Soós Péter
Pannonhalmi Főapátság

Az Orbis adatbázis webre vitele

Ezen írás a Networkshop 2001 konferenciára készített előadás anyagának szerkesztett és aktualizált változata. 1994–95 óta sok jelentős katolikus egyházi könyvtár anyagának feldolgozását végzik Orbis adatbázis-kezelő programmal. A programban feldolgozott anyag legnagyobb része ún. modern (az egyházi gyűjtemények értelmezésében 1850 után megjelent) könyv. A feldolgozás előrehaladtával szükségessé vált az ezen adatok belső, illetve nyilvános hálózaton keresztüli, platformfüggetlen elérése. A problémára szinte magától kínálkozott webes megoldás, mivel mind intraneten, mind interneten keresztül lehetővé teszi, hogy megfelelő webböngészővel rendelkező rendszerről elérhetőek legyenek az adatok.

Az Orbis DOS alapú program, ezért szükséges az adatok offline átvitele valamilyen webkapcsolatra képes adatbázisba. Ez a gyakorlatban egy adatexportálást jelent az Orbis programból, majd az exportált adatok beolvasását egy SQL alapú relációs adatbázisba. A megoldásban fontos szempont, hogy az alkalmazott szoftverek képesek legyenek az adatbázisban megtalálható összes adat karakterhelyes megjelenítésére, vagyis ebben az esetben a kelet- és nyugat-európai nyelvek karaktereinek, továbbá a görög ábécé betűinek megjelenítésére. További lényeges szempontok: a rendszer rendelkezésre állása, stabilitása és – az egyházi és közgyűjtemények anyagi helyzetét ismerve – a szoftverek ára is, ezért a megvalósításban előnyben részesítettük a szabad szoftvereket.

A fenti szempontokat figyelembe véve a megvalósítás lehetséges kizárólag szabad szoftverek felhasználásával. A rendszer motorja egy Linux operációs rendszeren futó PostgreSQL relációs adatbázis-kezelő, amely a könyvek adatait tárolja Unicode (UTF-8) kódrendszerben. A weben való megjelenítést a széles körben használt – ugyancsak Linuxon működő – Apache webszerver biztosítja egy Perl programnyelven írt CGI program segítségével. Az adatbázis feltöltése és karbantartása szintén Perl programok segítségével történik.

A megvalósított rendszer képes több könyvtár állományát is kezelni, mint az a tesztrendszerben is látható a *http://biblio.osb.hu* címen. Jelenleg a következő három könyvtár Orbisban feldolgozott adatai találhatók meg a tesztadatbázisban: *Pannonhalmi Főapátsági Könyvtár*, *Kalocsai Főszékesegyházi Könyvtár* és a *Sapientia Szerzetesi Hittudományi Főiskola könyvtára*. A szoftvereket a Pannonhalmi Főapátság szerverén fejlesztették ki, és ott is üzemelnek, a HTML felületet a Sapientia Szerzetesi Hittudományi Főiskola munkatársai tervezték.

Az Orbis adatbázis-kezelő

Az Orbis olyan DOS alapú nyilvántartó program, amely alkalmas nagy mennyiségű hosszú szöveges adat vagy kulcsszó és számítógépre vitt kép kombinált nyilvántartására. Nevét az *Orbis Sensualium Pictus* címmel 1658-ban megjelent első európai képes szótárról kapta, amelyet *Jan Amos Comenius* (1592–1670), a Magyarországon és Lengyelországban tanító cseh tudós és pedagógus 1650 és 1654 között a sárospataki kollégiumban írt. Comenius születésének négyszázadik évfordulóján, 1992-ben indult meg az Orbis program fejlesztése számos magyar múzeum és tudományos intézmény összefogásával. A fejlesztés anyagi hátterét az *Országos Katolikus Gyűjteményi Központ* és a *Soros Open Society Institute* biztosította; tervezésében és tesztelésében a Kalocsai Főszékesegyházi Könyvtár és *MTA Művészettörténeti Kutató Intézet*, illetve a program kifejlesztésére és elterjesztésére Soros OSI támogatással alakult *Orbis Alapítvány* vállalta a legfontosabb szerepet.

