XI. Erdélyi Tudományos Diákköri Konferencia

Adatbázisalapú fájlrendszerek

Szerző
Roth Róbert

Babeş-Bolyai Tudományegyetem
Matematika és Informatika Kar
Informatika szak, 3. évfolyam

Vezető tanár
Dr. Robu Judit

Babeş-Bolyai Tudományegyetem
Matematika és Informatika Kar
Informatikai Rendszerek Tanszék

Kolozsvár
2008. Május 23-24.

Tartalomjegyzék

Bevezető

Képzeljük el, hogy a testvérünk karácsonyi képei közül keresünk egyet. Az óriási kapacitású merevlemezünk nagy részét a képek foglalják el. Milyen egyszerű volna, ha csak beírnánk a testvérünk nevét és a "karácsony" szót, és máris láthatnánk az összes ilyen képet. Ehelyett azonban minden karácsony után felmásolt fotónkat végig kell néznünk, mivel minden képet nem neveztünk el úgy, hogy az esemény, a hely, és a szereplők nevei is szerepeljenek a névben, mert ezek a nevek nagyon hosszúak lennének. Az adatbázisalapú fájlrendszerek éppen erre próbálnak megoldást adni úgy, hogy egy alternatív keresési módot biztosítanak a fájlok közti keresésre, az azokhoz a felhasználó által társított információk alapján való keresést.

Az adatbázisalapú fájlrendszerek nem csak leegyszerűsítenék a keresést, de a háttérben található adatbázis által nagy mértékben fel is gyorsítanák azt. Egy kereséshez egy adatbázisalapú fájlrendszerben nem lenne szükség arra, hogy minden fájlhoz hozzáférjen a keresés közben az operációs rendszer, így a lemezműveletek számát nagymértékben csökkentené a hagyományos keresésekhez képest, és ezáltal merevlemez élettartamát is növelné.

Egy adatbázisalapú fájlrendszert gyakorlati előnyeivel való megismerkedés után sok számítógép-felhasználó használna, kezdve az átlagfelhasználótól egészen a programozókig. Bizonyítékként felhozható a számos hasonló célú próbálkozás, például az Apple MacOS X operációs rendszerébe beépített Spotlight[6], illetve a Windows Vista-ba tervezett, de végül kimaradt WinFS[9] ötletének népszerűsége már megjelenése előtt.

1. fejezet

Alapfogalmak

Ahhoz, hogy adatbázisalapú fájlrendszerekről beszélhessünk, előbb néhány fogalmat kell ismernünk. Ebben a fejezetben a fájlrendszer jelentéséről, meghatározásáról és típusairól lesz szó, majd a napjainkban megszokottá vált hierarhikus fájlrendszert, ennek előnyeit és hátrányait tárgyalom.

1.1. Fájlrendszerek

A legtöbb számítógép-felhasználó tisztában van azzal, hogy mire is jó, mi is egy fájlrendszer feladata, mi is egy fájl, egy könyvtár. Ez általában a felhasználó személyes számítógépes tapasztalatain alapul, éppen ezért fontosnak tartom egységesíteni ezt a meghatározást.

A számítógépek fő célja adatok kezelése, tárolása, betöltése és létrehozása. A fájlrendszer az, ami biztosítja a számítógépnek ezeket a lehetőségeket, azaz adatok egy perzisztens tárolón való tárolását, éppen ezért minden operációs rendszernek fontos része. Számos különböző megközelítés létezik a permanens tárolók kezelésére. Egyik végletben állnak az egyszerű fájlrendszerek, amelyek elkészítése könnyű, viszont kényelmetlen és nehézkes a használatuk a sok megszorítás miatt.

A másik végletben állnak a bonyolult objektum-alapú tárolású, objektum-orientált adatbázisokra épülő perzisztens tárolási lehetőségek, amelyek annyira felhasználó-közeliek, hogy sem a felhasználóknak, még a programozóknak sem szükséges tudniuk, hogyan is történik valójában a fizikai tárolástól való elvonatkoztatás. Ez a két véglet nagyon távol áll egymástól, és éppen ezért nagyon sok lehetséges megoldás, és nem csak egy jó megoldás létezik egy fájlrendszer megvalósítására.

Fájlok

A sok lehetséges megoldás azonban nem teljesen különböző, mivel vannak olyan tulajdonságok, melyekkel minden fájlrendszernek rendelkeznie kell. Ilyen például az egyik alaptulajdonság: minden fájlrendszernek lehetőséget kell nyújtania egy névvel ellátott adathalmaz tárolására, majd később a név alapján való betöltésére. Ezeket a névvel ellátott adathalmazokat nevezzük *fájloknak*. A fájlrendszer alapjában véve fájlokhoz biztosít műveleteket, és a programok az adatok tárolását ezeken keresztül valósítják meg. általában azonban nem csak néhány fájllal dolgozunk egy számítógépen, tehát egy fájlrendszernek tudnia kell kezelni sok fájlot, ahol a "sok" akár milliós nagyságrendű is lehet.

