LICEUL TEORETIC CAREI
PROFILUL: REAL
SPECIALIZAREA: MATEMATICĂ-INFORMATICĂ

LUCRARE DE ATESTAT

BAZE DE DATE ÎN PROGRAMAREA WEB

COORDONATOR
Prof. Gózner Róbert

ABSOLVENT
Beiland Arnold

2015

NAGYKÁROLYI ELMÉLETI LÍCEUM

Informatika szakvizsga

Adatbázisok a webprogramozásban

Felkészítő tanár
Gózner Róbert

Tanuló
Beiland Arnold

2015

Tartalomjegyzék

1. Adatbázisok

1.1 Bevezető

Az adatbázis adatok rendezett gyűjteménye. A benne szereplő információk gyűjtése és rendezése általában a világ valamely valóságelemének megragadására alkalmas modell alapján történik, oly módon, hogy a rendelkezésre álló adatok könnyen feldolgozhatóak és hozzáférhetőek legyenek. Például modellezhető egy üzlet raktára úgy, hogy ellenőrizhető és frissíthető legyen a termékkészlet, az egyes termékek ára. Így az adatbázisok és adatbázis-kezelő rendszerek gyakran alkalmazást nyernek például könyvelőségi, számviteli területeken is.

Az adatbázisok gyakran valamilyen cég adatait is tárolják, szerepük van az ügyfelek (esetleg online) kiszolgálásában, hozzáférést tesznek lehetővé bizonyos adatokhoz, kapcsolatot teremtenek az ügyfél és az adott intézmény között.

Nem hanyagolható el az adatbázis-felhasználás mértéke a weblapok esetében sem, ahol felhasználói adatok, bejegyzések, multimédia anyagok tárolására szolgálnak. Jelen dolgozat ezt szeretné bemutatni egy viszonylag egyszerű példán keresztül.

1.2 Adatbázis-kezelő rendszerek

Az adatbázis-kezelő rendszerek (Database Management Systems–DBMS) olyan szoftvereszközök, amelyek bizonyos modellek alkalmazása által adatbázisokat tartanak fenn, megengedik a felhasználóknak ezek manipulálását bizonyos deklaratív programozási nyelveken, parancssori vagy grafikus felü-

leteken keresztül. Egy adatbázis-kezelőnek biztosítania kell az adatok bevitelét, tárolását, módosítását és lekérdezését.

A mikroprocesszorok, memória, tárolóeszközök és számítógépes hálózatok növekedését követve a gyakorlatban használt adatbázisok és az ezeket kezelő szoftverek jelentős növekedésen mentek keresztül, mind méret, mind pedig összetettség szempontjából. Az adatbázis-kezelő rendszerek fejlődése három nagy korszakra osztható az adatmodell szerkezete szerint: navigációs, relációs és poszt-relációs rendszerek.

A két fő korai navigációs adatmodell a hierarchikus modell(amit az IBM kezdeményezett) és a CODASYL modell(hálózati modell) voltak, az ezeken alapuló adatbázis-kezelők már az 1960-as évek közepén elterjedtek.

A relációs modellt elsőként Edgar F. Codd javasolta 1970-ben. Az adatelérés itt táblázatalapú, inkább tartalom alapján mintsem linkek követése által történik. A számítógépek hardvere csak az 1980-as években vált elég fejletté ahhoz, hogy a relációs rendszerek széles körben elterjedhessenek.

A poszt-relációs rendszerek esetében az objektum-orientált adatmodell, a kulcs-érték alapú adatbázisok, illetve a dokumentum orientált adatbázisok a dominánsak. Ezek létrejöttét általában a relációs modell felhasználásakor megnyilvánuló bizonyos hiányosságok indokolták(például objektum-orientált programozási nyelvben nem célszerű táblázatszerű adatbázisból lekérdezni).

1.3 Relációs adatmodell

A relációs modell táblákon alapul. Az alábbi példában szereplő reláció(táblázat) ügyfeleket ír le: a nevüket, a telefonszámukat és az email-címüket. Minden ügyfélhez egy-egy sorbejegyzés tartozik.

név	telefon	email
Kiss Zoltán	0789654321	kiss.zoltan@fiktiv.ro
Nagy Andrea	0793123456	nagy.andrea@masik.hu
Beke Géza	0744555666	bekegeza@nincs.com
...	...	...

1.3.1 Attribútumok és sorok

A reláció oszlopait az *attribútumok* látják el névvel, az előbbi példában ezek: név, telefon, email. Általában az attribútumok megadják az abban az oszlopban szereplő adatok jelentését. Minden attribútumhoz tartozik egy adattípus, amely megszabja az oszlopban előforduló értékek milyenségét.

A reláció azon sorait, amelyek különböznek az attribútumokból álló fejléc sorától, *soroknak* (tuple) nevezzük. A reláció minden egyes attribútumához tartozik a sorban egy komponens.

Az idők során a relációk többször is változhatnak. A változások egy része várhatóan a reláció soraira fog vonatkozni, mint például új sorok beszúrása, azaz az új ügyfelek adatbázisba vétele, létező sorok megváltoztatása, esetleg sorok törlése. A relációséma(attribútumok összessége) megváltoztatása kevésbé általános, és nagy adatszettek esetén igen költséges művelet(előfordulhat például, hogy több millió sort kell átírni).