A programot ma már számos magyar és külföldi múzeum, könyvtár és más intézmény használja gyűjteményeinek nyilvántartásra, emellett sok kutató munkáját könnyíti meg egyéni jegyzetelőprogramként is.

Az adatbázis felépítése

Az Orbis terminológiája a könyvtárosok és egyéb humán területeken dolgozó szakemberek észjárásához igazodik, így az adatbázis-kezelésben megszokott tábla fogalmát átveszi a *fiók*, a rekord fogalmát a *cédula*, a mező fogalmát a *rovat*. A továbbiakban, ha az Orbis adatszerkezetéről lesz szó, ezeket a fogalmakat használjuk.

Az adatbázis alapegysége a cédula. A cédulát előre megszerkesztett rovatok töltik ki. Ezek számát, méreteit, elhelyezkedését és különleges tulajdonságait az adatbázis felépítésekor adjuk meg, de mindez menet közben is változtatható. Az egyik, előre meghatározott (elsődleges) rovatban lévő adat a cédula neve vagy címe, amelynek az adatbázis kezelésében kiemelt szerepe van.

A cédulák egyes rovatai kétféle típusúak lehetnek, attól függően, milyen módon tudunk azokba adatot bevinni.

Az adatbázisban lehetnek a beírható (fix) rovatok, amelyek kitöltése egyszerű begépeléssel történik. Ezeket különféle speciális tulajdonságokkal láthatjuk el, és számítógépre vitt képeket is köthetünk hozzájuk, hogy azokat közvetlenül az adatbázisból jeleníthessük meg. Emellett az Orbis adatbázisaiban lehetnek még kapcsolódó rovatok. Ezek olyan cédulákra utalnak, amelyeket az eredeti cédulához kapcsoltunk: ilyenkor a rovatban a kapcsolódó cédula címét olvashatjuk, és át is léphetünk rá, hogy megtekintsük. A kapcsolódó rovatok hasonlóak a relációs modell idegen kulcsaihoz, de az Orbisban egy „rovathoz" tetszőleges számú kapcsolatot hozhatunk létre, amit a relációs modellben csak kapcsolótábla alkalmazásával tudtam megoldani.

Adatcsere

Adatcserének az Orbisban azt a fajta adatbevitelt nevezzük, amikor strukturált, azaz mezők szerint elrendezett adatokat automatikusan vitetünk be a programmal az adatbázisba. Ilyen módon cserélünk adatokat két Orbis-adatbázis között, ha mezőiket meg tudjuk feleltetni egymásnak, és így emelünk át más adatbázisokba bevitt adatokat is. Az adatcseréhez eszerint mindenekelőtt ezt a strukturált fájlt kell előállítanunk, amelyet cserefájlnak nevezünk. Ez egyszerű szövegfájl, amely attól függően, hogy milyen szerkezetben tartalmazza a kapcsolódó mezőket, kétféle lehet: „okos" és „teljes". A cserefájlt többnyire az eredeti (Orbis vagy más) adatbázis-kezelő program exportfunkciója állítja elő, de kézzel is elkészíthető vagy javítható.

„Okos" cserefájl

A fájl első három sora rendre a cédula és rovat végét és a lábjegyzetet jelző karakter, míg a negyedik a cserefájltípusát jelzi. Az utóbbi „S" karakter „okos", illetve „1" karakter teljes cserefájl esetén.

A további sorok blokkokra tagolódnak, a blokk végét a cserefájl elején cédula vége karakter jelzi. Minden blokk egy cédulát ír le. A blokk első sora az adott cédula cserefájlon belüli egyedi azonosítóját adja meg, amely a cédula típusát, és a cserefájlon belüli egyedi sorszámát tartalmazza, szóközzel elválasztva. A lehetséges cédulatípusokat az adatbázis szerkezetét leíró fájlok határozzák meg. Minden ilyen leíró fájl elején található a cédulatípust megadó kód „Hívójel" néven. Például jelen esetben a könyveket leíró cédulák kódja „ko". A cserefájlban a cédulán belül minden sor egy rovatnak felel meg. A sor a rovat kódjával kezdődik, amely a cédulatípust megadó kódból és a rovat cédulán belüli sorszámából tevődik össze (pl. „ko01"). A rovat kódjával egy sorban van a rovat tartalma, amely vagy szöveg, vagy egy másik cédula cserefájlon belüli azonosítója, attól függően, hogy fix vagy kapcsolódó rovatról van-e szó. A rovat kódját és tartalmát egy szóköz karakter választja el. A rovat végét a cserefájl elején megadott rovat vége karakter jelzi.

