MESTERSÉGES SZENZORADATOK GENERÁLÁSA ÉS TÁROLÁSA

Szabó Anna[a], Pödör Zoltán[b*]

[a] *ELTE Informatikai Kar, programtervező informatikus*
[b] *ELTE Informatikai Kar, Numerikus Analízis Tanszék, egyetemi docens*

ABSZTRAKT

Bemutatunk egy relációs adatbázis struktúrát, mely alkalmas általános szenzoradatok fogadására és tárolására. Az adatbázis szerkezet magában foglalja a relációs adatbázisok nyújtotta strukturáltság előnyeit, ugyanakkor biztosítja azt a rugalmasságot, ami elvárható egy szenzorrendszer esetében. Így lehetőség van többek között új szenzorok, új mérések, mérési körülmények rögzítésére. Bemutatunk egy Python nyelven megvalósított adatgeneráló alkalmazást is, mely szenzorrendszerek működését szimulálva alkalmas egyrészt megfelelően beállított peremfeltételek mellett szenzoradatok generálására, másrészt az adatbázisba történő továbbítására és a későbbiekben ehhez kapcsolódó mobil-, asztali alkalmazások fejlesztésének támogatására, tesztelésére. A két elem komplex egységet alkotva képes egy szenzoros környezet működését szimulálni az adatok generálásától kezdve azok hatékony eltárolásáig.

Kulcsszavak: *szenzoradatok, adatgenerálás, adattárolás, Python*

1. Bevezetés

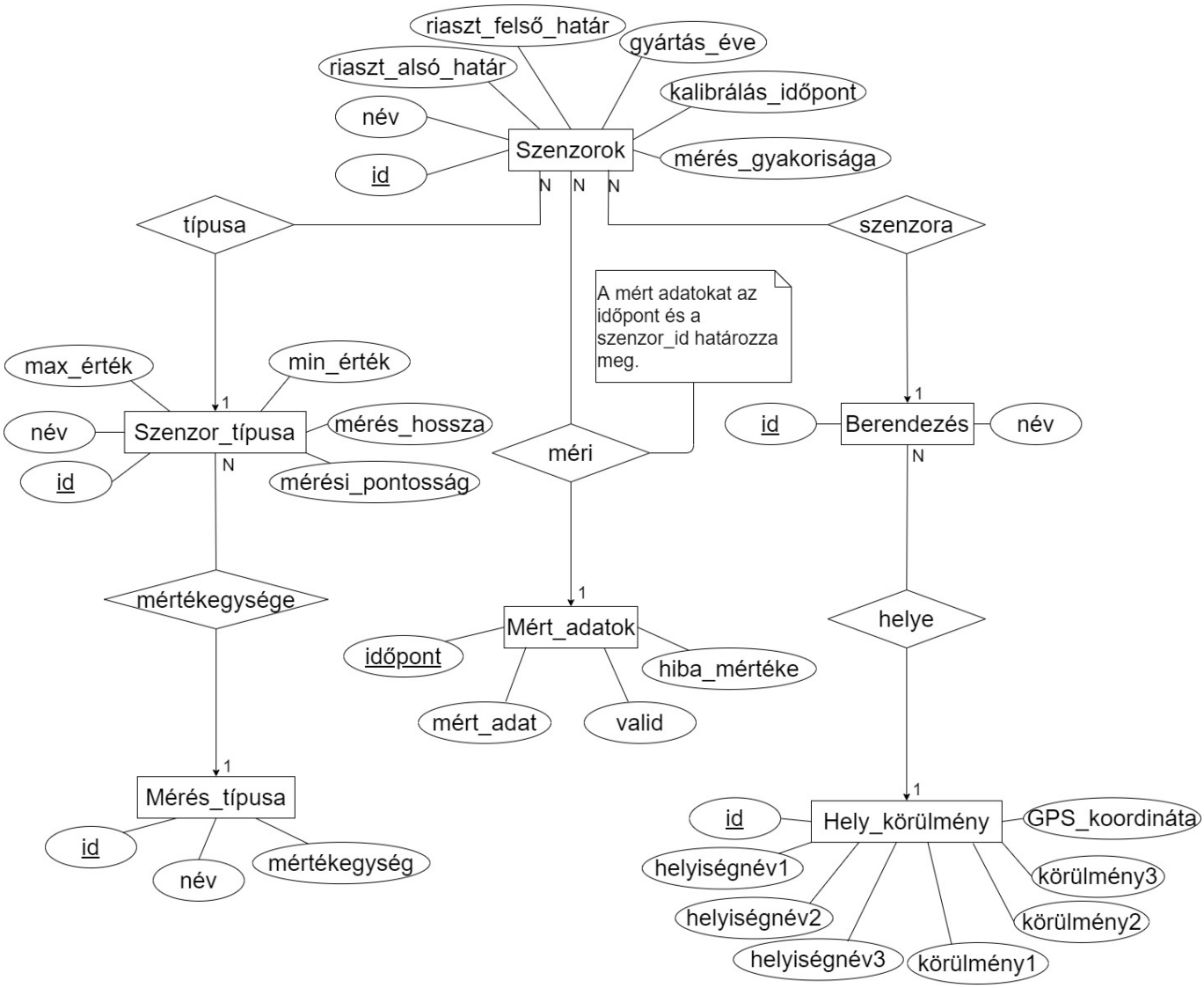
A mai világban egyre több helyen és egyre több alkalommal találkozhatunk szenzorokkal. A szenzor egy olyan eszköz, amely a környezetéből érkező jeleket méri és olyan adattá alakítja, amit az ember vagy gép már képes feldolgozni, értelmezni [1]. A szenzoroknak rengeteg előnye van, sokat közülük már alacsony áron megvásárolhatunk, ezen kívül a rendelkezésre álló szenzorok segítségével szinte bármilyen értéket mérhetünk, a hagyományosnak tekinthető hőmérséklet vagy fogyasztás adatokon túl. Emiatt nem meglepő, hogy az ipar 4.0-hoz kapcsolódóan egyre több cég, gyár szerez be és alkalmaz ilyen eszközöket különböző adatok mérésére [2], hogy ezek segítségével hatékonyabbá tegyék például a termelést, csökkentsék a költségeket, növeljék a nyereséget. Megemlíthetjük azonban az okos otthon-, okos épület-rendszereket is, amelyeknél szintén ilyen érzékelőket használnak, és egy mobilapplikáció segítségével akár a világ másik feléről is vezérelhetjük az otthonunkban lévő különböző okos eszközöket.