1.3.2 Indexek

Az index a táblához kapcsolódó, gyors keresést lehetővé tevő táblázat, amelyet az adatbázis-kezelő rendszer tart fent. Tartalmazza, hogy a tábla rekordjai(sorai) egy vagy több oszlop alapján(például név) sorba rendezve hogyan következnének egymás után. Ez nem jelenti a teljes tábla megismétlését többféle rendezettséggel: az index csak egy mutató, amely hivatkozik a táblára. Az indexek fizikai(implementáláskor megadott) szerkezete általában B-fa, ami a tábla végigjárásánál nagyságrendileg gyorsabb(kisebb komplexitású) keresést tesz lehetővé. Az indexek létrehozása jelentősen növeli az adatbázis hatékonyságát, de méretnövekedést is okoz.

1.3.3 Megszorítások

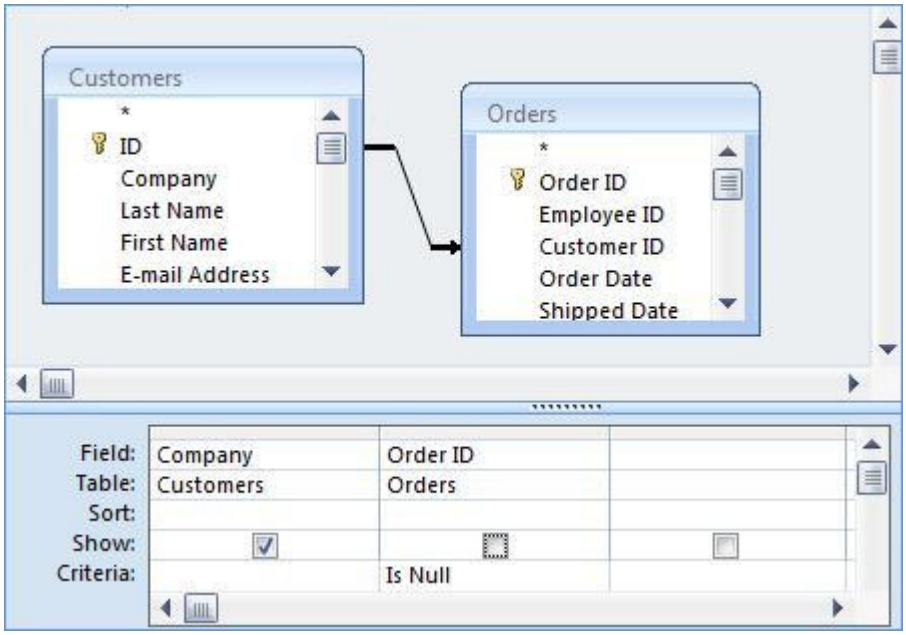
A megszorítások vagy kényszerek(constraitns) a lehetséges adatok halmazát leíró korlátozási szabályok. Sokan a tábla elsődleges kulcsát is egyfajta megszorításnak tekintik, hiszen az elsődleges kulcs maga után von egy egyediségi (UNIQUE) kényszerfeltételt.

A külső kulcs egy olyan megszorítás, amely esetén hogy egy tábla bizonyos oszlopa csak egy másik tábla egy bizonyos oszlopának értékeit veheti fel. Ezzel az előírással az adatbázis integritását, helyességét biztosítjuk, ezért is szokták a külső kulcs megszorításokat integritási megszorításoknak is nevezni (integrity constraint).

1.3.4 Nézetek

A nézet tulajdonképpen egy állandósított lekérdezés: egy vagy több tábla valamely oszlopai egymás mellé rendezve. Ha több tábláról van szó, akkor a nézet az összekapcsolás szabályait is tartalmazza. Mint neve is mutatja, általában arra használjuk, hogy adatainkat egy bizonyos szemszögből, egy bizonyos rendezettséggel mutassa.

Az adatbázisban a nézet fizikailag nem létezik, csak relációs műveletek eredményeként. Bizonyos adatbázis-kezelő rendszerek külön grafikus felületet biztosítanak nézetek szerkesztésére.



ábra 1.1: Nézetszerkesztés *Microsoft Access*ben

1.4 Adatbázis-tervezés

Az adatbázis-tervezés egy folyamat, mely több lépésből tevődik össze. Először az adatbázisban leképezendő rendszert elemzésnek vetjük alá és meghatározzuk a tárolandó adatok körét, azok egymás közötti kapcsolatait és az adatbázissal szemben felmerülő igényeket (fogalmi séma).

Ezután következik a rendszer tervezés, melynek eredménye az adatbázis logikai modellje. Végül fizikai szinten képezzük le a logikai adatbázis modellt az alkalmazott szoftver és hardver függvényében.

Az elemzés egyik lépése a funkcionális függőségek megkeresése. Azt mondjuk, hogy funkcionális kapcsolat áll fenn bizonyos adatok között, ha egy vagy több adat konkrét értékéből más adatok egyértelműen következnek. Például a személyi szám és a név között funkcionális kapcsolat áll fenn, mivel minden embernek különböző személyi száma van.