Könyvtárak

Sok fájlt elég nehéz karbantartani, rendszerezni, bármennyire is beszédes neveket adunk a fájljainknak. Ugyancsak a fájlrendszer feladata lehetőséget nyújtani a fájlok rendszerezésére. A legtermészetesebb rendszerezési módnak a hierarhikus rendszerezést találták a fájlrendszer-tervezők, és ez vált hagyományossá. A hierarhikus fájlrendszerek első verziói még csak egy, majd két mélységű hierarhiát tudtak kezelni, azonban ezt általánosították, így már elméletileg bármilyen mélységű hierarhiát létre lehet hozni, ma már határt a könyvtárstruktúra mélységére csak a fájlrendszer implementációja szabhat.

1.2. A hierarhikus fájlrendszer

A fájlok rendszerezését megkönnyíti a hierarhikus fájlrendszer, de napjainkban ez már nem elég. A merevlemezek mérete ma már a több ezer megabájtot is eléri. Ekkora tárhelyen akár több millió fájl is elfér, amelyeket még ha csoportosítunk is valamilyen szempont szerint könyvtárakba, több ezer könyvtárunk lehet, és még egyszerűbb esetekben is legalább négy könyvtár mélységet kell felhasználnunk. Minden fájlunk helyét megjegyezni lehetetlen. A könyvtárak nevei bizonyos mértékben megkönnyíthetik a keresést, viszont elég gyakran előfordul, hogy egy fájl több kategóriába is beleillik, tehát szívünk szerint több könyvtárba is belemásolnánk, de sajnáljuk a helyet a merevlemezen, és csak egy helyre másoljuk be.

Például: kaptam valakitől egy könyvet, melynek címe *Java játékprogramozás*, amely az OpenGL Javában való programozását tárgyalja játékfejlesztésben alkalmazva, és nekem be kell másolnom valahová: mondjuk van egy *Könyvek* könyvtáram, de ezen belül hova is tegyem? A *Java* könyvek mellé, a *Játékprogramozás*-ba, vagy talán az *OpenGL* könyvtárba? És mi történik, ha amikor megkaptam, épp siettem órára, és mivel az egyetemre

kell egy projektemhez, így az *Egyetem* könyvtárba másoltam, ami még nem is a *Könyvek* mappában van? Hol fogom keresni, mikor szükségem lesz rá?

1.3. Metaadat-indexelés és a keresőprogramok

Erre a problémára napjainkban a keresőprogramok jelentik a megoldást. Ezek arra specializálódtak, hogy több gigabájtnyi adathalmazban keressenek valamit *elfogadható* időn belül. A felhasznált ötlet egyszerű: mikor a gép nincs leterhelve, indexelik a merevlemezen található fájlokat, eltárolnak bizonyos információkat ezekről egy index-adatbázisban:

- a fájl nevét és elérési útvonalát
- fájlformátum-specifikus információkat, mint például MP3 esetén az ID tag, vagy JPEG esetén a méret
- a támogatott formátumok tartalmát, például Office dokumentumok esetén a cím, alcímek

Ezeket az információkat röviden csak metaadatoknak nevezik, aminek jelentése *adat az adatról*[13]. Mivel ezek az adatok eltárolódnak egy index-adatbázisban, a keresés már nem sok fájl közötti keresés, hanem egy adatbázisban való keresés lesz, amelynek sebessége nem a lemezmeghajtó sebességétől függ elsősorban, hanem az index-adatbázis felépítésétől, a felhasznált adatstruktúráktól, és a keresési algoritmustól. Ennek nagy előnye, hogy mindezek cserélhetőek, ha nem vagyunk megelégedve a sebességgel, és implementáltunk egy gyorsabb algoritmust, vagy kitaláltunk egy jobb adatstruktúrat az indexelésre.

Napjainkban a keresőprogramok között erős harcok folynak, és hogy ki nyer, függ a szempontoktól. Léteznek ingyenes programok Windows-ra, Linux-ra, MacOS-ra, léteznek több platformon használhatóak is. Például a Google Desktop Windows, Linux és MacOS operációs rendszer alatt is működik, viszont amint tudjuk, az általánosság a teljesítmény rovására megy, ezért teljesítményben lemarad egyes termékektől[5], amelyek csak egy platformon futnak, mint a Copernic Desktop Search Windows alatt, vagy a Beagle Linux alatt.

2. fejezet

Adatbázisalapú fájlrendszerek

Amint láthattuk, az indexelő programok az operációs rendszerek közti radikális különbségek miatt inkább egy-egy operációs rendszerre specializálódnak, és ott kiemelkedő teljesítményt nyújtanak. Ez az oka, hogy az elmúlt három évben többen is gondolkodtak már azon, hogy mi történne, ha beépítenék a keresést és indexelést az operációs rendszerbe? Az ötlet mára már megvalósult, a főbb operációs rendszerek legújabb verzióiban már léteznek beépített idexelési illetve kereső mehanizmusok. A MacOS X-ben (2007) beépítve megtalálható a Spotlight[6] tehnológia, a Windows Vista (2007) a Windows Search[12] nevű keresőmotorral rukkolt elő, a Linux disztribúciók pedig még csak KDE és Gnome ablakkezelőkkel használhatnak ilyen megoldásokat, például KDE alá egy államvizsga dolgozat eredményeképpen megszületett, kevésbé ismert DBFS[4], Gnome alatt pedig a Tracker[3], valamint a Beagle[1], amely mindkét ablakkezelővel működik.