Így egyre jelentősebb szerep jut azoknak az alkalmazásoknak, applikációknak, melyek szenzoradatok kezelésével, feldolgozásával, megjelenítésével foglalkoznak, különösen az ipari környezetben, de természetesen igaz ez az élet minden területére. Ugyanakkor ezek az adatok sok esetben szenzitívek lehetnek (termelési, könyvelési, személyes adatok), akár annyira, hogy a kapcsolódó alkalmazás fejlesztéséhez kötődően a tényleges mérési adatok nem átadhatók, legfeljebb a mérési adatoknak a karakterisztikája. A cikkben bemutatunk egy két modulból álló rendszert, ahol a két komponens, az adatgenerátor és a kapcsolódó adatbázis lehetővé teszi, hogy a megfelelő peremfeltételek ismerete mellett (értéktartomány, mérési gyakoriság, mértékegység, pontosság stb.) véletlen adatokat generáljunk, azokat az általános adatbázisban letároljuk. Így lehetővé téve a fejleszteni kívánt applikáció számára az éles adatokhoz nagyon hasonló tesztadatok és tesztadatbázis könnyű, gyors kialakítását.

© ELTE, Informatikai Kar, Savaria Műszaki Intézet, 2022
*Kapcsolattartó: Pödör Zoltán, pz@inf.elte.hu
https://doi.org/10.37775/EIS.2022.1.1

A szenzoradatok kezelési folyamatának egyik alapeleme a hatékony, rugalmas és megbízható eltárolásuk. Erre lehetőséget nyújtanak, mind a relációs, mind a nemrelációs adatbázisok minden előnyükkel és hátrányukkal együtt. Népszerűségüket tekintve a relációs adatbázisok még mindig sokkal elterjedtebbek [3], lekérdezések szempontjából komplexebb műveletek elvégzésére alkalmasak, mint a másik típus. Annak ellenére, hogy a nemrelációs adatbázis felépítése nem kötött, nincsenek táblasémák és kapcsolatok, a szerkezet könnyen módosítható, ha elég jól megtervezzük az adatbázis szerkezetét, akkor a relációs adatbázisok is viszonylag rugalmasak lehetnek [4].

