



Budapesti Műszaki és Gazdaságtudományi Egyetem
Közlekedésmérnöki és Járműmérnöki Kar
Közlekedésautomatikai Tanszék

GTFS alapú tömegközlekedési adatok előállítása

TDK dolgozat

Készítette: Cseh Attila, közlekedésmérnöki (MSc) szak
Konzulens: Bécsi Tamás adjunktus

2011. november

Tartalomjegyzék

1	Bevezetés	3
2	Mi is az a GTFS?.....	5
2.1	Az ötlet	5
2.2	A specifikáció formai követelményei.....	7
2.3	A specifikáció tartalmi értelmezése [4]	8
3	A keretprogram rendszere.....	13
3.1	A Google Earth szerepe	14
3.2	A MS Access szerepe	15
3.2.1	Az adatbázis felépítése.....	16
3.2.2	Az adatbázis használata	18
3.3	A keretprogram feladatai.....	21
3.3.1	Importálás a Google Earthből	21
3.3.2	Az indulási időpontok szerkesztése	23
3.3.3	A GTFS alapú adatok generálása	25
3.4	A publikálás.....	28
4	Tapasztalatok a szegedi adatok feldolgozása közben	31
5	Lehetséges alkalmazások, fejlesztési lehetőségek.....	33
6	Összefoglalás	36
	Források	37
	Melléklet	38

1 Bevezetés

Idén nyáron érett be a Budapesti Közlekedési Központ (BKK) fejlesztésének gyümölcse, amikor a BKV Zrt. útvonaltervezője „átköltözött” a Google Térkép felhasználói alkalmazásra. A cél természetesen a szolgáltatás színvonalának növelése, illetve új utasok megnyerése a minél szélesebb körű információnyújtás segítségével. Ez inspirált arra, hogy a témában elmélyülve ennek a fejlesztésnek a technikai részleteivel foglalkozzak.

A kulcs a General Transit Feed Specification (GTFS) alapú adatok megismerése volt. Ez eredetileg a Google által kifejlesztett olyan egységes adatformátum, amely összerendeli a közösségi közlekedés menetrendjét a GPS-ből származó földrajzi pozíciókkal. Ezáltal lehetővé válik a viszonylatok, megállóhelyek térképes megjelenítése, és a menetrend alapú útvonaltervezés is. A GTFS állományt nyilvánosságra hozva bárki a saját maga által készített alkalmazásához is felhasználhatja az adatokat, ezáltal tovább bővítve a tömegközlekedést igénybe vevők körét. A GTFS tulajdonképpen egy szabvány, így a különböző szolgáltatók által rendelkezésre bocsátott fájlok egységesek, a felhasználói programok által mindegyik egyformán feldolgozható, így az útvonaltervezés hatóköre kibővíthető országos, mi több, globális szintig is.

Dolgozatom egy saját keretprogram fejlesztéséről szól, amely képes a GTFS szerinti adatok előállítására. Az alkalmazás három részből áll. A Google Earth programban lehet a megállóhelyek elhelyezkedését és a járatok útvonalait bejelölni, innen származnak a földrajzi pozíciók. Egy Microsoft Access adatbázisban szerkeszthetők a megállóhelyek, viszonylatok, járatok tulajdonságai, kapcsolatai, és ez tartalmazza a Google Earthből kinyert koordinátákat – hozzárendelve az előzőekhez. A harmadik rész az maga a keretprogram, amely Delphi fejlesztői környezetben készült. Feladata a járatok indulási időpontjainak felhasználó általi bevitele és tárolása, a Google Earth és a Microsoft Access közötti adatátvitel biztosítása, illetve magának a GTFS formátumnak az előállítása.

A dolgozat részletezi az egyes programok funkcionalitását, illetve ezek kialakulásának előzményeit a GTFS specifikációk tekintetében. A fejlesztés végén elkészült Szeged város helyi közösségi közlekedésének „GTFS alapú menetrendje”, amely innentől

kezdvé már nemcsak az indulási időpontokat és megállóneveket tartalmazza, hanem a járatok útvonalának és a megállóhelyeknek a koordinátáit is. Ismertetem az adatbevitel során fölmerült anomáliákat, az ezekre kitalált megoldásokat. Az elkészült fájlt igyekeztem a megfelelő webes felületeken hozzáférhetővé tenni, és olyan kiszolgálást nyújtani mellé, hogy a felhasználók mindig naprakész információkkal tölthessék föl az adatbázisukat.