Speciális lehetőség, hogy a fix rovatoknál a szövegben a lábjegyzetjelek között hivatkozhatunk más cédulákra is, ami tovább bonyolítja a relációs adatbázisba való átvitelt, illetve a webes megjelenítést.

„Teljes" cserefájl

A „teljes" cserefájl abban különbözik az „okostól", hogy míg a kapcsolódó rovatok megadásakor az utóbbi a cserefájlon belüli utalásokkal egyszerűsíti és rövidíti a fájl szerkezetét, addig a „teljes" változat minden egyes alkalommal kiírja minden kapcsolódó rovat teljes tartalmát.

Miért van mégis szükség „teljes" cserefájlra? Azért, mert bizonyos esetekben jóval könnyebb ilyet gyártani, mint „okosat". Többnyire olyankor, ha más adatbázis-kezelő programból kell adatokat importálni az Orbisba.

A feldolgozandó adatbázis

A feladat a katolikus egyházi könyvtárak modern könyv adatbázisának feldolgozása. Itt hívom fel a figyelmet arra, hogy ez a megvalósítás csak a katolikus egyházi könyvtárakban alkalmazott Orbis adatbázis-szerkezet feldolgozására alkalmas, nem bármilyen szerkezetű Orbis adatbáziséra. Természetesen készíthető lenne ilyen általános célú program is, de erről a rendelkezésre álló idő és az erőforrások szűkössége miatt le kellett mondani.

A feldolgozás első lépése az Orbis adatbázisból egy „okos" cserefájl előállítása, amely megőrzi az adatok közötti kapcsolatokat. Ennek menete az Orbis felhasználói kézikönyvében megtalálható. A gyakorlatban ez úgy működik, hogy az egyes könyvtárak valamelyik dolgozója, vagy más ezzel megbízott személy létrehozza a szükséges cserefájlt, amelyet azután eljuttat a PostgreSQL adatbázis karbantartójához, aki importálja az adatokat a PostgreSQL adatbázisba egy Perl nyelven készített import program segítségével.

A PostgreSQL adatbázisból a webes keresőprogram nyeri majd az adatokat. A bevitel során a feldolgozandó adatbázisban meglevő, de nem használt, illetve a webes megjelenítés szempontjából szükségtelen rovatokat el lehet hagyni, és csak a valóban szükséges mezőkre szorítkozni.

A feldolgozandó adatbázisban a könyvek leírásában a megjegyzésekben előfordulhatnak hivatkozások más könyvekre is, ezeket a hivatkozásokat kezelik a programok.

A webes rendszer tervezése és megvalósítása

A feladat megoldásakor a következő szempontokat kellett figyelembe vennem. Mivel webes megoldásról van szó, természetesen mindenekelőtt szükség van egy megbízható webszerverre. Szükség van továbbá egy stabil relációs adatbázis-kezelőre, amely az adatokat fogja biztosítani a lekérdezőrendszer számára. Mivel a szabványos megoldások alkalmazására törekedtem, SQL alapú rendszert kerestem.

A megvalósításhoz kell még valamilyen programozási nyelv, amelyben a programok készülnek. Itt több lehetséges programnyelv és környezet közül – részben szubjektív okokból – a Perlre esett a választás.

Végül, de nem utolsósorban kell egy operációs rendszer, amelyen a fenti szoftverek futni fognak. Ez az operációs rendszer a Linux, amelyet már évek óta alkalmazunk egyéb feladatok megoldására.

A feladat természetesen megoldható más Unix-szerű operációs rendszeren, más webszerverrel, más adatbázis-kezelővel és más programozási nyelvek használatával is, több-kevesebb változtatással a megoldásban. Sőt elképzelhető olyan megoldás is, hogy a webszerver és az adatbázisszerver nem is ugyanazon a gépen működik, nem is ugyanolyan operációs rendszeren. A szerverfunkciók szétválasztására egyébként a jelenlegi megvalósítás is alkalmas.