2.1. Eddigi próbálkozások

Észrevehető azonban, hogy ezek a megoldások nem "épülnek" bele az operációs rendszerbe, általában csak annyiban, hogy az operációs rendszer indítja az indexelő folyamatot, és egy felületet biztosít a kereséshez. Pontos magyarázatot nem lehet találni. Lehetséges, hogy valamilyen tervezési okokból nem akarták ennél mélyebbre beépíteni az operációs rendszerekbe, mint például egyfajta szétválaszthatóság, pedig a fájlok indexelése inkább a fájlrendszer feladata lenne szerintem, mint egy teljesen különálló folyamaté. Talán az is egy elfogadható magyarázat a fájlrendszerbe beépítés elkerülésére, hogy a legtöbben meg szeretnék őrizni a kompatibilitást az előző verziókkal, de szerepelhet az is a listán, hogy egy teljes fájlrendszer megírása igencsak nagy feladat még tapasztalt programozók és rendszertervezők számára is. Ehhez még társulhat az is, hogy a háttérben levő adatbáziskezelő rendszert is meg kellene valósítani, ha a cég nem hajlandó open-source vagy más

felhasználható projektet beépíteni, ami szintén sok idő és erőforrást igényel.

Azonban mindezek a tények nem mindenkit rettentettek vissza, ugyanis már elég régen megvalósították az első igazán adatbázisalapú fájlrendszert, azonban ez nem vált ismertté. Ez a fájlrendszer a BeFS[2] volt, amely 1996-ban jelent meg, mint a BeOS operációs rendszer fájlrendszere, azonban mivel ez az operációs rendszer csak speciális hardveren, BeBox gépeken működött, nem futott be, mint vele egyszintű társai, a Windows vagy a Linux. A BeFS eddig talán az egyetlen fájlrendszer, amelybe beépítettek metaadat támogatást és ezek indexelését is, így elég gyors keresést biztosítva a fájlok között a metaadatok alapján.

Az ötlet már-már feledésbe merült, míg a Microsoft cégtől ki nem szivárogtak bizonyos információk a WinFS (Windows Future Storage) nevű forradalmian új "fájlrendszerről", amely a Windows Vista részeként fog megjelenni. Ez végül a Windows Vistából kimaradt, de közelsem mondtak le róla, továbbra is fejlesztik. A Microsoft dokumentációi szerint a WinFS nem csak fájlrendszer, hanem a fájlok rendszerezését forradalmasítja a háttérben levő adatbázis segítségével, de ugyanakkor kompatibilis lesz az NTFS fájlrendszerrel is[9].

2.2. Előnyök és hátrányok

Összehasonlítva a hierarhikus fájlrendszerrel, számos előnye lenne egy működő adatbázisalapú fájlrendszernek, így érthetővé válik a szoftveróriás törekvése a megvalósításra. Azonban ennek is léteznének hátrányai is.

A hierarhikus fájlrendszerek esetében a fájl nevéből és elérési útjából információkat szerezhetünk a fájl tartalmáról, azonban a fájl és könyvtárak neveire elég sok megszorítás létezik. Például a fájlnevek nem tartalmazhatnak akármilyen karaktereket, hosszuk is lehet korlátozott, de korlátozás nélkül belátható, hogy kényelmetlenné válhat használatuk. Erre megoldásként elhelyezhetjük fájljaikat beszédes nevű könyvtárakban. Azonban ha CD-re, vagy DVD-re akarjuk írni őket, ezeknek a standard fájlrendszere csak korlátozott mélységű könyvtárstruktúrát és viszonylag rövid fájlneveket enged meg. Ezzel szemben egy adatbázisalapú fájlrendszer használatával a fájlról és tartalmáról nem csak a fájl nevéből és eléresi útvonalából tudhatunk meg dolgokat, hanem a hozzá kapcsolódó metaadatokból, attribútumokból is információkat kapunk. Ehhez viszont nekünk is tennünk kell valamit, azaz a fájljainkat megfelelő attribútumokkal kell ellátnunk. Például ha a barátnőmről karácsonykor készült képeket szeretném megnézni, akkor hierarhikus fájlrendszer esetén az összes karácsonyi fotót át kellene nézzem, viszont ha felmásoláskor a megfelelő metaadatokkal elláttam a képeket, azaz a képekhez társítottam a "karácsony" kulcsszavat, illetve mindegyik képhez még attribútumként odaírtam a szereplőket, akkor csak a "karácsony" és egy név beírásával láthatom az összes képet, amire szükségem van.

