Bevezetés

A feladatmankó pdf áttekintése után a megfelelő sorrend kialakítása a feladat. Kezdeti terv lépései:

1. Adatfájlok struktúrájának megállapítása
2. ER diagram elkészítése
3. Adatok betöltése pandas dataframe-be
4. Adattisztítás
5. Táblákra bontás
6. Lekérdezések
7. Teljesítmény optimalizálás
8. Jogosultságkezelés

A feladat megoldása közben kiderült, hogy ez a sorrend nem feltétlenül ideális, mert ha hamarabb készítjük el az ER-Diagrammot, akkor előfodulhat, hogy az adattisztítás során az adattáblák struktúrája változik, így az ER-Diagram módosítására lehet szükség.

# Adatfájl struktúra megállapítás

Adagok.csv oszlopnevek javítás, új oszlopnevek:

* ADAGSZÁM
* Kezdet\_DÁTUM
* Kezdet\_IDŐ
* Vége\_DÁTUM
* Vége\_IDŐ
* ADAGKÖZI\_IDŐ
* ADAGIDŐ

A kezdet\_DÁTUM+Kezdet\_IDŐ oszlopokból képezhető egy Kezdet oszlop, hasonlóan a vég adatokkal Vége néven, természetesen datetime formátumban.

Az Adagok.csv-ben csak 32 értékes sor van, a többit mely NULL értékeket tartalmaz eldobjuk.

A Hutopanelek.csv-ben lévő Time oszlopokat is formázzuk pandas dataframe datetime formátumra, hogy a tartalomnak megfelelően tudjuk az adatokat kezelni.

# Adattisztítás

Hiányzó adatok keresése

Szerencsére nincs hiányzó adat sor-szinten, de hiányzik a 7. szenzor teljes adata.

Nem lehetünk biztosak abban, hogy az Adagidő valóban helyesen felvitt értékeket tartalmaz, így kiszámoljuk és összehasonlítjuk a meglévő adatokkal.

Megállapítjuk, hogy van-e hiányzó adat a Hűtőpanelek.csv-ben Dataframe.isnull() segítségével. Nincs hiányzó adat.

Végül megjelenítjük a 2 dataframe statisztikáit a describe() függvény segítségével, hogy átfogó képet lássunk az adatokról. Látjuk, hogy a Hűtőpanelek.csv hőmérséklet érték oszlopainak statisztikái mind NaN értékűek. Megvizsgálva néhány értéket a fájlban, látszik, hogy a lebegőpontos értéknél a tizedesvessző valóban vessző, ahogy Európában szokás jelölni. A pandas viszont pontot vár, így át kell alakítanunk az értékeket.

Hiányzik az ADAGKÖZI\_IDŐ. A névből arra következtetünk, hogy a két adag között eltelt idő. Ezt könnyen kiszámíthatjuk és hozzáadhatjuk mint Adagkozi\_ido oszlop.

# ER diagram elkészítése

Megállapítjuk az entitásokat. Ez könnyű, mivel 2 táblánk van amik szignifikánsan elszeparálták az entitásokat, úgy mint Adagok és Hűtőpanelek.

A terv, hogy kapcsolatot keressünk a 2 entitás között. Felfedezzük, hogy minden mérés valamelyik adaghoz kapcsolódik, így ezt a kapcsolatot kell megteremteni. Mivel nincs kézenfekvő hozzárendelési szabály(tekintve, hogy nincs a méréskhez adag-id rendelve) létrehozunk egy új oszlopot a hűtőpanelek táblában, mely egyértelműen azonosítja, hogy melyik mérés melyik adaghoz tartozik.

A későbbi feladatok miatt(pl: hogy demonstrálni tudjuk a UNION műveleteket), szétbontjuk a Hűtőpanelek táblát 15 táblára, mindnek ugyanaz a struktúrája.