Az Apache webszerver

Webszervernek a széles körben használt Apache szervert választottam, mert ez a világon az egyik legkiforrottabb és legrugalmasabb webszerver szoftver. Mutatja ezt az is, hogy – független statisztikák szerint – a világ WWW szervereinek legnagyobb része Apache-t használ. Ez – mivel szabad szoftverről van szó – feltehetően a szoftver minőségével, nem pedig egy cég erős marketingtevékenységével magyarázható.

Az adatbázis-kezelő

A tervezés során az adatbázis-kezelő kiválasztása okozta a legtöbb fejtörést. Két szabad forráskódú adatbázis-kezelő tűnt megfelelőnek: a PostgreSQL *(http://www.postgresql.org)* és a MySQL *(http://www.mysql.com)*. Mindkettő SQL alapú relációs adatbázis-kezelő. Először a PostgreSQL 6.5.3 és a MySQL 3.22.32 verziószámú változatát vizsgáltam, amelyek a tervezés időpontjában a hivatalos stabil változatok voltak. Az összehasonlítás szempontjai a következők voltak.

Képes-e többnyelvű adatokat tárolni?

Mivel a feladatban szereplő Orbis adatbázisban saját, nem szabványos 8 bites kódtábla alapján vannak kódolva a karakterek, ezért célszerű erről a valamilyen szabványos kódrendszerre áttérni, hogy az adatbázis minél rugalmasabban legyen használható más célokra is. A legszűkebb szabványos kódrendszer, amely lefedi az Orbis által használt karakterkészletet, a Unicode kódrendszer

16 bites változata, az UCS-2. A PostgreSQL támogatja ennek a kódrendszernek az ASCII-kompatibilis változatát, az UTF-8-at.

A MySQL nem támogatja a Unicode kódrendszert, továbbá a használt 8 bites kódrendszert is fordításidőben kellett megadni neki. A Unicode-támogatás a feladat szempontjából létfontosságú, ugyanis többféle nyelv karaktereit kell kezelnie a rendszernek. A PostgreSQL esetén fordításidőben megadható, hogy 8 bites vagy ún. multibájt karakterkódolást használunk-e, és az utóbbi esetben a táblák létrehozásakor megadható azok kódrendszere, amely lehet UTF-8 is.

Ugyan megoldható lett volna, hogy az adatbázis az Orbis 8 bites kódrendszerében tartalmazza az adatokat, majd a lekérdezéskor konvertáljuk azokat Unicode-ra, de ez egyrészt rugalmatlan megoldás, másrészt az adatbázis egyéb célra való felhasználhatóságát is erősen korlátozza.

Képes-e tranzakciókezelésre?

További hiányossága volt a MySQL adatbázis-kezelőnek, hogy – ellentétben a PostgreSQL-lel – nem támogatta a tranzakciókezelést. A legújabb verziókban úgy tudom már van valamiféle tranzakciókezelés, de ennek – mivel addigra már elkészült a PostgreSQL alapú megoldás – nem néztem utána. A tranzakciókezelés hiánya a feladat szempontjából kritikus, mivel – mint azt később látni fogjuk – az adatok beolvasása a Orbis cserefájlból több menetben történik. Először a fix mezők beolvasása történik meg, majd a második menetben a hivatkozások feloldása. Amíg be nem fejeződik a hivatkozások feloldása, addig nem lehet az adatokat véglegesíteni, ezért van szükség a tranzakciókezelésre. Minden cserefájl beolvasása egy tranzakción belül történik.

Mennyire teljes SQL-megvalósítás?

Egyik rendszer sem teljes SQL92 megvalósítás – amire nincs is feltétlenül szükség –, és mindkettő tartalmaz saját bővítéseket is, így azt vizsgáltam, hogy a feladat szempontjából szükséges elemek melyikben vannak meg maradéktalanul. A dokumentációk áttekintésekor derült ki, hogy egyik adatbázis-kezelőben sincs implementálva az idegen kulcsok kezelése, vagyis az adatbázis-kezelő nem képes az adatbázis integritásának idegen kulcsok alapján történő ellenőrzésére. Szerencsére időközben megjelent a PostgreSQL 7-es főverziószámú változata, amely már képes az idegen kulcsok kezelésére.