Mindezekből kiindulva munkánk célja tehát egyrészt egy olyan relációs adatbázis tervezése és megvalósítása volt, amely alkalmas bármilyen szenzoros környezetből érkező adat hatékony tárolására, minél rugalmasabban lehetővé téve az adatbázis bővíthetőségét új szenzorok, mérések, mérési körülmények megjelenése esetében is. Másrészt egy olyan adatgeneráló alkalmazást is implementáltunk Python nyelven, ami bizonyos paraméterek megadása mellett ezeknek megfelelő adatsorokat képes generálni és az említett, kapcsolódó adatbázisban letárolni. Így ez a kétkomponensű rendszer alkalmas szenzoradatok kezelését végző applikációk fejlesztési, tesztelési folyamatainak támogatására anélkül, hogy a tényleges, bármilyen szempontból érzékeny mérési adatokat át kellene adni a fejlesztőnek.



1. ábra: Adatbázis felépítése egyed-kapcsolat diagramon megjelenítve

2. Fejlesztett komponensek

2.1. Adatbázis

Az adatbázis tervezése során arra törekedtünk, hogy egy minél általánosabb és rugalmasabb relációs adatbázis szerkezetet alakítsunk ki. Szenzoradatok kapcsán alapvető elvárásként fogalmaztuk meg, hogy az adatbázis bővíthető legyen szenzor típusokkal, konkrét szenzorokkal, mérési típussal, illetve a mérési körülményekkel. Ennek megfelelően kialakítottunk egy univerzális adatbázismodellt, amit az 1. ábrán látható egyed-kapcsolat diagram ír le.

Adatbázis az alábbi hat táblából épül fel (1. ábra):

- **Szenzor_típusa tábla:** a különböző szenzortípusok általános jellemzése, majd az adott típushoz tudunk tényleges egyedi szenzorokat rendelni. Minden szenzortípust egy egész szám azonosít egyértelműen (*id*), emellett egy, a gyakorlatban könnyen értelmezhető nevet is kapnak. Fontos attribútum a *max_érték* és *min_érték*, amelyek az adott szenzortípus által mérhető legnagyobb, illetve legkisebb értéket adják meg. Eltároljuk még a *mérés_hossza* elemben a mérési időt is milliszekundumban, tehát azt, hogy mennyi ideig tart egy konkrét adat mérése, valamint a mérési pontosságot is, például hány tizedesjegy pontossággal történik a mért adat tárolás (1. ábra, *Szenzor_típusa*).
- **Szenzorok tábla:** egy adott típushoz tartozó konkrét szenzor jellemzőit írja le. Minden szenzoregyedet egy egész szám azonosít egyértelműen (*id*), emellett egy nevet is kapnak. Lényeges attribútum a *riaszt_alsó_határ* és *riaszt_felső határ*, ezek azokat a minimális, illetve maximális értékeket jelentik, amelyek alatt, illetve felett már biztosan nem megfelelő a mért érték. Továbbá eltároljuk a szenzoregyed gyártási évét, legutóbbi kalibrálásának időpontját, valamint a mérési mintavétel és tárolás gyakoriságát (*mérés_gyakorisága*) (1. ábra, *Szenzorok*).
- **Mérés_típusa tábla:** a mérés típusának adatait írja le. Minden mérési típust egy egész szám azonosít egyértelműen (*id*), emellett egy nevet is kapnak, vagyis, hogy mit mér a szenzor (például hőmérséklet, nyomás stb.). Eltároljuk még a mérés típusához tartozó mértékegységet is (1. ábra, *Mérés_típusa*).
- **Mért_adatok tábla:** a szenzor által mért adatokat tartalmazza. Minden mérést két attribútum azonosít egyértelműen: időpont és *szenzor_id*. Előbbi a mérés pillanatának időpontja, utóbbi pedig annak a szenzornak az azonosítója, amely a mérést végezte. Alapvető elem a *mért_adat*, ami az adott szenzor által mért tényleges adat értéke. A *valid* attribútum egy bit érték, azaz csak 0 vagy 1 értéket vehet fel. Ha a mért adat a felső és alsó határok között van, azaz megfelelő, akkor értéke 1, ellenkező esetben 0 lesz. Ha a *valid* értéke 0, akkor a *hiba_mértéke* attribútum megadja, hogy mekkora az eltérés a felső/alsó határoktól és milyen irányú ez az eltérés. Ha az alsó határ alatt van az érték, akkor negatív, ha a felső határ felett van, akkor pozitív lesz. A *valid* és a *hiba_mértéke* értékét automatikusan, egy megfelelő trigger segítségével állítjuk be minden egyes új adat beszúrása során (1. ábra, *Mért_adatok*).
- **Berendezés tábla:** a szenzorokat tartalmazó berendezések adatait tartalmazza, hiszen általában a szenzorokat egy adott eszközhöz, berendezéshez kötik, rögzítik. Minden berendezést egy egész szám azonosít egyértelműen (*id*), emellett egy nevet is kapnak (1. ábra, *Berendezés*).
- **Hely_körülmény tábla:** a szenzorok, illetve berendezések helyének és körülményeinek adatait tartalmazza. Minden helyet/körülményt egy egész szám azonosít egyértelműen (*id*), emellett egy nevet is kapnak. A *helyiségnév1*, *helyiségnév2*, *helyiségnév3* attribútumok a helyiségekre vonatkozó adatokat tárolják: például megjeleníthetjük a cég nevét stb. A *GPS_koordináta* a berendezés pontos helyét tárolja hosszúsági és szélességi fokok megadásával. A *körülmény1,*