* adagok
  + Primary Key: adagszam
  + Fields: adagkozi\_ido, adagido, kezdet, vege
* hutopanel1
  + Foreign Key: adagszam (references adagok.adagszam)
  + Fields: Ido, Ertek
* hutopanel2
  + Foreign Key: adagszam (references adagok.adagszam)
  + Fields: Ido, Ertek

A diagram of a computer code

Description automatically generated

Ezen az ER diagrammon csak 2 hűtőpanel táblát ábrázoltunk, a többi 12-t(mivel a 7. hiányzik) ugyanilyen struktúrában töltjük be az adatbázisba.

Hasonlóan járunk el az adatbázisba töltés esetén is. Azért, hogy demonstrálni tudjuk a feladatkiírásban jelölt műveleteket, nem ad hozzá hogy 14 merőpanel adatait mutassuk, elég ha 2 szerepel. Amennyiben az összesre szükség van, mindössze hozzá kell fűzni a többi táblát a lekérdezésbe, mivel az adatbázisban szerepelnek.

Az adatok betöltése megtörtént az adabázisba. Ettől a ponttól kilépünk a python világából és megérkezünk a tiszta SQL műveletek világába.

Az adatbázis műveletek sorban ismertetem(ahol szükséges) és a megállapításokat SQL commentként írom.

A csatolt adatbázis fájlban ellenőrizhető az elkészített táblák és annak tartalmai.

Mivel az sqlite nem alkalmas arra, hogy elsődleges/másodlagos kulcsot adjon táblához, így rögötön demonstrálni tudom, hogy kell új táblát létrehozni, a régi tartalmát átmásolni és átnevezni.

-- Oszlopok atnevezese

ALTER TABLE "adagok"

RENAME COLUMN "ADAGSZÁM" TO "adagszam";

ALTER TABLE "adagok"

RENAME COLUMN "ADAGIDŐ" TO "adagido";

ALTER TABLE "adagok"

RENAME COLUMN "Kezdet" TO "kezdet";

ALTER TABLE "adagok"

RENAME COLUMN "Vége" TO "vege";

ALTER TABLE "adagok"

RENAME COLUMN "Adagkozi\_ido" TO "adagkozi\_ido";

-- Elsodleges kulcs hozzaadasa

CREATE TABLE "new\_adagok" (

"adagszam" INTEGER PRIMARY KEY,

"adagido" REAL,

"kezdet" TIMESTAMP,

"vege" TIMESTAMP,

"adagkozi\_ido" REAL

);

INSERT INTO "new\_adagok" (adagszam, adagido, kezdet, vege, adagkozi\_ido)

SELECT adagszam, adagido, kezdet, vege, adagkozi\_ido FROM adagok;

DROP TABLE adagok;

ALTER TABLE "new\_adagok" RENAME TO "adagok";

-- Idegen kulcs hozzaadas

CREATE TABLE "new\_panel1" (

"time" TIMESTAMP,

"value" REAL,

"adagszam" INTEGER,

FOREIGN KEY ("adagszam") REFERENCES "adagok"("adagszam")

);

INSERT INTO "new\_panel1" (time, value, adagszam)

SELECT time, value, adagszam FROM "panel1";

DROP TABLE "panel1";

ALTER TABLE "new\_panel1" RENAME TO "panel1";

CREATE TABLE "new\_panel2" (

"time" TIMESTAMP,

"value" REAL,

"adagszam" INTEGER,

FOREIGN KEY ("adagszam") REFERENCES "adagok"("adagszam")

);

INSERT INTO "new\_panel2" (time, value, adagszam)

SELECT time, value, adagszam FROM "panel2";

DROP TABLE "panel2";

ALTER TABLE "new\_panel2" RENAME TO "panel2";

CREATE TABLE "new\_panel3" (

"time" TIMESTAMP,

"value" REAL,

"adagszam" INTEGER,

FOREIGN KEY ("adagszam") REFERENCES "adagok"("adagszam")

);

INSERT INTO "new\_panel3" (time, value, adagszam)

SELECT time, value, adagszam FROM "panel3";

DROP TABLE "panel3";

ALTER TABLE "new\_panel3" RENAME TO "panel3";