Szintén fontos feladat a redundanciák megszüntetése. Redundanciáról akkor beszélünk, ha valamely tényt vagy a többi adatból levezethető mennyiséget ismételten (többszörösen) tároljuk az adatbázisban. A redundancia, a szükségtelen tároló terület lefoglalása mellett, komplikált adatbázis frissítési és karbantartási műveletekhez vezet, melyek könnyen az adatbázis inkonzisztenciáját okozhatják. Egy adatbázis akkor inkonzisztens, ha egymásnak ellentmondó tényeket tartalmaz.

A reláció elmélet módszereket tartalmaz a redundancia megszüntetésére, az úgynevezett normál formák segítségével. Egy reláció első normál formában van, ha minden attribútuma egyszerű, nem összetett adat. A második normál forma feltétele, hogy a reláció minden nem elsődleges attribútuma teljes funkcionális függőségben legyen az összes relációkulccsal.

Íme egy reláció első normál formában:

Diákazonosító	Időpont	Terem	Előadás	Férőhely
C105	08:00	01	Analízis	150
C106	09:00	02	Algebra	100
C107	10:00	01	Mitológia	150
...	...	...	...	...

Figyeljük most meg, hogy az előbbi táblázatban fölöslegesen szerepel többször egy adott terem férőhelyeinek száma, tehát redundanciával találkozunk. Ez megszűnik, ha az előbbi relációt szétbontjuk, megszüntetjük a redundanciát és az adatszerkezetet második normál formára hozzuk:

Diák-azonosító	Időpont	Terem	Előadás
C105	08:00	01	Analízis
C106	09:00	02	Algebra
C107	10:00	01	Mitológia
...	...	...	...

Terem	Férőhely
01	150
02	100
...	...

Elkerültük az adatismétlődést, ezzel megelőztük azon problémákat, amelyek egy adott teremben a férőhelyek számának megváltozásakor merülnének fel(most már nem kellene az összes előfordulást kicserélni).

Nyilvánvaló tehát, hogy a tárolni kívánt adatok elemzése, az adatbázis tudatos tervezése és elméleti módszerekkel való ellenőrzése nem csak elvi érték, hanem a gyakorlatban is nagyon fontos lépés. Értelmezettek az adatbázis-elméletben magasabb rendű normál formák is, ezek tárgyalásával viszont jelen dolgozat nem foglalkozik.

2. Webprogramozás

A webprogramozás egy gyűjtőneve mindazon fejlesztési tevékenységnek, amelyet egy weblap létrehozása, vagy egy webszerver működtetése céljából végzünk(például szerver oldali szkriptek írása).

A World Wide Web(WWW) széleskörű elterjedése óta a webprogramozás komoly iparággá növekedett, amit azon vállalkozások tápláltak leginkább, amelyek szerették volna termékeiket és szolgáltatásaikat online eladni a vásárlóknak(mint például egy webáruház).

Mivel az idők folyamán egymást követték az új technológiák, az iparág eljutott oda, hogy a webfejlesztők már nagyon sok szoftveralkalmazást webes szolgáltatásként is elkészítenek(gondolhatunk például a Google Earth – maps.google.com párosra). Ez elméletileg komoly előnyökkel jár, mert platformfüggetlenné lehet tenni a szolgáltatásokat. A gyakorlatban azonban a böngészők különbözősége néha nagyon is közrejátszik a fejlesztésben.

A weblapok a kommunikáció, tartalomszolgáltatás és reklám helyévé is váltak, megjelentek a szociális hálók, amelyek több millió felhasználónak nyújtanak szabadidős elfoglaltságot, kapcsolatteremtési lehetőségeket.

Mivel a WWW-hez használt legtöbb hálózati protokoll kliens-szerver alapú, a webfejlesztés általában több részre tagolódik: kliensoldali programozás, szerveroldali programozás, adatbázis-technológiák felhasználása. A különböző szoftverek együttműködéséhez szükség volt a szabványosításra, programozási nyelvek bővítésére.

2.1 HTML

A Hypertext Markup Language(röviden HTML) a weblapkészítés standard jelölőnyelve. HTML elemekből(úgynevezett *tag*ekből, fonetikusan: teg) épül fel. A webböngészők a HTML kódot fordítják le weblapokká, felhasználva esetleg egy stílusjelölő fájlt(például CSS fájlt). A HTML kód tehát a weblap szerkezeti elemeit írja le és a bemutatni kívánt tartalmat foglalja magába, ezen fájlok kiterjesztése általában *.html* vagy *.htm*.

A HTML kód eredete visszanyúlik Tim Berners-Lee fizikushoz, aki egyik cikkében egy Internet alapú hypertext-rendszer felhasználását javasolta a CERN-ben. Az 1990-es években dolgozta ki az első HTML specifikációt, valamint szerver és kliens szoftvert is írt. A HTML nyelv számos fejlődési szakaszon ment keresztül és máig is több standard van használatban. A szabványosításban több szervezetnek is szerepe volt, ilyenek a: World Wide Web Consortium(W3C), International Organization for Standardization(ISO), valamint a Web Hypertext Application Technology Working Group (WHATWG).