körülmény2, körülmény3 mezők pedig a különböző körülmények tárolására alkalmasak: például zárt vagy nyílt térben van az adott szenzor, illetve a berendezés folyadékban helyezkedik-e el stb. (1. ábra, *Hely_körülmény*).

Ahhoz, hogy egy táblából hozzáférjünk egy másik tábla adataihoz, kapcsolatokat kell létrehozni közöttük:

- Minden szenzoregyedhez tartozik egy szenzortípus. Másképp megfogalmazva vannak úgynevezett szenzorcsaládok, amelyek rendelkeznek néhány általános jellemzővel, és ezekből az adott rendszerben definiálható több (szenzor) egyed is. Ez egy egy-több típusú kapcsolat, ugyanis egy szenzoregyedhez egy szenzortípus tartozhat, de egy szenzortípusnak több szenzoregyede is lehet. (1. ábra, *típusa*).
- Minden szenzortípushoz hozzákötjük, hogy mit mér és annak mi a mértékegysége. Ez egy egy-több típusú kapcsolat, ugyanis egy szenzortípus egyféle dolgot mérhet, azonban egyféle értéket több szenzortípus is mérhet (1. ábra, *mértékegysége*).
- Minden mért adathoz hozzákapcsoljuk, hogy mely szenzorból kaptuk azt az adatot. Ez egy egy-több típusú kapcsolat, mert egy mért adathoz csak egy szenzor tartozhat, de egy szenzor több adatot is mérhet (1. ábra, *méri*).
- Minden szenzoregyedhez megadjuk, hogy melyik berendezéshez tartozik. Ez egy egy-több típusú kapcsolat, mert egy szenzor csak egy berendezéshez tartozhat, de egy berendezéshez több szenzor is elhelyezhető (1. ábra, *szenzora*).
- Minden berendezéshez megadjuk, hogy hol és milyen körülmények között helyezkedik el. Ez egy egy-több típusú kapcsolat, mert egy berendezés csak egy helyen lehet, de egy helyen több berendezés is előfordulhat (1. ábra, *helye*).

Az adatbázis hatékony és részben automatizált működését a már említett trigger segíti. Segítségével eldöntjük, hogy az éppen beszúrásra kerülő adat határértékeken belül van-e, vagy azokon kívül esik. Ennek megfelelően letároljuk, hogy az adott mérési adat valid-e vagy sem és ezt az információt letároljuk. Majd szükség esetén számítjuk a hiba előjeles mértékét a mért adat és a határ különbségeként, egyébként pedig nullát szúrunk be a hiba mértékéhez. Az adatbázist és a kapcsolódó elemeket MSSQL-Server Management Studio 18.6-os verziójában alakítottuk ki.

