Új fejlesztési irányok programozási feladatok megoldáskiértékelő rendszereihez

Kádek Tamás[1], Kósa Márk[2], Orosz Anett[3], Tóth Róbert[4]

{[1]kadek.tamas, [2]kosa.mark}@inf.unideb.hu,
{anett.orosz89, tothr94}@gmail.com
Debreceni Egyetem Informatikai Kar

Absztrakt. A Debreceni Egyetem Informatikai Karán már hosszú évek óta működő ProgCont rendszer használata során szerzett tapasztalatokra alapozva két lehetséges fejlesztési irányt mutatunk be. Egyfelől, a ProgCont rendszerben már jelenleg is megtalálható több száz feladathoz készítünk hasznos megoldási segédleteket. Ezek az útmutatók a felhasználó (tanár vagy diák) szerepkörétől függően más és más információkkal járulnak hozzá egy-egy probléma megoldásához. Másfelől, valós idejű ellenőrzés által nyújtott támogatást szeretnénk elérhetővé tenni az adatbáziskezeléssel kapcsolatos tantárgyak hallgatói számára is. Jelenleg egy olyan webalkalmazás kialakításán dolgozunk, amely felületén az oktatók feladatsorokat állíthatnak össze, a hallgatók pedig feltölthetik a rájuk elkészített megoldásokat. Egy olyan rendszert szeretnénk megalkotni, amely különféle SQL utasítások ellenőrzését támogatja. Reményeink szerint a fejlesztésünk eredménye nemcsak az egyetemi hallgatók otthoni gyakorlását segítheti, hanem egy olyan órai számonkérések lebonyolítására is alkalmas környezet jöhet létre, amely a közoktatásban is segítheti az érettségi vizsgára készülő diákokat. Cikkünkben vázoljuk ezen fejlesztési irányok koncepcionális terveit és a megvalósításuk aktuális helyzetét.

Kulcsszavak: programozás, adatbáziskezelés, megoldási útmutatók, automatizált értékelés

1. Bevezetés

A Debreceni Egyetem Informatikai Karán az eredetileg programozó versenyek támogatására szánt ProgCont rendszert [1] 2011 óta használjuk arra, hogy megkönnyítsük a programozási ismeretek oktató tantárgyak számonkéréseinek lebonyolítását, és hogy segítsük a hallgatók felkészülését. Az eltelt évek alatt felgyülemlett tapasztalatokra alapozva [2, 3] folyamatosan merülnek fel fejlesztési igények akár a rendszer funkcióinak bővítésére, akár a ProgCont alkalmazáshoz hasonló, az oktatást segítő rendszerek kialakítására. Ebben a tanulmányunkban a legújabb fejlesztési irányokból mutatunk be egyet-egyet.

2. Megoldási segédletek

A ProgCont rendszert elsődlegesen programozási versenyek lebonyolítására terveztük. A rendszer elsődleges feladata az volt, hogy automatikusan és objektíven értékelje a programozási feladatokra beérkező megoldásokat. A programozási versenyek feladatainak megoldása alatt egy-egy forráskódot értünk. A lefordított forráskódot különböző teszteseteken keresztül értékeljük, a legegyszerűbb esetben úgy, hogy megvizsgáljuk, hogy a beküldött program a tesztesethez tartozó bemeneten futtatva a megfelelő kimenetet állítja-e elő. A rendszer előnye, hogy a beküldött forráskódok értékeléséhez emberi beavatkozásra nincs szükség, és így az objektív értékelésre vonatkozó követelmény is teljesül.

A rendszert a programozási nyelvekkel kapcsolatos zárthelyi dolgozatok értékelésénél is bevetettük két lényeges előnyt kihasználva:

- a zárthelyi dolgozat során a rendszer az automatizált értékelési folyamatnak köszönhetően közel azonnali visszajelzést produkál a dolgozatot írók felé, akiknek így lehetősége adódik a javításra,

- a zárthelyi dolgozatra való otthoni felkészülés során hasonló feltételekkel gyakorolhat és ellenőrizheti tudását a hallgató.