Az adatbázisalapú fájlrendszer ellen szól viszont az a tény, hogy a felhasználónak újabb felületet kellene megszoknia a metaadatok hozzáadásához, szintén egy újat a kereséshez, nem is beszélve az áttérésről, amely az összes fájl "felcímkézését" jelenthetné. Ezen a téren azonban a hierarhikus fájlrendszerek sem állnak jobban, mivel a kereséshez ott is egy keresőprogramot kell használni, mivel a standard általában a sok fájlművelet miatt lassú, de legalább nem kell megadni metaadatokat, ami amúgy időbe telne, és lehet, hogy soha nem válna hasznunkra.

Egy másik közös hátrány a két megközelítésben, hogy ha CD-re vagy DVD-re írjuk fájljainkat, majd a merevlemezről töröljük őket, akkor már nem kereshetünk rá metaadatokra ezek között, mert az optikai lemezre való átíráskor "elvesznek" a metaadatok.

2.3. Mire van szükség?

E tények ismeretében elgondolkodtató, hogy érdemes-e elkezdeni egy adatbázisalapú fájlrendszer megvalósítását. Talán ezt eldönteni segíthet, ha felvázoljuk, hogy mire is lenne szükség az implementációhoz.

- Mivel adatbázisalapú fájlrendszerről beszélünk, elsősorban szükségünk lenne egy fájlrendszer megtervezésére és megírására annak teljes bonyolultságában, hardverközeli szinten.

- Szükséges lenne ugyanakkor a fájlokhoz kapcsolódó metaadatok tárolása is. Ez lehetséges lenne akár a fájlrendszerben is, de elképzelhető akár az is, hogy a fájlokhoz kapcsolódó információkat csak egy metaadat-adatbázisban tároljuk el.

- Amennyiben a metaadatok a fájlrendszerben találhatóak, kereséskor szintén nagy mennyiségű fájlműveletre lenne szükség, akárcsak a hierarhikus fájlrendszer esetén, tehát egy adatbázisra is szükség van, amelyben eltároljuk a metaadatokat. Ennek velejárója egy adatbáziskezelő-rendszer, amely az adatbázishoz való hozzáférést és a vele való műveleteket biztosítja, mint beszúrás, törlés, keresés.

- A keresés adatbázisokban általában indexek alapján történik a sebesség növelése érdekében, tehát egy indexelési adatszerkezetet is szükséges választanunk és megvalósítanunk a vele kapcsolatos műveleteket, vagyis magát az indexelést.

- A fájlok metaadatait az adatbázisban frissíteni kell, legrosszabb esetben minden fájlművelet után, tehát szükséges valamiféle kapcsolatot kiépítenünk a fájlrendszer és az adatbázis között, ezzel helyettesítve az indexelő-programok folyamatos háttérben futó indexelését.

- Biztosítani kell a felhasználónak egy felületet, amellyel módosíthatja a fájlokhoz kapcsolódó metainformációkat. Erre a legkézenfekvőbb talán az lenne, ha a fájlok mentésekor lehetősége lenne megadni ezeket.
- A kereséshez szintén szükséges lenne egy felület, amellyel az adatbázisban levő metaadatok alapján megtalálhatnak bizonyos fájlokat.
- Opcionálisan az adatbázis karbantartásához is hasznos lehet egy felület, azonban ez nem létfontosságú.

Amint látható, nagyon sok feladat vár ránk egy teljes adatbázisalapú fájlrendszer megvalósításához, ha úgy döntünk, hogy belevágunk, azonban a feladatok száma csökkenthető már meglevő programok felhasználásával.

3. fejezet

A megvalósításról

3.1. A cél-operációs rendszer kiválasztása

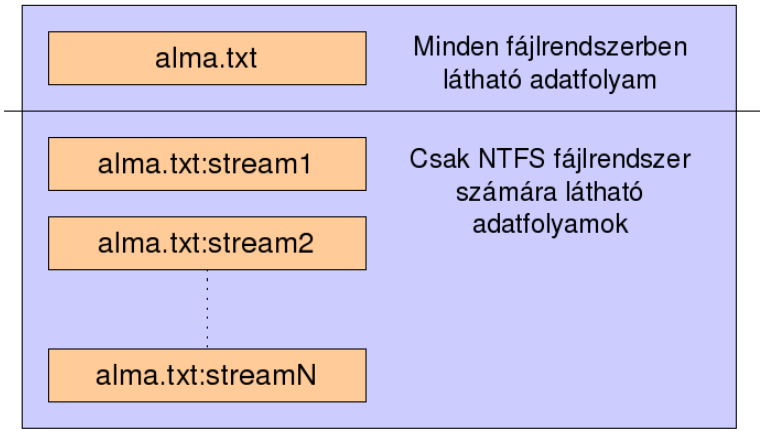
Már láthattuk, hogy az operációs rendszerek közti különbségek miatt nehéz egy fájlrendszert általánosan megírni, és ha sikerül is, a sebesség rovására megy, tehát úgy döntöttem, hogy egy operációs rendszerre próbálom megvalósítani az alkalmazásom, azaz egy "virtuális" adatbázis-alapú fájlrendszert, mivel nem egy valódi fájlrendszerről van szó. Az operációs rendszer megválasztása az első feladat.