2.2. Adatgenerátor

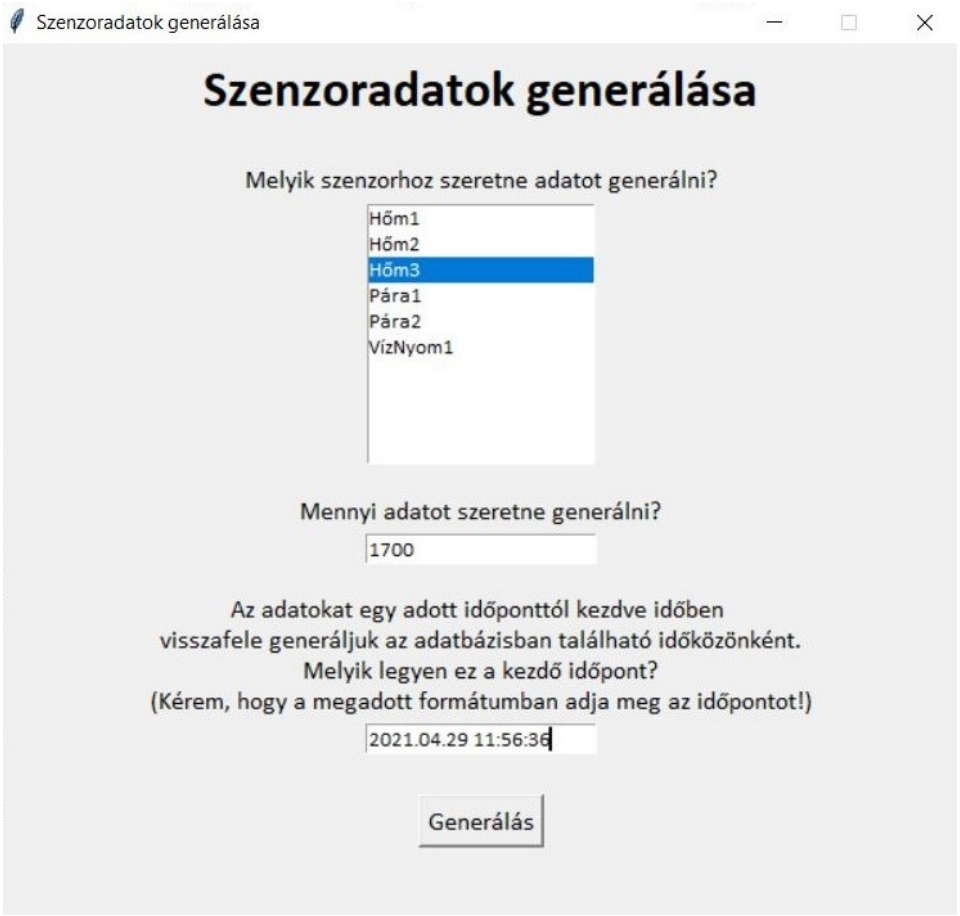
Az adatgenerátor feladata, hogy a beállított paramétereknek megfelelő mesterséges adatokat generáljon, melyeket továbbít az adatbázis felé, ahol azokat tároljuk. Az alkalmazást Python nyelven implementáltuk, munkánk során felhasználtunk egy publikus Github projektet is, a Mandrova-t. Ezt egy koreai fejlesztőcsapat készítette. A Mandrova szó jelentése „make it”, szenzoros kontextusban „make sensor data”, azaz szenzor adat készítése [5]. A Github projekthez a fejlesztők készítettek egy leírást is, illetve példa projekteket, amelyek segítségével mi is tudtunk adatokat generálni az adatbázisunkba. A Mandrova projektben több lehetőség is van adatszimulálásra abból a szempontból, hogy az adatok milyen eloszlással jöjjenek létre. Létezik például normál eloszlás, többváltozós normál eloszlás stb.