A HTML tagek egy hierarchikus rendszert alkotnak. Egy HTML fájl tartalma például a **<hmtl>** és **</html>** tagek közé kerül(A / jel általában egy tag lezárását jelöli). Íme egy egyszerű weblapszerkezet HTML nyelven:

```
1  <html>
2     <head>
3        <title>Weblap címe</title>
4     </head>
5
6     <body>
7
8        <h3>Közepesen nagy fejléc</h3>
9
10       <p>Ez pedig egy <b>rövid</b> bekezdés.</p>
11
12       <p>Ez egy másik bekezdés.</p>
13
14    </body>
15
16 </html>
```

ábra 2.1: Egyszerű weblap HTML-ben

Menjünk végig most az előbbi kódrészleten. A weblap tartalma **<hmtl>** és **</html>** közé kerül, ezt azonban megelőzheti a dokumentumtípus megadása(például HTML5-ben írt kód esetén: **<!DOCTYPE html>**).

A további kód két részre tagolódik: az úgynevezett fejlécre és testre. Ezeknek a **<head>**, **</head>**, illetve a **<body>**, **</body>** a jelölőjük. A fejlécbe kerülnek az oldallal kapcsolatos meta-információk, jelen esetben a cím(**<title>** és **</title>** tagek között), de szerepelhet itt még karakterszett, kulcsszavak a keresőmotorok számára, a készítő neve és elérhetősége, illetve a csatolni kívánt szkriptek és stílusfájlok deklarálása.

A tulajdonképpeni tartalom a **<body>** részbe kerül: szöveges bekezdések, címek, táblázatok, űrlapok, képek vagy más multimédiás tartalom. A ** **, ** ** páros egy bekezdést jelöl. A **<h3>**, **</h3>** egy fejlécszerű címet. Az első bekezdésben pedig a **** és **** közé írt szöveg félkövér karakterekkel jelenik majd meg.

A .html fájlokat közvetlenül is megnyithatjuk egy böngészővel, nincs feltétlenül szükség egy webszerverre csak akkor, ha valamilyen hálózaton keresztül kívánjuk elérhetővé tenni ezeket.

2.2 JavaScript

A JavaScript egy objektum-orientált szkriptnyelv, amelyet szintén a kliensoldali webfejlesztésben használunk, ha a weblap valamilyen eleméhez dinamikus tartalmat szeretnénk adni(például: ha egy felhasználó egy adott gombra kattint, tűnjön el vagy jelenjen meg valamilyen szövegrész).

A JavaScript kód vagy a HTML fájlban vagy külön (jellemzően *.js* kiterjesztésű) szövegfájlban van. A futási környezet itt szintén a böngésző. A JavaScript nyelv már nem csak leíró, hanem rendelkezik az imperatív programozási nyelvek elemeivel: változók, adattípusok, utasítások, szelekciók é ciklusok is megtalálhatóak a szintaxisban.

A JavaScript nyelvhez gyakran írnak különböző programkönyvtárakat, egyik ilyen a *jQuery*, amely sokkal rövidebb kódírás árán teszi lehetővé a bonyolultabb eljárások leírását is. Vonatkozhatunk a weblap elemeire, illetve kérhetünk le adatokat a szerverről is, mindezt háttérben, a felhasználó

megzavarása nélkül.

Íme egy kódrészlet JavaScriptben a jQuery programkönyvtár felhasználásával, mely aszinkron módon egy szöveges adatot kér le a webszervertől, és megjeleníti azt. A megjelenés helye a weblap egyik objektuma, melyet az azonosítójával adunk meg(jobtext). Hiba esetén figyelmeztetjük a felhasználót.

```
7  function getJobText(id){
8     $.ajax({
9        type: "POST",
10       url: "../script/getjobtext.php",
11       data: {session: getCookie('aeval_session'), id: id},
12       success: function(result){
13          var res=result.split('?');
14
15          if(res[0] == "OK"){
16             $('#jobcontent').show();
17             $('#jobtext').html(res[1]);
18          }
19          else{
20             alert("Error occured: " + res[0] + " Try again.");
21             location.reload();
22          }
23       }
24    });
25 }
```

ábra 2.2: JavaScript/jQuery kód

2.3 PHP

Térjünk most a szerverben elvégzendő feladatokra. A weblapok több módon is kérhetnek vagy küldhetnek információkat a webszervernek, ilyenkor általában rövid programok, szkriptek lefutását idézik elő. Egy széles körben elterjedt szerveroldali szkriptnyelv a PHP.

A PHP(Hypertext Preprocessor) szintén teljes értékű programozási nyelv, beágyazható közvetlenül is a HTML kódba. Ilyenkor még a weblap leküldése előtt lefut a kód a szerveren, ennek kimenete veszi át a kód helyét.

A PHP kiválóan alkalmas olyan feladatok elvégzésére, amelyeket a kliensoldali szkriptnyelvek nagyon körülményesen, vagy egyáltalán nem tudnának

elvégezni. Kapcsolatot teremthet például egy adatbázissal, vagy előidézheti a szerveren bizonyos folyamatok lefutását.

Segítségével a látogatókról statisztikát tudunk készíteni, hozzáférve olyan adatokhoz, mint például a böngésző neve és verziószáma vagy a használt operációs rendszer. Ennek akár piackutatási értéke is lehet. Szintén gyakran alkalmazott az űrlapok feldolgozásában, adatok rendszerezésében és ellenőrzésében.

2.4 SQL, MySQL

Az SQL(Structured Query Language) a relációalgebrára épülő deklaratív nyelv(néhány procedurális elemmel), amelyet legfőképp relációs adatbázisrendszerek manipulálására használunk, adatok lekérdezésére, beiktatására, frissítésére, törlésére, vagy akár relációsémák hozzáadására, megváltoztatására is.