A MacOS-t könnyű volt kizárni, mivel nincsen megfelelő hardverem a futtatásához, ismereteim is korlátozottak róla, és már létezik beépítve a Spotlight hasonló célokkal. Tehát maradt a számos Linux disztribúció és a közkedvelt Windows. A Linux vonzónak tűnhet, mivel nyílt forráskódú és bármire lenne szükségem, megkereshetném, de ezt már mások megtették, amint már említettem, hasonló projektek egyre nagyobb számban léteznek. Mivel azonban az adatbázisalapú fájlrendszer célja az átlagfelhasználó életének könnyítése, aki általában Windows-t használ, ez elég nagyot billent a mércén a Windows irányába.

3.2. Metaadat-támogatású fájlrendszer

A Windows választásának több előnye is van, mivel már számos dolog készen rendelkezésemre áll, csak fel kell használnom ezeket. Mivel a fájlrendszer jelentheti az egyik legnagyobb nehézséget, ezért hasznos volna egy kész fájlrendszert felhasználni. Választási lehetőségeim szűkösek, FAT32 illetve NTFS közül kell választanom, azonban az NTFS fájlrendszer tűnik a legjobb választásnak, mivel az újabb Windows verziók (Windows 2000 óta) ezt használják, ugyanis a Windows 2000-be beépített NTFS fájlrendszer számos újítást hozott. Ezek közül legfontosabbnak az alternatív adatfolyamokat[11] tartom

a feladatom szempontjából, mivel ezek lehetőséget nyújtanak egy fájlhoz több adatfolyam társítására. Ez azt jelenti, hogy a fájl aktuális tartalmán kívül, ami a fájl fő adatfolyama, társíthatunk más adatfolyamokat, amelyekre a fájlnévvel és az adatfolyam nevével hivatkozunk (ld. 3.1 ábra).



3.1. ábra. Alternatív adatfolyamok

Például egy *alma.txt* fájlhoz társíthatunk egy *szoveg* nevű adatfolyamot , amelyre *alma.txt:szoveg* névvel hivatkozhatunk, és módosíthatjuk a tartalmát. Ezeket az alternatív adatfolyamokat használhatjuk a Windows 2000-től kezdődően a metaadatok tárolására. Ennek érdekében létezik egy speciális adatfolyam minden fájlban, a *SummaryInformation* adatfolyam, amelyben Szerző, Cím, és hasonló mezőket tárolhatunk, amelyeket a fájl tulajdonságai között, a *Summary* fülön módosíthatunk.

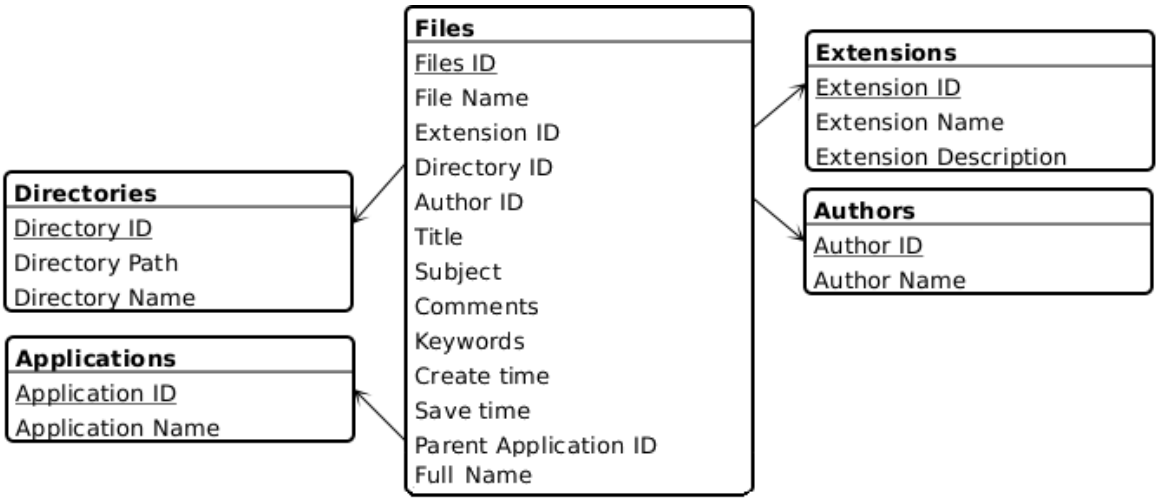
A *SummaryInformation* írására és olvasására használt függvényekről nem találtam információkat, azonban a tárolt adatstruktúrának egy leírását megtaláltam[16], ezért megvalósítottam saját függvényeimet ezek módosítására. Ennek segítségével bizonyos mennyiségű plusz információt társíthatok minden fájlhoz. Így az NTFS fájlrendszer felhasználásával nem szükséges fájlrendszert implementálnom, csak majd a fájlműveletekkel egyidőben a metaadat-adatbázist kell frissítenem.