Ezenfelül a PostgreSQL-ben van egy kényelmesen használható hosszkorlátozás nélküli szöveges típus, a text. Ez különösen ilyen nagy mennyiségű, szöveges adat esetén jön jól, mint az Orbis adatbázis.

A fentiek alapján a PostgreSQL 7-es változata mellett döntöttem, amely letölthető az *ftp://ftp.postgresql.org* címről.

A Perl programozási nyelv

A Perl (Practical Extraction and Report Language), mint a neve is mutatja, elsősorban szövegfájlokban található adatok feldolgozására optimalizált programozási nyelv, bár képes bináris adatokkal is dolgozni. Könnyen tanulható nyelv, ezért viszonylag hamar lehet benne működő programokat írni (mint ezt magam is tapasztaltam a feladat megoldása során :-)). További vonzó lehetőségek a nyelv laza típusossága és rugalmas adatszerkezetei, amelyek ilyen jellegű szövegfeldolgozási feladatok esetén nagyon jól kihasználhatók. Ami a legvonzóbb volt – amiért végül is mellette döntöttem –, az a nyelv hatékony mintaillesztési lehetősége. Reguláris kifejezésekkel ugyanis könnyű megoldani mindenféle szövegátalakítást (pl. kódkonverzió, hivatkozások azonosítása).

A legtöbb adatbázis-kezelőhöz – így a PostgreSQL-hez is – létezik Perl modul. Ez lehetővé teszi, hogy beágyazott SQL utasítások segítségével elérjük az adatbázisban tárolt adatokat, továbbá több webprogramozással kapcsolatos modult is kifejlesztettek hozzá.

A feladat megoldásában végül két kiegészítő Perl modult alkalmaztam. Az egyik a PostgreSQL-hez való kapcsolatot biztosító, és annak részeként terjesztett *Pg(3)* modul, amely lehetővé teszi, hogy Perl objektumok segítségével SQL parancsokat adjunk ki, amelyekben használhatunk Perl változókat is, és az eredményeket Perl változókba olvassuk vissza.

A másik a *CGI(3)* modul, amelyet CGI írásához fejlesztettek ki, és az újabb Perl változatok már eleve tartalmazzák. A CGI(3) modul lehetővé teszi, hogy ne kelljen a HTML nyelv szintaxisával bajlódnunk, egyszerű objektumokat és függvényeket biztosít ehhez a feladathoz.

Az adatbázis

Az adatbázis megtervezésénél alapvető szempont volt, hogy nem egy új adatbázis-szerkezetet kell létrehozni, hanem – a lehetőségekhez mérten – az Orbisban meglévő és a feladat szempontjából lényeges adatszerkezetek minél hűbb leképezését adni. Ezért csak azok az adatok kerülhettek be az adatbázisba, amelyek az eredeti adatbázisban is megtalálhatók.

Szerencsére az Orbis objektumai (fiók, cédula, rovat) többé-kevésbé megfeleltethetők a relációs adatbázis-kezelőkben megszokott objektumoknak (tábla, rekord, mező). Van azonban az Orbisnak egy olyan tulajdonsága, ami miatt mégsem lehet teljes megfeleltetést végezni: minden kapcsolódó rovat egyszerre kapcsolódhat több cédulához is. Például egy könyvnek lehet több szerzője is, ekkor az adott könyv szerzőt leíró rovata több cédulához kapcsolódik a személyek fiókban. Emiatt végül úgy alakítottam ki az adatbázis szerkezetét, hogy minden Orbis fióknak megfeleltettem egy PostgreSQL táblát, amely az Orbis fiók fix (nem kapcsolódó) mezőit tartalmazza, továbbá minden kapcsolódó mezőhöz létrehoztam egy kapcsolótáblát, amelynek két mezője van. Az első mező tartalmazza a hívó objektum saját táblájában kapott azonosítóját, míg a második a kapcsolt objektum saját táblájában kapott azonosítóját. Így a kapcsolótáblában minden kapcsolathoz tartozik egy-egy rekord.