Az érkezett visszajelzések, és internetes kutatások alapján bemutatom a GTFS adatok lehetséges alkalmazási területeit, amelyek részben már megvalósultak, és nagy népszerűségnek örvendenek. A GTFS adatok napjainkban kezdenek igazán elterjedni, és az adatkészítők a Google csapatával együtt igyekeznek a specifikációt is a felmerülő egyre nagyobb igényekhez igazítani. A keretprogram továbbfejlesztési lehetőségeit ennek, és a GTFS adatok újabb forrásainak tükrében vizsgálom.

2 Mi is az a GTFS?

2.1 Az ötlet

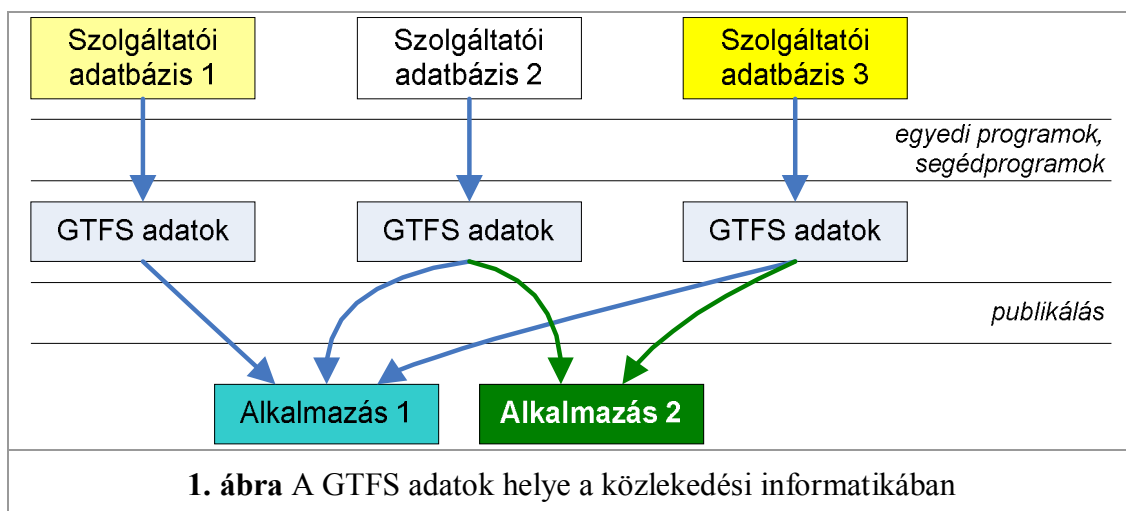
A General Transit Feed Specification, vagyis a „Tömegközlekedési Adatok Általános Leírásának” története 2006. szeptember 25-én kezdődött, amikor a specifikáció első változatát nyilvánosságra hozták. [1] A Google ekkora már kifejlesztette a Transit [2] nevű programot, amely a Google Térkép [3] alkalmazásba integrálva kibővítette az addig csak gyalogos és autós útvonalajánlatokat a tömegközlekedési szolgáltatókra is. Az útvonaltervezés továbbra is a megszokott módon történt a felhasználók szempontjából, a képernyőkön viszont az addigi két közlekedési módválasztó fül mellett megjelent a tömegközlekedést jelképező kis ikon is. Ennek kiválasztása után a felajánlott útiterv már az adott környék közlekedési szolgáltató cégének menetrendjét figyelembe véve készült el.

A Google térképek a műholdfelvételek megjelenítése és globális lefedettségük miatt közkedveltnek mondhatók, és ez a Transit bevezetésével csak tovább erősödött. Ahhoz, hogy egy ilyen világméretű dolog működni tudjon, rengeteg adatra van szükség. A klasszikusnak mondható útvonaltervezők általában vagy egy adott vállalat menetrendjét, vagy az adott lefedettségi területen szolgáltató cégek eltérő adatbázisait használják fel. Természetesen a Google ezt az utat nem vállalhatta fel globális szinten, inkább megfordította a rendszert: nem a Transit egységesítette a szolgáltatóktól kapott információkat, hanem a Google kért minden érintettől egységes tömegközlekedési információkat. Nyilván az erőviszonyok is hozzájárultak ahhoz, hogy a Google mindezt megtehesse, hiszen az adatok feldolgozását ingyen vállalja, a vállalatoknak pedig érdekük a minél hatékonyabb, széleskörű, gyors és egyszerű utastájékoztatás. Persze a Google a cégeket nem hagyta magukra a feladattal: közzétett egy jól felépített, áttekinthető leírást [4] a várt tömegközlekedési adatokról. Ez a GTFS, amely mellé több kisegítő programot is fejlesztettek annak érdekében, hogy a szolgáltatók maguk is megjeleníthessék, ellenőrizhessék, vagy frissíthessék az előállított adatokat. Több webes oldalon keresztül írnak a GTFS adatok internetes megosztásának mikéntjéről, illetve az előállításukhoz felhasználható térképes vagy más segédprogramokról. Természetesen önálló „GTFS-készítő programot” a Google nem tud mellékelni, hiszen az egyes vállalatok teljesen eltérő adatbázisokban tartják nyilván a menetrendeket. Egyedül a