3.3. A metaadat-adatbázis

Szerkezet

Az adatbázis tartalma nagyrészt attól függ, hogy milyen meta-adatokat akarunk tárolni a fájlokról. Az eltárolandó metaadatok esetemben nagyrészt megegyeznek az NTFS által támogatott metaadatokkal. Az NTFS a következő metaadatokat támogatja: Cím,

Téma, Szerző, Kulcsszavak, Megjegyzések, Sablon, Revízió szám, Szülőalkalmazás, Létrehozás ideje, Utolsó mentés ideje, Kategória[11]. A fent említettek közül úgy döntöttem, nem fogom eltárolni a Sablon, a Revíziószám és a Kategória adatokat, mivel az első kettőt szerintem átlagfelhasználók nem használnák, míg a Kategória tárolásával az a gond, hogy ezt az NTFS egy újabb speciális adatfolyamban tárolja, külön a többitől, és ezen adatfolyam struktúrájanak kitalálása dokumentáció hiányában időigényes és szinte lehetetlen. Azonban már enélkül is a felhasználó rendelkezésére áll öt darab szöveges mező adatok tárolására, ami szerintem elégséges, vagyis mindenképpen több, mintha csak a fájl nevét tárolhatnák el.



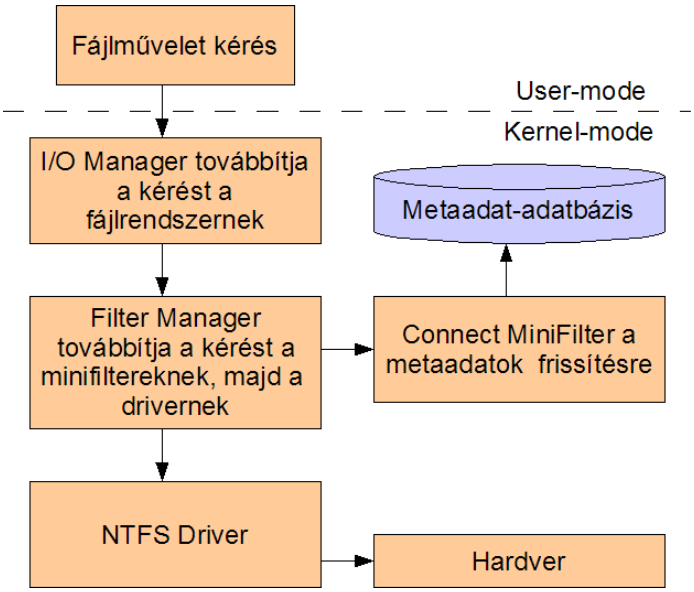
3.2. ábra. Az adatbázis szerkezete

Az adatbázis szerkezetének egyszerűnek kell lennie (ld. 3.2 ábra), hogy a keresések megfelelően gyorsak legyenek. Az adatbázisban fájlokról tárolok információkat, ezért a *Files* tábla a legfontosabb tábla, minden más tábla ehhez kapcsolódik. A redundancia elkerülése érdekében a könyvtárneveket, szerzőket, kiterjesztéseket és alkalmazásneveket külön táblákban tároltam, és a fájlok táblájában csak a megfelelő ID-ket tárolom.

Frissítés

Elsőre nehéznek tűnhet a fájlműveletek figyelése, azonban a Microsoft a programozók kezére bocsájtja a minifilter tehnológiát[8], amely ezt lehetővé teszi. A minifilterek lehetőséget nyújtanak bármilyen fájlművelet előtt vagy után a programozó által megírt műveletek végrehajtására. Ezt a tehnológiát számos területen alkalmazzák már, kezdve vírusellenőrzéstől, fájlok titkosításán át egészen a sűrítésig.

Esetünkben minden lemezre írás után ellenőrizhetjük, majd ha szükséges, frissíthetjük a módosított fájlhoz tartozó metaadatokat az adatbázisban (ld. 3.3 ábra).



3.3. ábra. Fájlműveletek végrehajtása[8]

Adatbáziskezelő rendszer

A metaadatok tárolására szükségem van egy adatbázisra, valamint egy adatbáziskezelő rendszerre. Mivel a programot a lehető legkevesebb szoftver-követelménnyel szeretném megvalósítani, ezért nem jöhetnek szóba különálló adatbázis-kezelő rendszerek, mint a MySQL vagy a Microsoft SQL Server. Ezeken kívül létezik néhány, amelyek programokba beépíthetőek, viszont a legtöbb Java Virtual Machine alatt fut, tehát ezeket is kizárhatom. Standard C-ben megírt folyamaton belüli adatbázis-kezelő rendszert csak kettőt találtam, és mivel a Minifilter Windows Driverként fut kernel-módban, ezért C-ben kell megírni, így csak ezek maradtak.