Az Orbisban minden mező szöveges, ezért a kialakított PostgreSQL adatbázis mezői *text* típusúak. Ez a típus a PostgreSQL saját típusa, az SQL92 szabványban nincs ilyen. A text típus egy hosszkorlátozás nélküli szövegtípus, ami a feladat megoldásában igen kényelmes lehetőséget jelentett.

Az adatok importja

A cserefájlok beolvasását szintén egy Perl program végzi. A program „okos" cserefájlokat olvas be. A bevitel két menetben történik, az elsőben a program a cserefájlból a fix rovatokat olvassa be, a másodikban pedig a cédulák kapcsolódó rovatainak és megjegyzéseinek hivatkozásait oldja fel és viszi be az SQL adatbázisba. A fix rovatok beolvasása közben történik az adatok konverziója UTF-8 kódrendszerbe.

Lekérdezőfelület

A lekérdezőfelületnek lehetőséget kellett biztosítani internethálózaton keresztüli keresésekre. Erre a legalkalmasabb a Word Wide Web felület. Mivel a platformfüggetlenség fontos szempont, ezért szerveroldali aktív megoldást választottam, vagyis egy szerveroldali program állítja elő az adatbázisban való kereséshez szükséges SQL parancsokat, és a találatok alapján összeállítja a megjelenítendő listát, amit weblapként küld el a böngészőnek. A gyakorlatban ez egy CGI programot jelent, amely a keresőkérdést mint paramétereit kapja meg egy HTML űrlaptól. A CGI elkészítéséhez a Perl CGI(3) modulját használtam fel. A program biztonsági okokból nem végez fájlműveleteket, csak az adatbázisból kérdez le adatokat.

Bár a CGI script formálisan HTML 4.0 lapot generál, törekedtem arra, hogy a HTML nyelv 3.2 változatának megfelelő weblapot állítsak elő, amelyet a legtöbb webböngésző támogat. Így a használható böngészők egyetlen korlátja, hogy támogassák az UTF-8 kódrendszert.

Használat

A keresőrendszer a *http://biblio.osb.hu* címen érhető el. A címet megadva a böngészőnek, az betölti a keresőrendszer nyitólapját, ahonnan az ugrópontokat (linkeket) követve elolvashatjuk a használati utasításokat, vagy eljuthatunk a keresőprogramhoz.

A keresési feltételeket egy HTML űrlapon kell megadni. Lehetőségünk van a könyv címe, szerzője, a könyvvel kapcsolatos személyek, a kiadó neve, a könyv ISBN azonosítója, tárgyszavak és a kiadás éve alapján való keresésre. A könyvvel kapcsolatos személyek jelentik a könyv szerzőit, egyedi közreműködőit, a magánkiadót (ha magánkiadásról van szó) és a könyv által említett személyeket is. A kiadás évére való keresés lehetséges konkrét évként és időintervallumként is.

A keresési feltételek megadása után a „Keres" gombra kattintva indítható a keresés. Ha több mezőt töltünk ki, akkor a találatok halmaza az egyes mezőknek megfelelő találati halmazok közös része lesz. Ha valamelyik szöveges mező mellett bejelöljük a „Töredék" keresést, akkor elég megadni a keresendő szöveg töredékét. Ez akkor hasznos,

ha nem tudjuk pontosan, hogyan vitte fel az adatrögzítő az adott mezőt, vagy csak egy részre emlékszünk a keresendő adatból (pl. csak egy személy vezetéknevére). Ilyenkor nem számít, hogy kis- és nagybetűkkel adjuk meg a keresett töredéket. A töredékként való keresés tulajdonképpen a PostgreSQL által biztosított reguláris kifejezések kis- és nagybetűkre nem érzékeny üzemmódjában történik. Ha ismerjük a programozási nyelvekben használatos reguláris kifejezéseket, úgy azokat is alkalmazhatjuk ebben a keresési módban. Például néhány böngésző esetén problémát tapasztaltunk az ékezetes betűk megadásakor. Ez esetben az ékezetes betűk helyére egy . karaktert írva a keresés végrehajtható, bár lehet, hogy a találati halmaz némileg nagyobb lesz.