Természetesen az oktató feladata is módosul, értékelnie a beküldött feladatmegoldásokat nem kell, ugyanakkor a feladatokhoz tartozó teszteseteket körültekintően kell előre összeállítani. Reményeink szerint, abban az esetben, ha már elegendő számú feladat áll rendelkezésre a szoftverben, az oktató feladata korlátozódhat arra, hogy kiválassza a számonkérésre szánt feladatokat. Bár a ProgCont rendszer már most is több mint 1300 feladat leírását és teszteseteit tartalmazza, ebben a tekintetben jelenleg még nem tekinthetjük ideálisnak a helyzetet.

Az otthoni felkészülés rendkívül fontos része a programozási nyelvekkel kapcsolatos tantárgyak követelményeinek. A jó programozási technikák csak gyakorlással sajátíthatóak el. Ebben a ProgCont rendszer mindenképp előrelépést hozott, bár ez korán sem nevezhető elegendőnek. A rendszer visszajelzései ugyanis rendkívül szűkszavúak, a hibás programok kijavításának módjáról nem kap információt a próbálkozó. E tekintetben sokat segít, ha a hallgatónak lehetősége van saját programjának kimenetét összehasonlítani a helyes eredménnyel, és ezt nem csupán a feladathoz közzétett teszteseten, hanem tetszőleges – akár saját maga által összeállított – bemenetet feltételezve megteheti. Erre a célra „Informatikai versenyfeladatok" címmel készítettünk olyan elektronikus tananyagot [2], ahol egyes feladatokhoz a helyes kimenet előállítása is kérhető.

Egy-egy programozási nyelv elsajátítása természetesen még nem elegendő ahhoz, hogy sikerrel oldjunk meg programozási feladatokat. Ehhez a megoldás menetének végiggondolására, a megfelelő algoritmus megalkotására van szükség. Hiába a korlátlan számú tesztelési lehetőség és az output generátor, ha a megoldás menetét – a problémamegoldó algoritmus lépéseit – nem tudjuk meghatározni. Ehhez nyújthat segítséget egy-egy jó tanács vagy a problémához mellékelt útmutatás.

Úgy gondoljuk, sokat segíthetne, ha a feladatokhoz algoritmikus és nyelvspecifikus megközelítésű magyarázatok készülnének, amelyek nemcsak a diákok, hanem az őket felkészítő tanárok számára is iránymutatóak lehetnek. Jelentős mértékben segítené programozási képességeiket, készségeiket fejlesztését az is, ha elolvashatják az egyes feladatokhoz tartozó megoldási segédleteket, amelyek részletes leírást adnak az egyes problémák levezetésével és megoldásával kapcsolatban. Az egyes feladatokhoz tartozó leírások akár több megoldási lehetőséget is tartalmazhatnak, mivel számos feladatnál többféle úton, többféle megoldási technikával is el lehet jutni a helyes eredményhez.

A feladatokhoz készített magyarázatok tárolására a ProgCont rendszer feladatleírásaihoz használt formátumot vettük át. A szempontok mindkét esetben ugyanazok voltak:

- Lehetőleg már létező szabványos formátum legyen, amelynek elsajátítása nem igényel különösebb energiabefektetést. Ebből a szempontból megfelelőnek a HTML vagy DocBook tűnt.
- Tekintettel arra, hogy különféle kimeneteket szeretnénk előállítani a weboldalaktól kezdve egészen a nyomtatott könyvig vagy füzetekig, számítógéppel könnyedén feldolgozhatónak kell lennie. Ezen követelménynek például az XML vagy JSON felelne meg leginkább.
- Lehetőséget kell biztosítani arra, hogy a leírás több – jellemzően magyar és angol – nyelven is előállítható legyen, ugyanakkor az egyszerű karbantartás érdekében a nyelvtől független részek, például a minta bemenet és kimenet többszörözése kerülendő.
- Lehetőséget kell biztosítani magyarázó ábrák és képek beszerkesztésére.

A fenti szempontokat mérlegelve, a ProgCont rendszer feladatleírására az XHTML 1.0 Strict [3] formátumot választottuk, ahol az XML dokumentum gyökerének egy `div` elemnek kell lennie. A többnyelvűséget a feldolgozó programok a szabványos `xml:lang` attribútum használatával valósítják meg, ahol az `xml:lang` attribútummal nem ellátott elemek minden nyelvhez tartozó kimenetben

megjelennek. A képek és az ábrák külső hivatkozásként szerkeszthetőek be a HTML nyelvben megszokott `img` tag felhasználásával. A speciális elemeket, például a minta bemenetet vagy kimenetet `class` attribútumokkal azonosítjuk be.