CREATE TABLE "new\_panel4" (

"time" TIMESTAMP,

"value" REAL,

"adagszam" INTEGER,

FOREIGN KEY ("adagszam") REFERENCES "adagok"("adagszam")

);

INSERT INTO "new\_panel4" (time, value, adagszam)

SELECT time, value, adagszam FROM "panel4";

DROP TABLE "panel4";

ALTER TABLE "new\_panel4" RENAME TO "panel4";

CREATE TABLE "new\_panel5" (

"time" TIMESTAMP,

"value" REAL,

"adagszam" INTEGER,

FOREIGN KEY ("adagszam") REFERENCES "adagok"("adagszam")

);

INSERT INTO "new\_panel5" (time, value, adagszam)

SELECT time, value, adagszam FROM "panel5";

DROP TABLE "panel5";

ALTER TABLE "new\_panel5" RENAME TO "panel5";

CREATE TABLE "new\_panel6" (

"time" TIMESTAMP,

"value" REAL,

"adagszam" INTEGER,

FOREIGN KEY ("adagszam") REFERENCES "adagok"("adagszam")

);

INSERT INTO "new\_panel6" (time, value, adagszam)

SELECT time, value, adagszam FROM "panel6";

DROP TABLE "panel6";

ALTER TABLE "new\_panel6" RENAME TO "panel6";

CREATE TABLE "new\_panel7" (

"time" TIMESTAMP,

"value" REAL,

"adagszam" INTEGER,

FOREIGN KEY ("adagszam") REFERENCES "adagok"("adagszam")

);

INSERT INTO "new\_panel7" (time, value, adagszam)

SELECT time, value, adagszam FROM "panel7";

DROP TABLE "panel7";

ALTER TABLE "new\_panel7" RENAME TO "panel7";

CREATE TABLE "new\_panel8" (

"time" TIMESTAMP,

"value" REAL,

"adagszam" INTEGER,

FOREIGN KEY ("adagszam") REFERENCES "adagok"("adagszam")

);

INSERT INTO "new\_panel8" (time, value, adagszam)

SELECT time, value, adagszam FROM "panel8";

DROP TABLE "panel8";

ALTER TABLE "new\_panel8" RENAME TO "panel8";

CREATE TABLE "new\_panel9" (

"time" TIMESTAMP,

"value" REAL,

"adagszam" INTEGER,

FOREIGN KEY ("adagszam") REFERENCES "adagok"("adagszam")

);

INSERT INTO "new\_panel9" (time, value, adagszam)

SELECT time, value, adagszam FROM "panel9";

DROP TABLE "panel9";

ALTER TABLE "new\_panel9" RENAME TO "panel9";

CREATE TABLE "new\_panel10" (

"time" TIMESTAMP,

"value" REAL,

"adagszam" INTEGER,

FOREIGN KEY ("adagszam") REFERENCES "adagok"("adagszam")

);

INSERT INTO "new\_panel10" (time, value, adagszam)

SELECT time, value, adagszam FROM "panel10";

DROP TABLE "panel10";

ALTER TABLE "new\_panel10" RENAME TO "panel10";

CREATE TABLE "new\_panel11" (

"time" TIMESTAMP,

"value" REAL,

"adagszam" INTEGER,

FOREIGN KEY ("adagszam") REFERENCES "adagok"("adagszam")

);

INSERT INTO "new\_panel11" (time, value, adagszam)

SELECT time, value, adagszam FROM "panel11";

DROP TABLE "panel11";

ALTER TABLE "new\_panel11" RENAME TO "panel11";

CREATE TABLE "new\_panel12" (

"time" TIMESTAMP,

"value" REAL,

"adagszam" INTEGER,

FOREIGN KEY ("adagszam") REFERENCES "adagok"("adagszam")

);

INSERT INTO "new\_panel12" (time, value, adagszam)

SELECT time, value, adagszam FROM "panel12";

DROP TABLE "panel12";

ALTER TABLE "new\_panel12" RENAME TO "panel12";