TransXChange nevű, a brit közlekedési minisztérium által támogatott, az Egyesült Királyságban egységesen használt programban nyilvántartott XML formátumú adatok konvertálására létezik segédprogram. [5]

Szerencsére azonban a GTFS alapú adatok előállítása nem igényel túlságosan mély informatikai ismereteket (viszont a közlekedési folyamat átlátása elengedhetetlen). A tömegközlekedési vállalatok egy saját útvonaltervező program kifejlesztéséhez képest jelentősen kisebb erőfeszítéssel hatalmas előnyre tesznek szert, így nem csoda, hogy mára a világon körülbelül 500 város tömegközlekedése „hirdeti magát” a Google Térképen. 2011 júniusa óta a BKV Zrt. is ennek a körnek a tagja, az ő megjelenésük nagyobb sajtóvisszhangot is keltett. Viszont nem ők az elsők hazánkban – bár hivatalosan mégis. Mivel a GTFS egy teljesen publikus specifikáció, így az ennek megfelelő adatokat tulajdonképpen bárki elkészítheti: léteznek is nem hivatalos (tehát nem a szolgáltatók által közzétett) GTFS adatok például a MÁV-ról, GySEV-ről, Volánbuszról és a TDK munkám eredményeként immáron Szegedről is.

Az rendelkezésre álló egyre nagyobb számú egységes információ hullámaint meglovagolva a Transit mellett a Google-tól független, új alkalmazások is születtek, amelyek ugyancsak képesek a GTFS formátum feldolgozására, és a tömegközlekedést használók legnagyobb örömeire egyre szélesebb skálán kínálnak különféle hasznos megoldásokat az utastájékoztatás területén (lásd 5. fejezet).



A GTFS jelenleg kialakult vázlatos felhasználását az 1. ábra mutatja. Természetesen a felhasználói alkalmazások nem föltétlenül használják fel minden vállalat GTFS adatait, például egy adott városi menetrendet megjelenítő programnak elég csak az adott cégét

letöltenie. Az ábrán jól látszik, hogy egy intelligens közlekedési rendszer megvalósítását alapvetően két feladatkörre lehet bontani. Az egyik a megfelelő minőségű adatok előállítása és publikálása, ami tulajdonképpen a szolgáltató vállalatok feladata, és természetesen GTFS formátumban történik. A másik feladat pedig magának az intelligens alkalmazásnak az elkészítése, amelynek inputját az elkészült GTFS adatok szolgáltatják. Ezáltal több idő és figyelem jut a tervező, optimalizáló, megjelenítő algoritmusok kidolgozására, és egy felhasználóbarát felület kialakításra az adatforrásokkal való vesződség helyett. Ilyen módon az alkalmazásfejlesztők válnak felhasználókká a GTFS alapú információkat szolgáltatók szemszögéből.

Egy ilyen feladatmegosztás mindkét fél munkáját egyszerűbbé teszi. A szolgáltatók szemszögéből egy informatikai fejlesztés nagy munkát jelenthet, mert ilyenkor a vállalati adatbázishoz illeszteni kell az új rendszert, az interfészeket definiálni, az adatok konzisztensségét pedig biztosítani kell. Egy későbbi fejlesztés, adatbázis módosulás hasonló bonyodalmakkal járhat, amibe ráadásul újra be kell vonni a fejlesztő céget is. A GTFS előállításakor egyrészt mindez egyszer történik csak meg, másrészt a specifikáció egyszerűségéből adódóan vállalon belül is megoldható ez a feladat. Így a vállalat igényei egyszerűbben érvényesíthetők, könnyebb az utólagos változtatás is. Persze az adatok karbantartását föl kell vállalni ebben az esetben.

Az alkalmazásfejlesztők szemszögéből is előnyösebb egy olyan alkalmazást létrehozni, amely képes a GTFS adatok fogadására. A kulcs szintén a feladatok egyszeri elvégzésében rejlik. A fejlesztéskor éppen elég probléma adódik a különböző platformokra történő adaptációból is, így a különféle szolgáltatók számára készítendő, eltérő adatbázisokhoz kapcsolódó interfészek kialakítása csak fölösleges teher. Egy GTFS szerinti „csatlakozó felület” esetén ugyanaz az alkalmazás bármely szolgáltató (GTFS) adataival kompatibilis lesz.