A MySQL a világ egyik legtöbbet használt relációs adatbázisrendszere, mely elterjedtségét főleg a weben történő alkalmazásának köszönheti. Mint a neve is mutatja, SQL nyelven hozzáférhető, és a webhez igazodva, szerverszerű megvalósítása van, így fizikailag külön egységen futhat például a webszerver és a MySQL szerver.

A MySQL rendszerek másik hasznos funkciója a felhasználó alapú privilégium-rendszer, azaz az adatok manipulálását bizonyos felhasználók, csak korlátozott módon tehetik meg, a műveletek, amiket elvégezhetnek előre korlátozhatók.

A MySQL rendszerrel való kapcsolatteremtést számos programozási nyelven programkönyvtárak teszik lehetővé. A fentebb említett PHP nyelvhez például két olyan programozási interfész is tartozik, amely széleskörű használatnak örvend, ezek a MySQLi és a PDO.

Példának nézzünk egy PHP kódrészletet, amely csatlakozik egy adatbázishoz és bejelentkeztetés céljából adatokat kérdez le a megfelelő táblából:

```php
<?php
   require('data_constants.php');

   $db = new mysqli($AEVAL_MYSQL_URL, $AEVAL_MYSQL_USERNAME,
                    $AEVAL_MYSQL_PASSWORD,
                    $AEVAL_MYSQL_DB_NAME);

   if($db->connect_errno) echo "MySql connection error.";
   else{
      $stmt = $db->prepare("SELECT id,password,priv" .
                           "FROM users WHERE username=?");

      $stmt->bind_param('s',$_POST["username"]);
      $stmt->execute();
      $result = $stmt->get_result();

      /* ... */

      $db->close();
   }
?>
```

Észrevehető, hogy a MySQL szerverrel történő interakció egy vagy több kapcsolat megnyitásából, parancsok kiadásából és ezek eredményeinek feldolgozásából, végül pedig a kapcsolat lezárásából áll. Az itt használt MySQLi modul a PHP objektumorientáltságát is képes kihasználni.

Megfigyelhető továbbá, hogy milyen jellegű egy olyan SQL parancs, amelyet adatok lekérdezése céljából adunk ki(SELECT).

3. Aeval Project

3.1 Bevezető

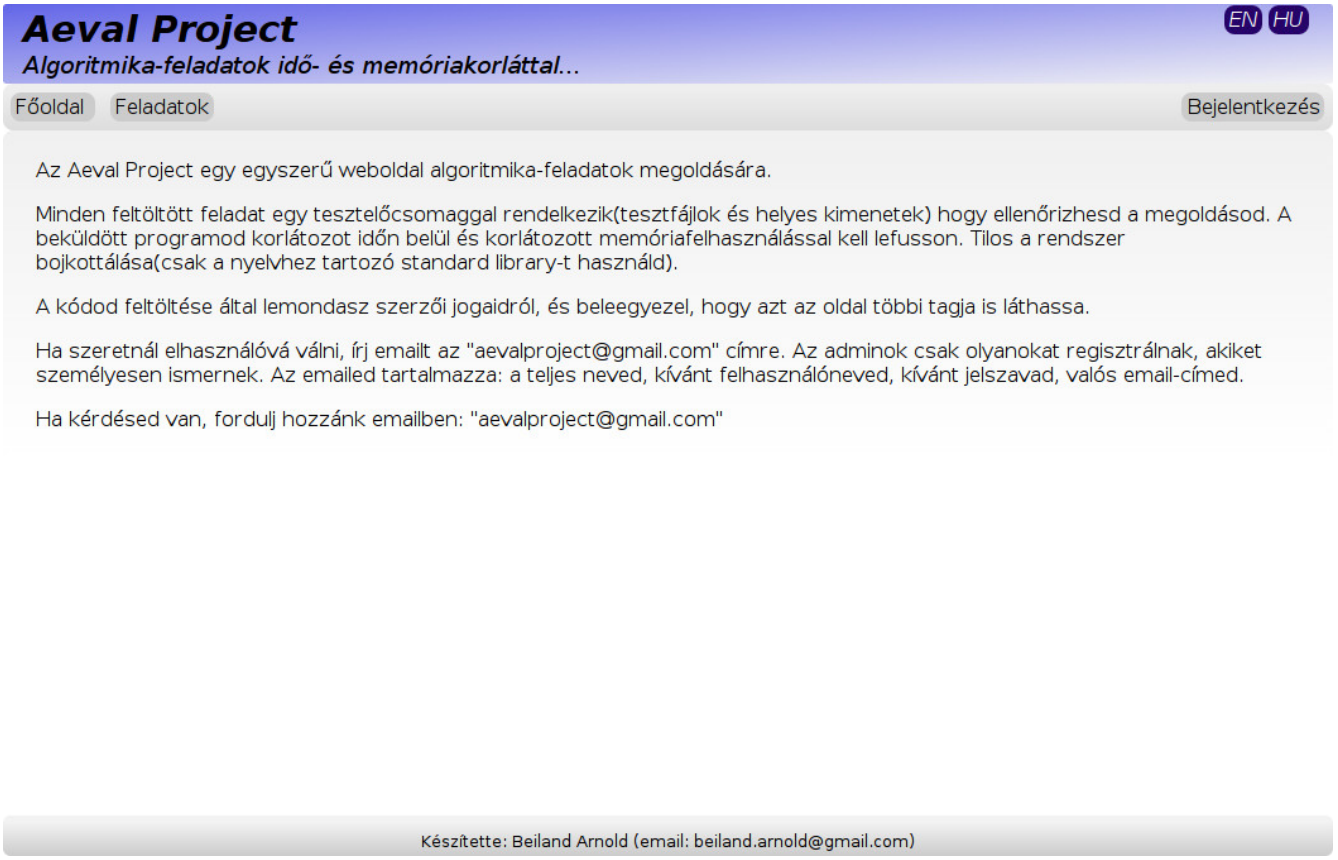
Az Aeval Project egy általam készített weboldal, amely arra hivatott, hogy bizonyos algoritmikaversenyen megjelent feladatokat oldhasson meg a látogató, valamint letesztelhesse a megoldásait.