A Novell cég *FLAIM*[14] nevű adatbáziskezelő rendszere szóba jöhetett volna, azonban ennek a fejlesztését már abbahagyták, és a dokumentációja hiányos, dokumentáció nélkül pedig nagyon nehéz kezelni. Ezzel ellentétben egyre gyakrabban használják az *SQLite*[15] nevű nyílt forráskódú adatbáziskezelő rendszert, amelyhez már több programozási nyelvre is írtak interfészeket, de eredetileg C-ben íródott, ezért ez tűnt a legmegfelelőbbnek. Persze ez sem tökéletes, mert magasszintű konkurrenciát nem tud még kezelni, azonban mivel a felhasználó egyszerre csak egy fájl metaadatait tudja módosítani, ezért konkurrencia-kezelésre nincs még nagy szükség, tehát ez egy elviselhető hiányosság.

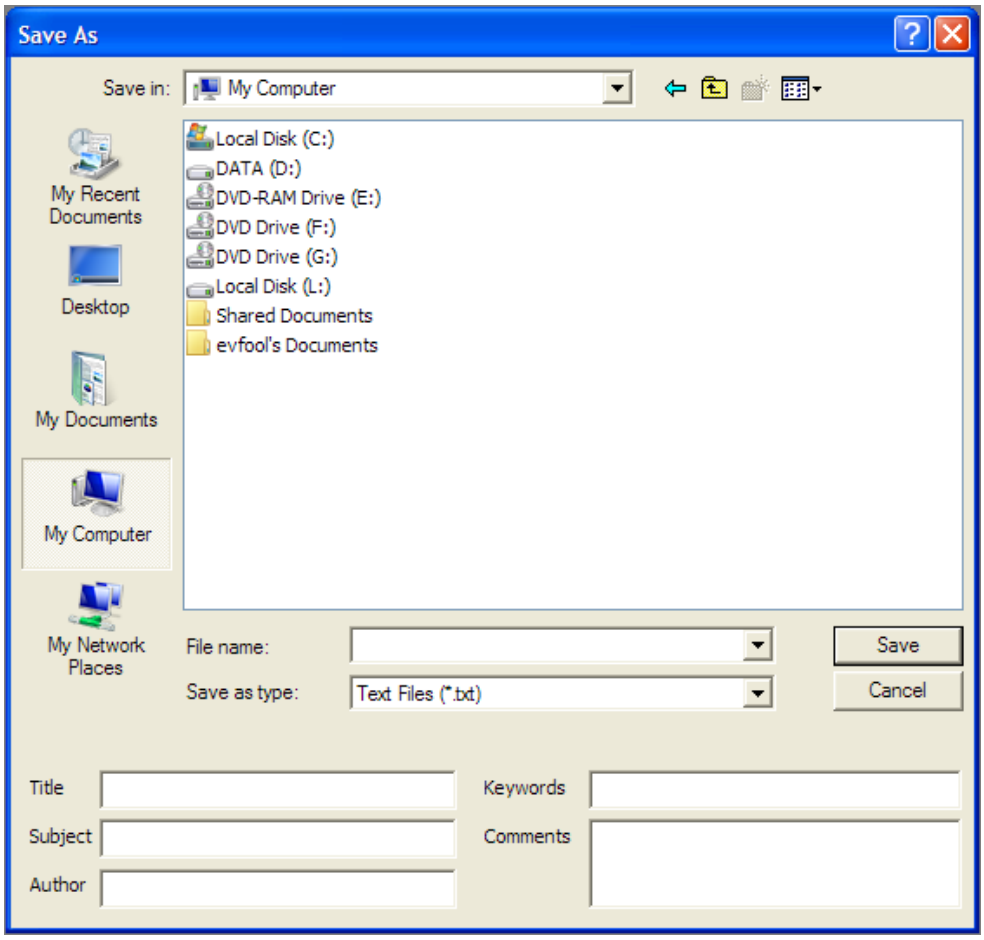
Az SQLite könnyen integrálható, akár forráskód szintjén is a minifilterbe, megoldja az indexelést, beszúrásokat, törléseket, tehát nem marad más hátra, mint az adatbázis megtervezése.

3.4. A felhasználói felület

Mindeddig csak a háttérben zajló dolgokról beszéltünk, azonban ez magában semmit sem ér, valamilyen felületet kell biztosítani a felhasználó számára is. Három felülettel kiegészítve a fájl tulajdonságainál található Summary fület, minden metaadat-műveletet el tud végezni a felhasználó, amire szüksége lehet.

Metaadatok társítása a fájlokhoz

A fájlokhoz metaadatok társítása történhetne egyszerűen minden esetben a Windows által biztosított Summary fülön, azonban ez a felhasználók munkáját szaporítaná: miután elmentette a fájlját, meg külön meg kellene nyissa a fájl tulajdonságait, és kitöltse a mezőket. Ezt a két lépést össze lehetne sűríteni egybe, ha a standard mentés párbeszédablak tartalmazna mezőket a metaadatok számára. Ezzel egy kézenfekvő megoldást biztosítanék a metaadatok megadására már a fájl mentésekor (ld. 3.4 ábra).