Példa:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE div PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- https://progcont.hu/progcont/proglabor-c/?pid=300233 -->
<div>
<h1 xml:lang="hu">Képlet</h1>
<p xml:lang="hu">
Készítsen olyan programot, amely meghatározza az alábbi kifejezés értékét!
</p>
<center>
<img src="expr.png"/>
</center>
<h2 xml:lang="hu">Példa bemenet</h2>
...
<pre class="download">1
2
3
4
5</pre>
...
</div>
```

3. Az SQL nyelv gyakorlását támogató fejlesztések

3.1. Az alapötlet

A 2017/18-as tanév őszi félévében merült fel az a gondolat, hogy az online megoldáskiértékelő rendszerek előnyeit tegyük elérhetővé az adatbáziskezeléssel foglalkozó hallgatók és oktatók számára is. Egy olyan rendszer tervezésén és fejlesztésén kezdtünk el dolgozni, amely az SQL nyelvvel kapcsolatos feladatsorok és feladatok kitűzését teszi lehetővé, emellett ki is értékeli a hallgatók rájuk feltöltött megoldásait. Az automatikus kiértékelés megkönnyíti az oktatók számára a megoldások első körű szűrését, hiszen a rendszer elutasítja azokat a beküldéseket, amelyek még a megadott tesztesetek helyes kimenetét sem képesek előállítani. A második körű, szemmel végzett ellenőrzés is könnyebb, ha a megoldások egy adatbázisból kényelmesen visszakereshetők, rendezhetők és szűrhetők, esetleg még eszközöket is biztosít a rendszer az elemzésükhöz. A fejlesztés során igyekszünk figyelembe venni a ProgCont rendszer több éves üzemeltetése során gyűjtött hallgatói és oktatói tapasztalatokat, továbbá alkalmazkodni az adatbáziskezelés sajátos igényeihez. A rendszer fejlesztése agilis szemlélettel, inkrementálisan zajlik. Sikeresen lebonyolítottunk több gyakorló órát és egy zárthelyi dolgozatot az első prototípus használatával, így a felmerült hallgatói észrevételek és oktatói tapasztalatok felhasználhatók a további fejlesztés során.

A weben több olyan környezet is elérhető, amely felületén keresztül SQL utasításokat hajthatunk végre [6, 7]. A célunk ugyanakkor egy olyan tanulást segítő platform elkészítése, amely az előbbi szolgáltatáson túl – a ProgCont rendszer mintájára – alkalmas feladatsorok közzétételére, a rájuk érkező megoldások ellenőrzésére, továbbá képes ellátni az oldal használatához és működtetéséhez szükséges hallgatói, oktatói és adminisztrátori funkciókat. A készülő alkalmazás az *SQLe* (*Smart Query Learning*

Enviroment) nevet kapta, jelezve, hogy távlati terveink túlmutatnak egy egyszerű kiértékelő rendszer megvalósításán.

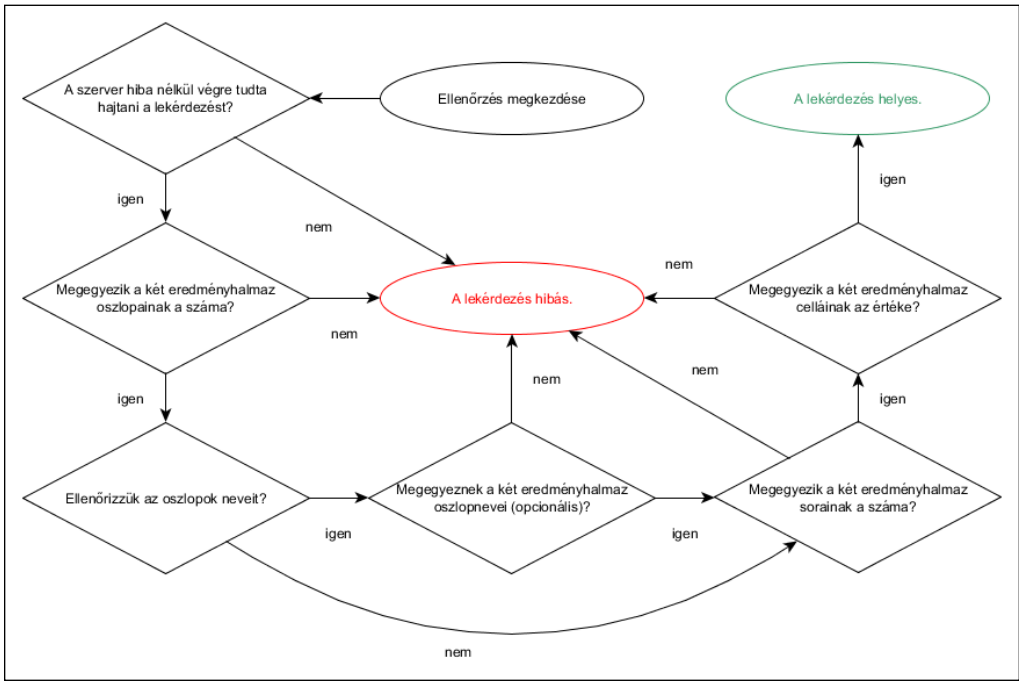
3.2. A megoldások ellenőrzése