Ezt egy evaluátor rendszer teszi lehetővé, ami bizonyos tesztfájlok és ezekre adott helyes kimenetek alapján dönti el a beküldött megoldásról, hogy helyes-e, közben pedig figyelembe veszi a futásidőt és a memóriahasználatot is.

A projekt tehát több részre bontható: egy weboldal(kliens és szerveroldali résszel), egy adatbázis, amelyben adatokat tárolunk és egy GNU/Linux operációs rendszeren futó program mely lefordítja, elindítja, illetve elbírálja a beküldött megoldást. A következő részekben ezt a három részt szeretném külön-külön bemutatni. A mellékletben megtalálható a teljes forráskód, ami az oldalt hostoló szerverről készített egyik biztonsági mentés(az oldal a 2015-ös tantárgyverseny-időszakban működtetve volt).

3.2 Weboldal

Az Aeval Project weboldal készítése annak figyelembevételével zajlott, hogy a tartalom, a stílus és az interaktív részek elkülönítése egy könnyen karbantartható, moduláris rendszert eredményez(például a dizájn megváltoztatásához csak a megfelelő .css fájlokat kell átírni, nem kell a HTML kódba nyúlni, és fordítva, új tartalom hozzáadásakor fel lehet használni a már definiált stílusosztályokat).



ábra 3.1: Aeval Project főoldal

A weboldal kétnyelvű, ahol erre lehetőség volt csak a HTML fájlok kerültek megkettőzésre a redundancia elkerülése érdekében. A továbbiakban bemutatom a magyar nyelvű változat esetében az egyes fájlok szerepét(az angol nyelvűben is szinte ugyanúgy épül fel a weblap). Ezek a következők:

http/index.php: Indexoldal, átirányítja a látogatót a http könyvtárból (ahová a domaincím mutat), a http/hu könyvtárba.

http/hu/header.part.html: A bizonyos lapok közös fejlécét tartalmazó fájl, ezek a saját kódjukba ékelik be, amely az Apache webszerveren a Server Side Include(SSI) nevű lehetőségnek köszönhető.

http/hu/footer.part.html: Hasonló az előzőhöz, de ez a közös lábléc.

http/hu/index.html: A főoldal, információkat, linkeket tartalmaz a többi oldalra, a szerver ezt tölti be először, miután az átirányítás az megtörtént a http/hu könyvtárra.

http/hu/login.html: Bejelentkezési oldal, amelyen az adminisztrátorok által regisztrált felhasználók(nyílt regisztráció nincs, mert az oldal zártkörüen működött) bejelentkezhetnek, ami a feladatok megtekintéséhez ugyan nem, de megoldások beküldéséhez és előzőek megtekintéséhez már elengedhetetlen.

http/hu/problemlist.php: A feladatok oldala, beékelt PHP kóddal, amely az adatbázisból kilistázza a feladatokat, illetve a bejelentkezett felhasználók esetén ezek eddigi legnagyobb pontszámát, amelyet az egyes feladatok esetén elértek.

http/hu/problem.php: Egy adott feladatot mutat, az átadott paraméterek alapján. Kiírja a feladat szövegét, a vele kapcsolatos korlátokat, a forrást, és a hozzáadás dátumát. Lehetőséget ad a bejelentkezett felhasználóknak, hogy megoldást küldjenek be az adott feladathoz.

http/hu/show_solutions.php: A beküldött megoldások eredményeit listázza ki az adatbázisból, paraméterek segítségével lehet megmondani, hogy melyik feladathoz, illetve melyik felhasználó beküldött megoldásait mutassa.

http/hu/admin.php: Adminisztrációs felület, csak a megfelelő jogosultságú felhasználók férhetnek hozzá. Itt lehet felhasználót hozzáadni, adatokat módosítani, illetve feladatot hozzáadni.

Mindezen fájlok tartalmat kínálnak, ami ugyan nem lenne jól kinéző a megfelelő .css fájlok nélkül. Ezek a következők.

http/css/main.css: A fő stílusfájl, azokat a beállításokat tartalmazza, amelyek a főoldalra, vagy általában a teljes weblapra azonosak. A legtöbb oldal ezt belinkeli, majd szükség esetén felülírja.

http/css/problem.css: A http/hu/problem.php oldalhoz készült stílusfájl, segítségével megoldható, hogy az egyes feladatok szövegei kevesebb helyet foglaljanak az adatbázisban és a feladatok egységesen nézzenek ki, a felhasználó így a tartalomra tud koncentrálni.

http/css/problemlist.css: A feladatlistázás stílusfájlja. Főleg táblázatformázó CSS szabályokat tartalmaz.

http/css/signup.css: A bejelentkezési űrlapot formázza.

http/css/solutions.css: A megoldások kilistázását formázza, főleg a táblázatot és az előugró részt.

http/css/admin.css: Az admin-felülettel kapcsolatos CSS szabályokat tartalmazza.