- A „Szerző" mezőbe beírt adatot a rendszer keresi az egyedi és testületi szerzők között is.
- A „Cím" mező a könyv címe alapján keres.
- A „Személyek" mező az egyedi szerzők és közreműködők alapján; ha magánkiadásról van szó, akkor a magánkiadó alapján; és az említett személyek alapján, vagyis minden, a könyvvel kapcsolatos személynév alapján keres.
- A „Kiadó" mező a könyv testületi és magánkiadói alapján keres.
- Az „ISBN" mezőbe a keresett könyv ISBN számát lehet beírni, ha tudjuk.
- A „Megjelenés éve" keresési feltételnél vagy a pontos évszámot kell megadnunk az első mezőben, vagy ezt a mezőt üresen hagyva a következő két mezőben egy időintervallum kezdetét és/vagy végét.
- A „Tárgyszó" mező a könyv tárgyszavai alapján keres.

A találatok maximális száma adja meg, hogy a találati halmazból maximum hányat jelenítünk meg. Ennek értéke 1 és 99 között változhat. Erre azért van szükség, mert elképzelhető olyan keresőkérdés, amely több száz vagy több ezer találatot is eredményezhet, és az eredménylista megjelenítése nagyon hosszú időt vehet igénybe, főleg akkor, ha lassú vagy erősen terhelt a hálózat. Tervezzük a rendszer továbbfejlesztését olyan módon, hogy a teljes találati halmaz megjeleníthető legyen, persze több lépésben.

Ha a „Töröl" gombra kattintunk, akkor a böngésző alaphelyzetbe hozza az űrlapot, törölve belőle minden előzőleg beírt adatot.

A találati lista minden elemét elérhetjük a lista tetején levő sorszámára való kattintással. Minden könyv leírása alatt van egy ugrópont a lap tetejére. A lap tetején és alján levő ugrópontok segítségével visszajuthatunk a keresőlapra. Mindezek még karakteres böngészőben (pl. Lynx) is működnek, ilyenkor természetesen kattintás helyett a megfelelő vezérlőbillentyűkkel aktivizálhatjuk a kívánt funkciót.

Ha egy könyv a megjegyzésben vagy mellékletként hivatkozik egy másikra, akkor a hivatkozás HTML ugrópontként jelenik meg, amelyre rákattintva az említett könyv adatai fognak megjelenni.

Böngészők

A rendszer bármilyen UTF-8 kódolást támogató böngészővel használható. Az általam sikeresen tesztelt böngészők:

- magyar nyelvű Microsoft Internet Explorer 5.01 magyar Windows 98 operációs rendszeren,
- magyar nyelvű Microsoft Internet Explorer 4.0 magyar Windows 98 operációs rendszeren,
- magyar nyelvű Netscape Communicator 4.51 magyar Windows 98 operációs rendszeren,
- angol nyelvű Netscape Communicator 4.x Linux operációs rendszeren,
- angol nyelvű Netscape Communicator 6 Linux operációs rendszeren,
- Lynx 2.8.x Linux operációs rendszeren (konzolon és X ablakban is).

Sikertelenül tesztelt böngészők:

- angol nyelvű Microsoft Internet Explorer 4.01 amerikai Windows 95 operációs rendszeren,
- angol nyelvű Netscape Navigator 3.01 magyar Windows 3.1x rendszeren,
- angol nyelvű Netscape Navigator 3.01 magyar Windows 98 operációs rendszeren,
- a KDE 1.1.2 grafikus felület integrált böngészője Linux operációs rendszeren,
- Opera 3.62 magyar Windows 98 operációs rendszeren,
- Opera Technical Preview 2 Linux operációs rendszeren.

Nem tesztelt, de tesztelendő böngészők:

- Netscape Communicator Windows 3.1x rendszeren,
- Microsoft Internet Explorer Windows 3.1x rendszeren,
- Microsoft Internet Explorer a Windows 9x angol nyelvű nemzetközi változatán,
- Netscape Communicator a Windows 9x angol nyelvű nemzetközi változatán,
- Microsoft Internet Explorer a magyar Windows 95 operációs rendszeren,

• más egyéb operációs rendszerek és böngészők.