CREATE TABLE "new\_panel13" (

"time" TIMESTAMP,

"value" REAL,

"adagszam" INTEGER,

FOREIGN KEY ("adagszam") REFERENCES "adagok"("adagszam")

);

INSERT INTO "new\_panel13" (time, value, adagszam)

SELECT time, value, adagszam FROM "panel13";

DROP TABLE "panel13";

ALTER TABLE "new\_panel13" RENAME TO "panel13";

CREATE TABLE "new\_panel14" (

"time" TIMESTAMP,

"value" REAL,

"adagszam" INTEGER,

FOREIGN KEY ("adagszam") REFERENCES "adagok"("adagszam")

);

INSERT INTO "new\_panel14" (time, value, adagszam)

SELECT time, value, adagszam FROM "panel14";

DROP TABLE "panel14";

ALTER TABLE "new\_panel14" RENAME TO "panel14";

-- Szeretnenk latni 10 olyan merest az 1. panel es a 2. panel mereseibol,

-- ami a 13-as adagbol szarmazik. Kivancsiak vagyunk a kezdet es veg idokre,

-- hogy ellenorizzuk, megfelelo volt az adagszam hozzarendeles

SELECT

p1.time AS time1, p1.value AS value1, p1.adagszam,

p2.time AS time2, p2.value AS value2,

a.adagido, a.kezdet, a.vege, a.adagszam

FROM

panel1 p1

JOIN

panel2 p2 ON p1.adagszam = p2.adagszam

JOIN

adagok a ON p1.adagszam = a.adagszam

WHERE

a.adagszam = 13

ORDER BY

p1.time

LIMIT 10;

-- Indexek

CREATE INDEX idx\_adagok\_adagszam ON adagok (adagszam);

CREATE INDEX idx\_panel1\_adagszam ON panel1 (adagszam);

CREATE INDEX idx\_panel2\_adagszam ON panel2 (adagszam);

-- Ezek gyorsaitjak a joinokat

CREATE INDEX idx\_panel1\_time ON panel1 (time);

CREATE INDEX idx\_panel2\_time ON panel2 (time);

-- Ezek gyorsitjak az ido alapu szureseket

CREATE INDEX idx\_panel1\_time\_adagszam ON panel1 (time, adagszam);

CREATE INDEX idx\_panel2\_time\_adagszam ON panel2 (time, adagszam);

-- Ha olyan lekerdezesek vannak, ahol az ido es az adagszam egyszerre szerepel

CREATE INDEX idx\_adagok\_adagido ON adagok (adagido);

CREATE INDEX idx\_adagok\_kezdet ON adagok (kezdet);

CREATE INDEX idx\_adagok\_vege ON adagok (vege);

-- Allekerdezesek

-- Keressuk a panel1 meresek kozul a legnagyobb erteket azok kozul a meresek kozul,

-- aminek adagideje nagyobb mint 100

SELECT max(value)

FROM panel1

WHERE adagszam IN (SELECT adagszam FROM adagok WHERE adagido > 100);

-- JOIN

-- Minden adag kezdeti es veg meres idpontja a meresekkel egyutt az 1. panelrol

SELECT a.adagszam, a.kezdet, a.vege, p1.time, p1.value

FROM adagok a

JOIN panel1 p1 ON a.adagszam = p1.adagszam

ORDER BY a.adagszam, p1.time;

-- Azon adagok listaja, aminel nincs meresi ertek a panelekrol

SELECT a.adagszam

FROM adagok a

LEFT JOIN panel1 p1 ON a.adagszam = p1.adagszam

LEFT JOIN panel2 p2 ON a.adagszam = p2.adagszam

WHERE p1.adagszam IS NULL AND p2.adagszam IS NULL

ORDER BY a.adagszam;

-- Csak 13-28 adagokrol van meres

-- UNION & Allekerdezes

-- Minden meres megjelenitese a panelekrol

SELECT "time", "value", "adagszam", 'panel1' AS panel

FROM panel1

UNION ALL

SELECT "time", "value", "adagszam", 'panel2' AS panel

FROM panel2

ORDER BY "time";