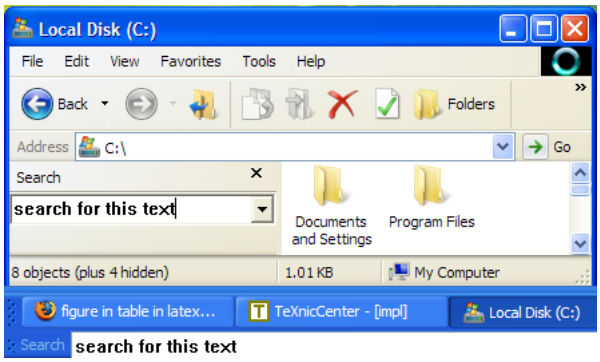
3.4. ábra. Az új mentés párbeszédablak

Hasonló párbeszédablak-kiterjesztésekre ad lehetőséget a Microsoft a párbeszédablak-sablonok definiálásával[10], vagyis a megfelelő sablon elkészítésével és a standard Windows mentés párbeszédablak kicserélésével megoldható az igények szerint beállított párbeszéd-ablak beépítése a Windowsba. A standard párbeszédablak kicserélése elsőre kihívásnak tűnt, azonban ez is egyszerűen megoldható, amely segítségével minden alkalmazásba futás közben saját DLL könyvtárainkat betölthetjük egy egyszerű érték megadásával a Registryben[7]. Ezt a trükköt a Microsoft párbeszédablak-sablonjaival összekombinálva minden alkalmazás a módosított mentés párbeszéd-ablakot fogja használni.

Keresés

Mivel egy kereséshez szeretnék egy felületet, amely csak néhány szó beírásához kell lehetőséget nyújtson, így elég egy egyszerű szövegmező. Ha csak ennyi a felület, előnyös lehet az operációs rendszerbe való integrálás terén is, mivel kis helyet foglal és beépíthető a Windows taskbar-ba, így mindig kézügyben lesz.

Ilyen integrálható komponensek készítésére is létezik megoldás, a Microsoft platform fejlesztői készletében leírt Explorer Band (ld. 3.5 ábra fent) illetve DeskBand (ld. 3.5 ábra lent) programozható komponensek[10], amelyek beépülnek a Windows Explorerbe illetve a Windows Taskbarba. Ezek megvalósítása egyszerű COM komponensek programozásával történik, amelyről már nagyon sok könyv létezik napjainkban.



3.5. ábra. Explorer Band és DeskBand a keresésre

Találatok megjelenítése

Egy keresés találatainak megjelenítésére számos lehetőség van. Talán a legkézenfekvőbb egy egyszerű kis program lenne, amely egy listánézetben megjeleníti a keresési kritériumoknak megfelelő fájlokat. Egy másik megoldás lenne a listanézet integrálása a keresés Explorer Band-be, aminek előnye volna, hogy szintén a standard Windows felületen jelenne meg, nem egy külön alkalmazásban.

Következtetés

A tárolandó adatmennyiség napjainkban egyre nő, adataink hatékony rendszerezésének illetve az ezek közötti hatékony keresés víziója számos programozót megmozgat napjainkban. Dolgozatom célja is hasonló: szemléltetni a lehetőséget adataink rendszerezésére a hierarhikus rendszerezéstől való elszakadás által, mégis ezzel a kompatibilitást megőrizve.

A dolgozat egyúttal szemlélteti a Windows operációs rendszer kiterjeszthetőségét, erre jó példaként szolgálva a részben implementált fájlrendszer minifilter, a Windows felületébe, az Explorerbe illetve a Taskbar-ba beépíthető keresőfelület, valamint a kiterjesztett mentés párbeszédablak, amelyek néhány egyszerű kiegészítéssel együtt egy felhasználóbarát virtuális adatbázisalapú fájlrendszert implementálhatnak.

Ábrák jegyzéke

Irodalomjegyzék

[1] Beagle Project, http://www.beagle-project.org. *Beagle.*

[2] Dominic Giampaolo. *Practical File System Design with the Be File System.* Morgan Kaufmann Publishers, Inc, 1999.

[3] Gnome, http://www.gnome.org/projects/tracker/. *Gnome Tracker.*

[4] O. Gorter. Database file system. Technical report, University of Twente, August 2004.

[5] Tom Noda. Shawn Helwig. Benchmark study of desktop search tools. Technical report, University of Wisconsin-Madison, April 2005.

[6] Apple Inc. Macos x leopard new features.

[7] Ivo Ivanov. Api hooking revealed. CodeProject Article.

[8] Microsoft. *File System FIlter Driver.* http://www.microsoft.com/whdc/driver/filterdrv.

[9] Microsoft, http://msdn.microsoft.com/en-us/library/aa186116.aspx. *Introducing "Longhorn" for Developers.* Chapter 4:Storage.

[10] Microsoft. *Microsoft Platform SDK 2003.*

[11] Microsoft. Msdn : File streams(windows). http://msdn.microsoft.com/.

[12] Microsoft. Windows search.

[13] Allan Nielsen. Metadata file systems. Technical report, National University of Singapore, November 2007.

[14] Novell. *FLAIM.* http://developer.novell.com/wiki/index.php/FLAIM.

[15] SQLite. *SQLite Documentation.*

[16] Anja Schaffhirt. Andre Wünsche. Notes on the summaryinformation stream. http://sedna-soft.de/summary-information-stream/.