A fejlesztésünk középpontjában egy olyan szolgáltatás elkészítése áll, amely képes eldönteni egy adott feladatra készített SQL nyelvű megoldásról, hogy az helyes-e vagy sem. Szeretnénk, ha az elkészülő alkalmazás teljes mértékben támogatná az *Adatbázisrendszerek* tantárgy gyakorlati tematikáját. Ehhez igazodva egy olyan szolgáltatás elkészítését szeretnénk megvalósítani, amely – a tantárgy tematikájához igazodóan – képes az alapvető DML utasítások ellenőrzésére (`SELECT`, `INSERT`, `UPDATE`, `DELETE`), továbbá a táblák és sémák definiálásához kapcsolódó alapvető DDL utasítások kiértékelésére is.

3.2.1. SELECT utasítások kiértékelése

A `SELECT` utasítások ellenőrzése a legegyszerűbb abból a szempontból, hogy a kiértékelendő megoldások nem hajtanak végre módosítást a teszteléshez használt táblá(ko)n. Ebben az esetben azt kell hatékonyan és megbízhatóan ellenőrizni, hogy a lekérdezések kimeneteként előállított eredményhalmaz megegyezik-e azzal az eredményhalmazzal, amelyet megoldásként várunk. Olyan módszert kell alkalmazni, amely egyaránt alkalmas egyszerűbb szelekciókból és projekciókból álló lekérdezések, csoportosító függvények, táblák összekapcsolások és beágyazott lekérdezések ellenőrzésére is.

A `SELECT` utasítások ellenőrzése legegyszerűbben úgy történhet, hogy végrehajtjuk az adatbázis egy – a hallgató által akár ismert – állapotán az elkészült megoldásokat. Emellett a feladatsort összeállító oktató is megad egy olyan `SELECT` lekérdezést, amely a feladat egy helyes megoldása, s lefuttatjuk ezt is. Így előáll egy ellenőrizendő és egy helyes eredményhalmaz, amelyek méretét, értékeit és tulajdonságait össze tudjuk hasonlítani (lásd 1. ábra).



1. ábra: A ellenőrzés lépéseit tartalmazó folyamatábra.

A mérnökinformatikus alapképzés *Adatbázisrendszerek* tantárgyának első zárthelyi dolgozatára elkészített megoldásokon a fenti kiértékelést alkalmaztuk. A hallgatóknak egy olyan adatbázis sémához kapcsolódóan kellett feladatokat megoldaniuk, amelyet már megismertek a gyakorlatok során [8]. Az adatbázis állapotát is ismerték a hallgatók, így tudták azt is, hogy a táblákban csak néhány rekord szerepel.

A dolgozat egyik feladata a következő volt:

A sorozatban kiadott rendszámoknak két része van: a három betűből álló első részt egy három számjegyből álló második rész követi. Listázd ki azon autók rendszámát, amelyek rendszámának számjegyekből álló része megegyezik az 1994. január 3-án állományba vett autó rendszámának számjegyeivel! Az 1994. január 3-án állományba vett autó rendszámát ne tartalmazza a lista!

Akadt olyan hallgató, aki az adatbázis méretét kihasználva manuálisan kikereste, hogy melyik autót vásárolták a feladatban szereplő napon, majd a következő megoldást töltötte fel az alkalmazás felületén keresztül:

```
SELECT rendszam FROM autok
WHERE rendszam LIKE ('%115%') AND rendszam !='CDB-115';
```

Ez az eset demonstrálta azt a gondolatunkat, hogy az elkészített beküldéseket nem elegendő az adatbázis egyetlen – a hallgatók számára akár ismert – állapotán végrehajtani. Kívánatos, hogy a kiértékelés több tesztesettel, az adatbázis eltérő állapotain történjen meg. Ez egyrészt kiszűri az előbb említett, konstansokkal előállított megoldásokat, másrészt pedig pontosan ellenőrizhetővé válnak olyan szelekciók is, amelyek az adatbázis egyetlen állapotán hibás kifejezések segítségével is a megfelelő kimenetet állítják elő.

A felmerült követelményeknek megfelel egy kiértékelés, amely az adatbázis több állapotán is tesztel egy-egy feltöltött megoldást. Különböző tesztesetek határozhatók meg oly módon, hogy az oktató által megadott `INSERT`, `UPDATE` és `DELETE` utasítások végrehajtása után – az eredeti állapotból kiindulva – előállítható az adatbázis egy olyan új állapota, amelyre tesztelhető a felhasználó által feltöltött megoldás. Úgy gondoljuk, hogy ezzel a módszerrel minimálisan nő az oktató által elvégzendő adminisztráció mennyisége, ugyanakkor elő lehet olyan állapotokat állítani, amelyek egy-egy szélsőséges tesztesetnek felelnek meg.

Példa:

Listázd ki városonként a kiosztott prémiumok átlagát egészekre kerekítve! A kimeneten két oszlop szerepeljen, a városok neve és az ottani dolgozók prémiumának átlaga (ebben a sorrendben). A kimeneten csak azok a városok szerepeljenek, ahol ez az összeg eléri a 6000 forintot!

Az adatbázis állapota ebben az esetben olyan, hogy nem tudjuk ellenőrizni a `HAVING` klóz helyes megírását, ugyanis nincs olyan város, ahol a kiosztott prémiumok átlaga pontosan 6000 forint. A felvázolt megoldással könnyen felvehetők olyan bejegyzések, amelyek segítségével lehet akár 5999, 6000, de 6001 forint értékű átlagokat is előállítani.

A több tesztesettel történő ellenőrzés lehetővé teszi azt is, hogy a megadjuk a hallgatóknak az adatbázis általuk ismert állapotából előállítandó kimenetet, továbbá visszajelzést adjunk a tesztesetek sikerességi arányáról is. Úgy gondoljuk, hogy az utóbbi implementáció a beküldések első körű, automatizált tesztelésére megfelelő megoldásként szolgál. Ugyanakkor még mindig előfordulhat az, hogy egy beküldés ugyan előállítja a helyes kimenet(ek)et, de mégsem tekinthető érvényesnek. A rendszer éppen ezért biztosítson lehetőséget arra az oktatónak, hogy érvénytelenítsen egy, az automatikus tesztelésen sikeresen átment megoldást.

3.2.2. Egyéb DML és DDL utasítások ellenőrzése

Az `INSERT`, `UPDATE`, `DELETE`, `CREATE TABLE` és `ALTER TABLE` utasítások ellenőrzése más algoritmust igényel, tekintve, hogy azok végrehajtása megváltoztatja a teszteléshez használt séma vagy