A következőkben nézzük meg az egyes kliens- és szerveroldali szkriptek szerepét és működési módját. Ezek felelnek azért, hogy az oldalon megjelenő tartalom aktív legyen, létrejöhessen a beléptetőrendszer, valamint a feladatok megoldásának beküldése.

http/script/data_constants.php: Adatkonstansokat tartalmaz, amelyekre a szerveroldali szkripteknek van szükségük. Például az adatbázishozzáféréshez használt felhasználónév és jelszó. A mellékletben néhány nem nyilvános adat helyén *** található.

http/script/jquery-2.1.1.min.js: A jQuery programkönyvtár, mely ingyenesen letölthető és leegyszerűsíti a bizonyos funkcióval rendelkező szkriptek írását.

http/script/main.js: A főoldalhoz, illetve az oldalakhoz közösen tartozó szkripteket tartalmazó fájl. Leírja például a gombok viselkedését(az egér föléjük vitelekor színt váltanak), ellenőrzi a bejelentkezettségi állapotot és annak megfelelően alakítja a weboldal viselkedését.

http/script/login.js: A bejelentkezési oldalhoz tartozó szkript. AJAX, vagyis Asynchronous JavaScript and XML technológiát használva küldi

tovább átlátszó módon a bejelentkező űrlap adatait a *process_login.php* szerveroldali szkriptnek.

http/script/process_login.php: Az előző szerveroldali párja, ellenőrzi az átadott felhasználónevet és jelszót, tárolja az adott bejelentkezési szesszió adatait, valamit beállítja a megfelelő sütiket, amelyek a következő HTTP csomagban jutnak el a böngészőhöz.

http/script/logout.php: Kijelentkezési szerveroldali szkript. A *main.js* által kerül meghívásra, amikor a felhasználó ki akar jelentkezni.

http/script/solutions.js: A megoldások oldalához tartozik a kliensoldalon. Azokat az eseteket kezeli, amikor a felhasználó az egyik beküldött megoldás forráskódjára, vagy elbírálási eredményére kíváncsi. Ezeket a szervertől szintén Ajax alapú kommunikációval kapja meg, meghívja a *getobjtext.php*, illetve *getobjsource.php* szkripteket, amelyek a megfeleló fájl vagy adatbázis-reláció tartalmát visszaszolgáltatják.

http/script/send_solution.php: A beküldött megoldás fogadásáért és az evaluátor elindításáért felelős. ír a megfelelő adattáblába, kezeli a fájlfeltöltést és meghívja az elbíráló programot.

A további szkriptek az adminfelület eszközei(szintén kliens-szerveroldali párok): insertproblem.js, insert_problem.php, signup.js, process_signup.php, usermod.js, process_usermod.php. Mint a nevük is mutatja, feladat beszúrását, felhasználó regisztrálását, illetve felhasználói adatok módosítását teszik lehetővé az erre jogosultaknak.

3.3 Adatbázis

Az Aeval Project-hez tarttozó adatbázis bizonyos adatok rendszerezett tárolását, rendezését és a gyors megkeresését biztosítja, szerkezete igyekszik a redundanciát kerülni(például a felhasználót egy id azonosítja az összes táblában, nem szerepel mindenhol a neve). Az adatbázis szerkezetéről és tartalmáról a mentés a mellékletben, az aeval-dump_03.03.2015.sql fájlban található,

amely azon SQL parancsok és MySQL makrók egymásutánja, amellyel a teljes adatbázist fel lehet építeni(jelen fájlban a nem nyilvános adatok helyét szintén *** veszi át).

A következőkben lássuk, hogy melyik reláció milyen mezőket tartalmaz, valamint ezeknek mi a szerepe:

Felhasználók táblája: users

A mezők: azonosító(id), felhasználónév(username), jelszó(password), emailcím(email), jogosultság(priv). Ezek közül az azonosító az elsődleges kulcs, és egyedi megszorítás valamint index van a felhasználónévre építve. A jogosultság mező azt tárolja, hogy az adott egyed adminisztrátorként, vagy normál felhasználóként tagja az oldalnak.

Feladatok táblája: problems

Ennek mezői: azonosító(id), feladat neve(name), nyelve(lang), a forrása(source), a hozzáadás dátuma(date), a feladat szövege HTML-ben(text), a megszabott időkorlát ms-ban(time) és a megszabott memóriakorlát kB-ban(memory). Az elsődleges kulcs az azonosító, valamint indextábla épül a feladatok neveire. A feladat szövege bizonyos feladatoknál csak egy linket tartalmaz a megfelelő dokumentumfájlra.