-- Latjuk az osszes merest

-- Adagonkenti homerseklet atlag kiszamitas allekerdezessel es unionnal

SELECT "adagszam", AVG("value") AS average\_value

FROM (

SELECT "value", "adagszam" FROM panel1

UNION ALL

SELECT "value", "adagszam" FROM panel2

) AS combined\_panels

GROUP BY "adagszam"

ORDER BY "adagszam";

-- INSERT + allekerdezesek

-- Adagonkenti homerseklet atlag kiszamitas es tablaba illesztes

CREATE TABLE results\_avg\_measurements (

adagszam INTEGER PRIMARY KEY,

avg\_panel1 REAL,

avg\_panel2 REAL

);

INSERT INTO results\_avg\_measurements (adagszam, avg\_panel1, avg\_panel2)

SELECT a.adagszam,

(SELECT AVG(value) FROM panel1 WHERE adagszam = a.adagszam) AS avg\_panel1,

(SELECT AVG(value) FROM panel2 WHERE adagszam = a.adagszam) AS avg\_panel2

FROM adagok a;

-- Tranzakciokezeles

BEGIN TRANSACTION;

INSERT INTO adagok (adagszam, adagido, kezdet, vege, adagkozi\_ido)

VALUES (1001, 5.0, '2024-01-01 08:00:00', '2024-01-01 08:05:00', 1.5);

INSERT INTO panel1 (time, value, adagszam)

VALUES

('2024-01-01 08:01:00', 10.5, 1001),

('2024-01-01 08:02:00', 10.7, 1001),

('2024-01-01 08:03:00', 10.8, 1001);

INSERT INTO panel2 (time, value, adagszam)

VALUES

('2024-01-01 08:01:00', 20.5, 1001),

('2024-01-01 08:02:00', 20.7, 1001),

('2024-01-01 08:03:00', 20.8, 1001);

COMMIT;

Néhány kérdés megválaszolunk:

Miért érdemes a kapott adatokat Relációs adatbázisban kezelni?

Mert a kapott adatok között összefüggéssek vannak és ezen összefuggések kimutatására a legalkalmasabb a relációs adatbázis, ahogy nevéből is látszik.

Jogosultság kezelés:

Az adatok betöltése során olyan jogosultságot biztosítok, mely minden műveletet engedélyez. Itt nincs annak veszélye, hogy felülírok adatot vagy törlök, illetve illetéktelenül férek hozzá, mivel most hozom létre az adatbázist.

A lekérdezések során olyan jogosultságot adok, mely pontosan annyi adathoz enged hozzáférést amennyihez feltétlenül szükséges. Hasolóank az adatbázis műveletekhez, least-privileges szerint adok jogokat.

Tranzakció kezelésre akkor van szükség, ha fontos, hogy egy mérési időpont bekerüljön minden mérőpanel mérése a megfelelő táblába, ezzel elkerülve a hiányzó adatokat.

Milyen módszert javasolnának az adatműveletek gyorsítására, ha megkapnának

10 évnyi adatot és azt kellene feldolgozni?

Semmiképpen se sqlite adatbázisban tárolnám. 10 évnyi adatot amennyiben postgresql/mysql/oracle adatbázisban tárolnám. Adattisztítást továbbra is valamilyen python alapú technológiával valósítanám meg(pl pyspark) és cask utána tölténem az adatbázisba.

Adatbevitelhez milyen érvényesítési szabályokat, milyen kényszereket javasolnának?

Nincs negatív érték, mivel az iparágban ha negatív egy érték fémek hűtése során, az nem elvárt. Ez igaz az adagszámra, adagidőre is.

Biztonsági mentés:

Teljes mentés havonta, differenciált mentés hetente, inkrementális mentés naponta.

Szerepkörök:

Adattöltő: INSERT,SELECT jogosultság a vonatkozó táblákra.

Reporting: SELECT jogosultság azokra a táblákra amiből a reportot készíti.

Admin: Minden jogosultság