a benne szereplő táblák állapotát. Ebben az esetben is két utasítást vagy utasítássorozatot hajt végre a kiértékelő rendszer: a hallgató beküldését és az oktató helyes megoldását. Mindkét esetben előáll az adatbázis egy-egy új állapota, a feladatunk pedig az, hogy meggyőződjünk azok azonosságáról. Ellenőrizendő, hogy a felhasználó megoldása minden szükséges módosítást elvégez, ugyanakkor nincs nem kívánatos mellékhatása. Ezt legegyszerűbben úgy tudjuk megtenni, hogy az oktató megad egy `SELECT` utasítást, amelyet az új adatbázis két állapotán végrehajt a kiértékelő rendszer. Ha az előállított eredményhalmazok tulajdonságai és értékei megegyeznek, akkor helyes volt a hallgató beküldése, ha eltérnek, akkor pedig hibás. Az ellenőrző utasítás hiányában a séma teljes állapotát is képes ellenőrizni a rendszer. A DML utasítások ellenőrzése tranzakciókban történik, így a folyamat végén egy `ROLLBACK` utasítás segítségével visszaállíthatjuk az adatbázis eredeti állapotát. Természetesen felmerül az igény arra, hogy ezen megoldások ellenőrzése is többféle tesztesettel történjen.

3.3. A platform elemei

Ebben az alfejezetben bemutatjuk a platform eddig megvalósított, illetve a közeljövőben megvalósítandó, már kidolgozott komponenseit. Az alkalmazás felhasználói háromféle jogosultsággal rendelkezhetnek: hallgató, oktató, adminisztrátor.

A kiértékelést ugyan egy szerveroldali Java alkalmazás végzi, de elengedhetetlen a felhasználókkal történő hatékony kapcsolattartás. Célunk, hogy a szerveroldali szolgáltatásokat minél több platformon elérhetővé tegyük, így olyan interfészt (webszolgáltatást) definiáltunk, amelyre egyaránt fejleszthető webes, mobilos vagy asztali alkalmazás is. A következőkben a kiértékelést végző backend és a weben elérhető frontend alkalmazások meglévő és közeljövőben megvalósítandó építő elemeit mutatjuk be.

3.3.1. Adatbázis sémák

Célunk, hogy olyan tesztelő szolgáltatást készítsünk el, amely tetszőleges adatbázis sémán képes kiértékelést végezni, ezen felül pedig többféle SQL technológiát is támogat. Az elkészült megoldások végrehajtása JDBC technológia segítségével történik, ez lehetőséget ad nekünk arra, hogy a teszteléshez szükséges JDBC-kapcsolat birtokában minden SQL verziót és adatbázis sémát támogassunk. A kapcsolatlétesítéshez szükséges adatokat az alkalmazás adatbázisában rögzítik az adminisztrátorok, ezután pedig az oktatók szabadon válogathatnak közülük egy-egy feladatsor létrehozásakor. Az adatbázis sémákhoz megadhatók szerkezeti diagramok, továbbá a nézetek, a táblák és a közöttük lévő kapcsolatok létrehozásához, valamint az adatok beszúrásához szükséges szkriptek is elérhetővé tehetők az oktatók és hallgatók számára.

3.3.2. Feladatsorok és feladatok

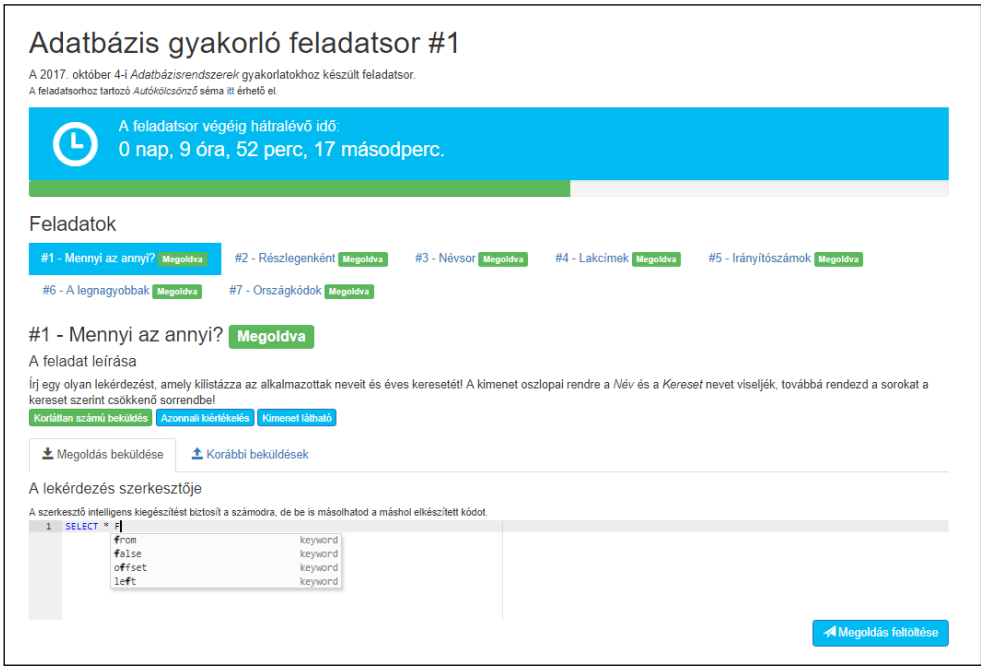
Az alkalmazás felületén különböző feladatsorok kerülnek kitűzésre, amelyekhez egy vagy több feladat tartozik. Ezek lehetnek órai vagy órán kívüli gyakorló feladatsorok, de akár zárthelyi dolgozatok is.

Az oktatói jogosultsággal rendelkező felhasználók létrehozhatnak új feladatsorokat, majd szerkeszthetik és adminisztrálhatják azokat. A feladatsorok testre szabhatók a számos különböző beállítási lehetőség segítségével:

- kiválasztható a feladatsorhoz tartozó adatbázis séma (kötelező);
- szerkeszthető a feladatsor címe, leírása, megadható a feladatsor elérhetőségének kezdete és vége;
- beállítható a feladatsor láthatósága, megadható vagy generálható a hallgatói hozzáféréshez szükséges kulcs;
