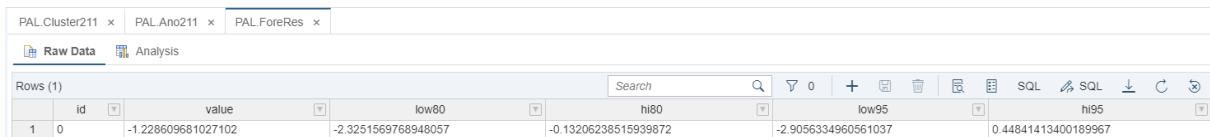
Egy hosszabb távú project során érdemes lehet a kiugró adatok okát vizsgálni, illetve az anomáliákat a szomszédokhoz igazítani, hogy megnézhessük, miként változnak az egyéb eljárások eredményei.

Végül az ARIMA következett, melyet a fent említetteknek megfelelően kétféle módon futtattunk. Az *artrainfull*, *artrainfullproc*, *arpredfull* és *arpredfullproc* az egész *fiveID* táblázatot megkapja inputként,

míg az *artrain211*, *artrain211proc*, *arpred211*, *arpred211proc* csak a 211-en szenzor hőmérséklet-adatait. Mindkét fajta eljárás eredményét a *PAL.TrainRes* és a *PAL.ForeRes* táblák tartalmazzák.

Az eredményeket látva jöttünk rá első eljárásunk hibájára, ugyanis nem vettük figyelembe, hogy az *ARIMA* egy végeredményt fog adni nekünk az input alapján, s ha ez az input vegyesen tartalmaz hőmérséklet, nyomás stb. adatokat, az előrejelzés értelmetlenné válik.

A második eljárás, amely csak a 211-es szenzor hőmérséklet-adatait vizsgálta, már pontosabb előrejelzést ad, kerekítve -2,91 és 0,45 fok közötti értéket ad eredményül.



PAL.Cluster211 x PAL.Ano211 x PAL.ForeRes x					
Raw Data Analysis					
Rows (1)					
	id	value	low80	hi80	low95
1	0	-1.228609681027102	-2.3251569768948057	-0.13206238515939872	-2.9056334960561037
					0.44841413400189967

Ábra: a 211-es szenzorra futtatott ARIMA eredménye

Összefoglalva tehát a PAL könyvtár három különböző területéről sikerült elsajátítanunk egy-egy eljárást. Egy hosszabb távú project célja lehetne az eredmények további elemzése, kiegészülve például helyadatokkal, s az eredmények alapján további, kifinomultabb adatbányászati eljárások írása és futtatása.