A program tervezése során törekedtünk arra, hogy olyan alkalmazást készítsünk, amely felhasználóbarát, tehát egyszerűen tudjunk nagy mennyiségű adatot előállítani és betölteni az adatbázisba. Így készült egy egyszerű felhasználói felület az adatgenerátorhoz, ahol a felhasználónak három dolgot kell megadnia: egy szenzort, amelyet az adatbázisból lekérdezett szenzorok listájából kell kiválasztani, a beszúrni kívánt adatok mennyiségét, és azt az időpontot, amitől kezdve időben visszafele az adatbázisban definiált időközönként beszúrjuk az új értékeket. Ha a felhasználó minden adatot (helyesen) megadott, akkor a háttérben megkezdődik az adatok generálása és beszúrása az adatbázisba.

1. Algoritmus: A *sigma* értékének megválasztása *max_value*, *min_value*, *alarm_max_value*, *alarm_min_value* változók felhasználásával

```
if max_value-min_value ≤ 10 then
    return 2
if max_value-min_value > 10 and max_value-min_value < 90 then
    return 5
if max_value-min_value ≥ 90 and alarm_max_value-alarm_min_value ≤ 10 then
    return 7
return 10
```

Jelen megoldásban minden adatgenerálás során normál eloszlást alkalmaztunk. Itt szükség van egy átlag és egy *sigma* érték megadására. Az átlag jelen megoldásban egy statikus érték, ami a kiválasztott szenzorhoz tartozó felső és alsó riasztási értékeknek az átlaga. A *sigma* a szórás értékét jelenti, azaz, hogy milyen mértékű a szórás az adatok átlagához képest [6]. Ezt az értéket igyekeztünk a jelenlegi megoldásban egyszerűen, de optimálisan megválasztani a felhasználó által megadott szenzor által maximálisan és minimálisan mérhető értékek, illetve a riasztási határok függvényében. Ezt az algoritmust az 1. pszeudokódban írtuk le. Az adatgenerátor közvetlen módon kapcsolódik az MSSQL-Server környezetben kialakított adatbázishoz, lehetővé téve a generált adatok közvetlen továbbítását és tárolását.



2. ábra: Tesztadatok beszúrása az adatbázisba a Hőm3 nevű szenzorhoz az adatgeneráló alkalmazás segítségével

3. Alkalmazás tesztelése, eredmények

Az adatbázis tervezése, megvalósítása és az adatgeneráló alkalmazás implementálása után megkezdődött a tesztelés. Az adatgyűjteményt manuálisan feltöltöttük néhány elemmel. Először létrehoztunk szenzortípusokat (pl. hőmérsékletmérő, páratartalom-mérő, stb.), megadtuk a tulajdonságaikat, mit mérnek, majd konkrét szenzoregyedeket is felvittünk a rendszerbe (pl. *Hőm1*, *Pára2*, *Víznyom1*, stb.). Ezen kívül feltöltöttük néhány adattal a *Berendezés* (pl. YY, ZZ stb.) és a *Hely_körülmény* (pl. A csarnok, D udvar stb.) táblákat is, de a legnagyobb hangsúlyt a mért adatokra fektettük. Ezt a táblát tehát a Python programunkkal töltöttük fel 10 200 darab sorral. Jelenleg hat darab szenzor van az adatbázisban, tehát szenzoronként 1 700 darab adattal rendelkezünk. A 2. ábrán egy példa látható arra, hogy a *Hőm3* nevű szenzorhoz 2021. 04. 29 11:56:36 időponttal kezdve 1 700 darab adatot szúrtunk be az adatbázisba.

A generálás gombra kattintva pár másodperc elteltével megjelent a visszajelző üzenet a képernyőn, miszerint sikeres volt a beszúrás. Ezután megnéztük az adatbázis tartalmát, és valóban megjelent 1 700 adat a *Mért_adatok* táblában, melyek megfeleltek a generálás során beállított paramétereknek. Ekkor a *Mért_adatok* tábla az 1. táblázatban található értékeket tartalmazta.

Ellenőriztük a *valid* és a *hiba_merteke* nevű oszlopok helyességét is, véletlenszerűen kiválasztott elemek manuális ellenőrzésével. A *Hőm3* szenzornak az alsó riasztási határa 15 °C, felső riasztási határa 25 °C. Az 1. táblázatban így már látható, hogy a trigger helyesen működött, megfelelően kezelte a riasztási határokat, és a határoktól való eltérést is helyesen számította ki.