- megadhatók a beküldések kiértékelésének, tárolásának, illetve az általuk előállított kimenetek láthatóságának beállításai;
- megjeleníthető a hallgatók számára a feladatsor végéhez tartozó visszaszámláló.

Az oktatók a feladatsorok létrehozásán és szerkesztésén túl megtekinthetik a feladataikra érkezett megoldásokat, kiértékelhetik, újraértékelhetik és érvénytelenné nyilváníthatják azokat, továbbá eredménytáblák is segítenek számukra a megoldások összesítésében. A feladatsorokhoz szöveges feladatok tartoznak. Az oktató tetszőlegesen fogalmazhatja meg a feladatok szövegét, majd megadhatja azok teszteseteit illetve az általa elkészített helyes megoldást is. Beállíthatja a feladatok láthatóságát, a kiértékelés módját, bekapcsolhatja az oszlopnevek ellenőrzését, illetve korlátozhatja a feladatra feltöltött beküldések számát.

A hallgatói felületre érve a felhasználó megtekintheti a feladatsorok listáját. A feladatsorok elérése egy-egy hozzáférési kulcs helyes megadása után lehetséges. A kulcs egyszeri helyes megadása után a felhasználó jogosultságot szerez arra, hogy megtekinthesse és megoldhassa a feladatsorhoz tartozó feladatokat. Tetszőlegesen válogathat a feladatok között, valamint elérheti a feladatsorral kapcsolatos általános információkat is. Az oktatói beállításoknak megfelelően megjelenik egy visszaszámláló is, amely visszajelzést ad a dolgozat végéig hátralévő időről, így segítve a hatékonyabb és tudatosabb időbeosztást és munkaszervezést. Egy feladatnak megtekinthető a pontos leírása, illetve megjelenik néhány, a hozzá tartozó beállításokra utaló címszó. A felhasználó egy intelligens, automatikus kiegészítést nyújtó szerkesztőben készítheti el a feladatra készült megoldását, amelyet ezután feltöltheti (lásd 2. ábra).



2. ábra: Egy gyakorló feladatsor a hallgatói felületen. A felhasználók akár szövegszerkesztő programok használata nélkül is összeállíthatják a kitűzött feladatokra készített megoldásaikat.

A feladatsor és a feladat beállításaitól függően az alábbiak egyike történhet egy megoldás feltöltése után a hallgatói felületen:

- a beküldése bekerül a kiértékelő szolgáltatás várakozási sorába, s az ellenőrzés eredményéről rövidesen visszajelzés érkezik, esetleg az adatbázis ismert állapotából előállított kimenet is megtekinthető;

- a felhasználó értesítést kap arról, hogy valamilyen okból kifolyólag sikertelen volt a megoldás feltöltése, vagy nem jogosult újabb megoldások (valós idejű) tesztelésére;
- értesítést kap a feltöltése sikerességéről, ha a beállításoknak megfelelően nem elérhető valós idejű kiértékelés az adott feladatsorhoz, hanem egy zárthelyi dolgozat végéig korlátlan számú alkalommal frissítheti a megoldását.

A megoldások szerkesztésén és feltöltésén túl a hallgató megtekintheti a feladatra korábban már feltöltött megoldásait és a kiértékelések eredményét. A beállításoktól függően a hallgató is megtekintheti az eredményeit összefoglaló táblázatot, vagy akár feladatsorhoz tartozó valamennyi felhasználó sikerességét tartalmazó eredménytáblát is.

3.3.3. Speciális igények