Megoldások táblája: jobs

A beküldött megoldásokat tartalmazza. Mezői: azonosító(id), feladat azonosítója(problemid), felhasználó azonosítója(userid), programozási nyelv(lang), elért pontszám(score), valamint az evalúátor visszajelzése HTML-ben(text), amely a kompillálási üzeneteket és az egyes tesztfájlokon a futtatási eredményt közli táblázatos formában.

Bejelentkezések táblája: sessions

A bejelentkezések tábla információkat tárol az aktív bejelentkezési szeszsziókról. Mezői: userid, sessionid, date. A *userid* a felhasználó azonosítója), a *sessionid* az adott bejelentkezés azonosítója(és a reláció elsődleges kulcsa), a *date* pedig a beiktatás dátuma. Erre a táblára azért van szükség, hogy a felhasználó ne mutathassa magát bejelentkezettnek csak akkor, ha valóban be volt jelentkezve.

3.4 Evaluátor

Az evaluátor rendszer a Linux kernel által közölt adatok alapján állapítja meg egy program futásidejét és a felhasznált memóriát. Teszteli továbbá, hogy a beküldött forráskódból fordított program ad-e eredményt egy adott tesztesetre, és helyes-e ez az eredmény. Minden feladathoz tartozik egy szett bemeneti fájl és egy szett helyes kimeneti fájl, ezekkel van összehasonlítva a program kimenete.

Vannak viszont olyan algoritmikafeladatok is az oldalon, amelyek megoldása nem egységes, ilyenkor a fájlok összehasonlítása mellett vagy helyett az adott feladathoz tartozó elbíráló program az adott helyzetnek megfelelően ellenőrzi a megoldás helyességét.

A projekt ezen része az aeval könyvtárban található meg a mellékletben. Ebben két könyvtár foglal helyett: core és userdata.

A userdata könyvtárban további két könyvtár található: sources és testfiles. A sources könyvtárban találhatóak a forráskódok, mindegyik olyan névvel, amivel a neki megfelelő megoldásbejegyzés az adatbázisban szerepelt(a mellékletben ez a könyvtár üres). A testfiles könyvtárban minden feladatnak van egy könyvtára, melynek neve a feladat adatbázisbeli azonosítója(a mellékletben csak az 1-es számú szerepel).

Egy feladat könyvtárában találhatók a bemeneti és kimeneti fájlok, a hozzá tartozó ellenőrző program(grader.cpp), valamint egy leíró fájl(descriptor), amely tartalmazza az összes tesztesetre a megfelelő fájlok neveit és egy pontszám-szorzót, amellyel az elbíráló program visszatérési értéke szorozódik, és úgy adódik majd hozzá a pontszámhoz.

A core könyvtár tartalmazza az evaluátor magvát, az evaluate.sh Bash-szkriptet(amely az előb említett send_solution.php-ből kerül meghívásra), valamint az evaluator.cpp fájlt, amely az időt és memóriát mérő program. Az evaluator.sh szkript megpróbálja kompillálni a beküldött forráskódot, majd siker esetén meghívja a mérőprogramot, amely az előbbit alfolyamatként futtatja és figyeli az erőforrás-felhasználását. Ez minden tesztesetre megtörténik, végül pedig egy HTML kódú táblázat áll össze az eredményekről, eljut az említett PHP szkripthez, ami az adatbázisba iktatja.

A mérőprogram futása közben a /proc fájlrendszerből olvassa ki a szükséges adatokat. Ha jelentős időtúllépés történik, például a program végtelen ciklust tartalmaz, akkor le is állítja azt, ugyanis ennek leszármazottjaként volt meghívva(fork, majd exec rendszermeghívással). Továbbá chroot metódussal az is korlátozva van, hogy a beküldöt program hova tud írni, illetve honnan tud olvasni(a csalások elkerülése végett).

```
void spawn(char* dir){
    pid_t child_pid;
    char name[]="/candidate";
    char* arg_list[]={name,NULL};

    child_pid=fork();

    if(child_pid!=0){ chprocid=child_pid; return;} //parent process
    else{
        chdir(dir);

        int fd=open("/dev/null",O_WRONLY);
        dup2(fd,1);
        dup2(fd,2);


        chroot(dir);

        if(setgid(AEVAL_GID)!=0){
            abort();
        }

        if(setuid(AEVAL_UID)!=0){
            abort();
        }

        execv(name,arg_list);
        abort(); //execv should never return!
    }
}
```

ábra 3.2: Kódrészlet az evaluator.cpp fájlból

4. Irodalomjegyzék

[1] Jeffrey D. Ullman - Jennifer Widom: *Adatbázisrendszerek*, második kiadás, Panem Könyvkiadó Kft. 2009, ISBN 978-963-545-481-5

[2] http://en.wikipedia.org/wiki/Database (letöltve 28.04.2015.)

[3] http://hu.wikipedia.org/wiki/Relációs_adatbázis (letöltve 28.04.2015.)

[4] Siki Zoltán: *Adatbáziskezelés és szervezés*, http://www.agt.bme.hu/szakm/adatb/adatb.htm (letöltve 29.04.2015.)

[5] http://en.wikipedia.org/wiki/Web_development (letöltve 2015.05.01.)

[6] http://en.wikipedia.org/wiki/HTML (letöltve 2015.05.01.)

[7] http://hu.wikipedia.org/wiki/JavaScript (letöltve 2015.05.01.)

[8] http://en.wikipedia.org/wiki/SQL (letöltve 2015.05.04.)