4. Összefoglalás

A szenzorok, szenzoradatok és az ezekhez kapcsolódó alkalmazások, applikációk egyre fontosabb szerepet töltenek be a mindennapokban. Egy-egy ilyen alkalmazás fejlesztése, tesztelése nagyon fontos feladat és sok esetben a tényleges mérési adatok szenzitívek, nem szívesen adják át a fejlesztőknek. Erre nyújtunk megoldást egy adatgenerátor és az ahhoz kapcsolódó adatbázis struktúrával. A cikkben bemutattuk a szenzoros adatok minél rugalmasabb tárolását lehetővé tevő adatbázis felépítését, a táblákat és a közöttük fennálló kapcsolatokat. Továbbá azt az adatgeneráló alkalmazást, mely a megfelelő feltételek mellett generál véletlen adatokat és továbbítja az adatbázisba, ahol azok letárolódnak és felhasználhatók a fejlesztendő alkalmazáshoz, magában a fejlesztési folyamatban, illetve a teszteléséhez.

A felépített rendszer alkalmas bármilyen szenzoros környezetből érkező adat tárolására, legyen szó akár gyárról, akár okos otthonról, hiszen az egyes szenzorokról, körülményekről sokféle információt tárolunk. Elmondható, hogy egy rugalmas adatbázist terveztünk, hiszen használtunk olyan általános elnevezésű attribútumokat is, mint például a *körülmény1*, *körülmény2*, de új oszlopokkal is könnyen bővíthetők a táblák. Az adatgeneráló alkalmazásnak köszönhetően bárki egyszerűen, felhasználóbarát módon és gyorsan tud tesztadatokat generálni, és ezeket később felhasználhatja bármilyen célra, mint például az adatkezelő alkalmazás fejlesztése, tesztelése.

1. táblázat: *Mért_adatok* tábla tartalmának részlete

idopont	*szenzor_id*	*mert_adat*	*valid*	*hiba_merteke*
2021-04-17 16:46:36.000	3	28,27	0	3,27
2021-04-17 16:56:36.000	3	21,31	1	0
...	...	...	...	...
2021-04-29 11:46:36.000	3	25,27	0	0,71
2021-04-29 11:56:36.000	3	22,21	1	0

A bemutatott két komponensű rendszer több továbblépési lehetőséget is tartalmaz, hiszen az adatgenerátor esetében jelenleg normális eloszlással generálódnak az adatok, de mód van egyéb eloszlások megvalósítására is. Az adatbázis komponens esetében is lehetőség van akár nemrelációs adatbázis komponens megvalósítására és integrálására is. Az adatgenerátor paraméterezhetőségének köre is bővíthető, a jelenleg kódba égetett, a generált adatok eloszlását leíró paraméterekkel. Illetve a fenti továbblépéseknek megfelelően a felhasználói felület is kiegészíthető akár egy adatvizualizációs réteggel, támogatva a tárolt adatok könnyen szemrevételezhető megjelenítését.

5. Köszönetnyilvánítás

A jelen publikációban megjelenő kutatások az Innovációs és Technológiai Minisztérium Nemzeti Kutatási, Fejlesztési és Innovációs Alapból nyújtott Kompetencia központok létrehozása - Kutatási infrastruktúra fejlesztése felhívás keretében támogatott, 2019-1.3.1-KK-2019-00011 számú projektben valósultak meg.

6. Irodalomjegyzék

[1] *Fierceelectronics, What is a sensor?*, letöltés dátuma: 2021. 06. 05, url

[2] J. Su, X. Chu, M. Chen, S. Kadry, *Internet-of Things-Assisted Smart Grid Applications in Industry 4.0*, 2021 IOP Conference Series Earth and Environment Science 621(1):012056, CrossRef

[3] *10 Most Used Databases By Developers In 2020*, letöltés dátuma: 2021. 06. 14, url

[4] *IBM, SQL vs. NoSQL Databases: What's the Difference?*, letöltés dátuma: 2021. 06. 06, url

[5] *Github, Mandrova: Sensor Data Generator for Python3*, letöltés dátuma: 2021. 04. 17, url

[6] *Mathsisfun, Standard Deviation and Variance*, letöltés dátuma: 2021. 06. 07, url