Oktatói igény mutatkozott arra, hogy a zárthelyi dolgozatok írása közben a rendszer ne végezzen azonnali kiértékelést, hanem a dolgozatra szánt idő lejárta után történjen csak meg a hallgatók beküldéseinek ellenőrzése. Ebben az esetben a rendelkezésre álló időtartam alatt a hallgató korlátlan számú alkalommal tölthet fel megoldást a feladatokra. Ekkor minden feladathoz az utoljára feltöltött megoldás kerül eltárolásra. A megoldások kiértékelése tetszőleges időpontban történhet az oktatói felületről elindítva.

Ugyancsak a zárthelyi dolgozatok lebonyolításához kapcsolódóan merült fel az az igény, hogy naplózható legyen a hallgatók IP címe, illetve korlátozható legyen a hozzáférés a címek alapján. Jelenleg naplózásra kerül minden hozzáférés a webszerveren, továbbá beállítható, hogy a rendszer csak egy IP címről fogadjon el megoldásokat (és a többit nyilvánítsa érvénytelennek), vagy maga a feladatsor megtekintése is IP címhez kötött legyen.

4. Összefoglalása

A megoldási útmutatók elkészítése és az SQL nyelv tanulását, gyakorlását és számonkérését támogató platform fejlesztése ugyan még folyamatban van, de az utóbbi alkalmazás prototípusait már teszteltük egyetemi laborgyakorlatokon. A tapasztalataink és a visszajelzések alapján igyekszünk a felmerülő hiányosságokat pótolni, illetve az újabb ötleteket megvalósítani. Az eddig megvalósított és a tervezett fejlesztések lehetővé teszik azt, hogy a közoktatásban is használható megoldásokat kínáljunk.

Köszönetnyilvánítás

A kutatást az „Integrált kutatói utánpótlás-képzési program az informatika és számítástudomány diszciplináris területein” (EFOP-3.6.3-VEKOP-16-2017-00002) című projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Irodalom

1. *ProgCont feladatkiértékelő rendszer*, https://progcont.hu (utoljára megtekintve: 2017.11.05.)
2. Kádek Tamás, Kósa Márk, Pánovics János: *Programozó versenyek támogatása webes alkalmazással*, In: Biró Károly-Ágoston, Sebestyén-Pál György (szerk.): ENELKO 2011 SZÁMOKT2011: XII Nemzetközi Energetika-Elektrotechnika Konferencia; XXI Nemzetközi Számítástechnika és Oktatás Konferencia. 318 p. Konferencia helye, ideje: Kolozsvár, Románia, 2011.10.06–2011.10.09. Kolozsvár: Erdélyi Magyar Műszaki Tudományos Társaság (EMT), 2011. pp. 184–187.
3. Kádek Tamás, Kósa Márk, Pánovics János: *A ProgCont szoftverrel támogatott programozó versenyek tapasztalatai*, In: Veronika Stoffová (szerk.): New Technologies in Science, Research and Education. Konferencia helye, ideje: Komárno, Szlovákia, 2012.09.10–2012.09.13. Komárno: Janos Selye University, 2012. pp. 152–157.

4. Kádek Tamás, Kósa Márk, Pánovics János: *Informatikai versenyfeladatok*, https://gyires.inf.unideb.hu/GyBITT/04/ (utoljára megtekintve: 2017.11.05.)
5. XHTML™ 1.0 The Extensible HyperText Markup Language, https://www.w3.org/TR/xhtml1/ (utoljára megtekintve: 2017.11.05.)
6. *The Try-SQL Editor*, https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_in (utoljára megtekintve: 2017.11.05.)
7. *SQL Fiddle*, http://www.sqlfiddle.com/ (utoljára megtekintve: 2017.11.05.)
8. Balogh Judit, Dr. Rutkovszky Edéné: *SQL Példatár*, KLTE Informatikai és Számítóközpont, http://www.kobakbt.hu/jegyzet/SQLpelda/sql_0bev.html (utoljára megtekintve: 2017.11.05.)