Teljesítményigény

Nehéz a szükséges hardverteljesítményre általános méretezési irányelveket adni, mivel ez nagyban függ az adott szoftverkonfigurációtól és attól, hogy mekkora forgalma van a szervernek. Alsó korlátot adhat a megoldásban alkalmazott RedHat Linux 6.2, illetve a PostgreSQL 7.0 által támasztott minimális hardverkövetelmény, de azok csak elméleti értékek, a gyakorlatban használhatatlanok. A jelenlegi rendszer egy kétprocesszoros Intel Pentium III 550 100 MHz alapú PC 512 MB memóriával. Ennél a gépnél a a processzor (CPU) sebessége és a memória sávszélessége és mérete jelenti szűk keresztmetszetet, mert adatfeltöltésnél az indexállományok felépítése igényel sok adatmozgatást a memória és a CPU között.

A nagyméretű táblák indexeinek felépítése miatt egy nagyobb cserefájl bevitele hosszú ideig is eltarthat. Egy nagyobb cserefájl (kb. 20–30 000 könyvrekord) beolvasása néhány óráig is eltart a fenti gépen. Például egy Intel Pentium III 450/100 MHz gépen, amely 128 MB memóriával volt felszerelve, egy kb. 30 000 könyvet és összesen kb. 80 000 rekordot tartalmazó cserefájl egyben való bevitele 14–16 óráig tartott.

A két CPU előnye ott mutatkozik, hogy ha az egyik CPU-n fut az adatfeltöltés, a másik közben kiszolgálhatja a kéréseket, vagy mindkettő párhuzamosan szolgálhat ki lekérdezéseket. Sok egyidejű lekérdezés esetén a memória mérete jelentheti a szűk keresztmetszetet.

A jelenleg üzemelő rendszerben kb. 70 000 könyvrekord és az azokhoz kapcsolódó egyéb adatok találhatók, kb. 75 MB helyfoglalással, ami a mai merevlemezek esetén nem túl jelentős.

Karbantarthatóság

Mint már említettem, ez egy tesztrendszer, amelynek megvalósítása során elsősorban a feladat megoldására, megoldhatóságára koncentráltunk. A frissített adatok cserefájlok formájában érkeznek. Ha új adatok vannak a cserefájlban, akkor csak be kell őket tölteni a rendszerbe. Probléma akkor adódhat, ha módosított adatok vannak a cserefájlban. Sajnos a cserefájlokban nincs olyan információ, amely lehetővé teszi a módosított adatok észlelését. Így univerzális megoldásként marad az adott könyvtár adatainak törlése és újra betöltése, többnyire éjszaka.

Mivel a rendszert elsősorban saját használatra készítettük, ezért az üzemeltetés nem jelent problémát, mert rendelkezésünkre áll hozzáértő személyzet, de ettől függetlenül igyekeztünk úgy elkészíteni a programokat, hogy könnyen használhatók legyenek.

Fejlesztési elképzelések

Számos probléma vár még megoldásra, amelyen folyamatosan dolgozunk. Ezek közül a legfontosabbak:

- a teljes találati halmaz több lépésben való megjeleníthetősége,
- a lekérdezésekben a kis- és nagybetűk közti különbségtétel megszüntetése,
- az ékezetes betűk kezelésének javítása,
- a programok teljesítmény szempontjából történő optimalizálása,
- használat során felmerülő egyéb igények beépítése a programokba,
- folyamatos hibajavítás.

Beérkezett: 2001. VII. 5-én.

Könyvtörténeti adatbázis

A *Book History Online* a nyomtatott könyv és a könyvtárak történelmét feltáró folyóiratcikkek és könyvek adatbázisa. Az adatbázis alapját a témával foglalkozó éves bibliográfiai összeállítás képezi, és ehhez hasonlóan a könyvek előállításának, forgalmazásának, megőrzésének, leírásainak és elemzésének szakirodalmát fogja át. Ezeken felül a könyvekkel kapcsolatos művészetekkel, mesterségekkel, technikával és berendezésekkel, valamint a könyvek gazdasági, társadalmi és kulturális környezetével foglalkozó szakirodalmat tartalmaz. A bibliográfiát és az adatbázist több mint 30 ország történészei állítják össze, és 1989 óta a hollandiai Királyi Könyvtár kezeli az immár 25 000 rekordot tartalmazó adatbázist.

További információ: **www.kb.nl**

/Information Retrieval and Library Automation, 37. köt. 1. sz. 2001. p. 9./

(R. P.)