2.2 A specifikáció formai követelményei

A GTFS formátumú állomány nem más, mint egy egyszerű, tömörített (ZIP kiterjesztésű) fájl. Ez tartalmazza azokat a közönséges TXT kiterjesztésű szöveges dokumentumokat, amelyekben már valóban tömegközlekedési adatokat találunk (2. ábra).

```

route_id,agency_id,route_short_name,route_long_name,route_desc
35,1,78A,egyirányú körjárat,
40,1,84,,
41,1,84A,,
42,1,90H,műszakszállító járat,
43,2,1,, "Anna-kút, Tavasz u. megállókban nem áll meg!"
44,2,5,trolibusz,
46,2,3F,,
47,2,4,,
48,2,7,trolibusz,

```

2. ábra Egy GTFS szövegfájl részlete

A dokumentumok adatbázis táblákhoz hasonlítanak, az első sorokban a mezőnevek, a többi sorban ezeknek megfelelően az egyes rekordok láthatók. A mezőket vesszővel, a rekordokat új sorral választjuk el egymástól. A specifikáció több mezőnevet is definiál, ebből némelyek feltüntetése kötelező (viszont az egyes rekordoknak már nem feltétlenül kell értékeket is tartalmazniuk ezekhez a mezőkhöz), némelyeket pedig csak akkor használják, ha valamilyen többletinformációt szeretnénk közölni. Például a *route_long_name* tulajdonságot kötelező feltüntetni, viszont nem mindenhol értelmezhető, ezért ezekben a sorokban a mezőket elválasztó vesszők közé nem írunk semmit. Ha egy leíró jellegű tulajdonságban, mint amilyen például a *route_desc*, vesszőt szeretnénk használni, akkor ez egész szöveget idézőjelek közé kell tenni, de arra is van megoldás, hogy ilyen esetekben az idézőjelet a mondat közben is lehessen használni. A mezők adattípusai itt nem érdekesek, hiszen mindent nagyon egyszerűen, szöveggént (string) adunk meg – részben ettől lesz egységes a GTFS, részben pedig a használt karakterkódolástól, ami az UTF-8-nak megfelelő szabvány, így a magyar ékezetes betűk is helyesen fognak megjelenni a felhasználás során.

A lehetséges 13 darab txt fájl közül – a mezőkhöz hasonlóan – nem kell mindegyiket használni, csak hatot kötelező, a többiről igény szerint dönthetünk, attól függően, milyen minőségű tömegközlekedési információkat szeretnénk nyújtani. Ennek a rugalmasságnak köszönhetően egyes szolgáltatókról akár a viszonylatok vonalvezetésének koordinátái nélkül is készíthetők GTFS adatok, míg más vállalatokról még a tarifarendszert is lehet publikálni.

2.3 A specifikáció tartalmi értelmezése [4]

Szeged város helyi közösségi közlekedési adatainak GTFS formátumban való elkészítésekor egyrészt a helyi tömegközlekedés, másrészt a szegedi viszonylatstruktúra

sajátosságait figyelembe véve „válogattam ki” a specifikáció azon elemeit, amelyeket a rendelkezésemre álló adatforrások alapján föl tudtam használni a lehető legpontosabb tájékoztatást nyújtva az elkészült GTFS adatsomaggal.

Mindezek fényében nyolc szövegfájl állítottam elő. Az ezekben használt mezőnevek jelentését igyekszem megvilágítani a következőkben, amelyek nagyjából megegyeznek a specifikációban leírt magyarázatokkal, azonban akadnak kisebb eltérések, illetve a néhol túl általános leírást konkretizáltam a „szegedi projektnek” megfelelően.

agency.txt – a szolgáltató cégekről szóló információkat írja le

`agency_id`: a szolgáltató azonosítója, minden rekordban különböző

`agency_name`: a szolgáltató neve

`agency_url`: a szolgáltató honlapjának címe

`agency_timezone`: a szolgáltató területén érvényes időzóna (ennek értéke Magyarországon mindenütt: Europe/Budapest)

`agency_phone`: a szolgáltató telefonszáma

`agency_lang`: a szolgáltató által használt nyelv (ez a mi esetünkben: hu)

stops.txt – a megállóhelyekre vonatkozó információkat írja le

`stop_id`: a megállóhely azonosítója, minden rekordban különböző

`stop_name`: a megállóhely neve, ez lehet azonos például végállomásokon vagy átszállási csomópontokban, de alapjában véve is a megálló általában párban található, a nevük megegyezik, az azonosítójuk viszont eltérő

`stop_lat`: a megállóhely szélességi koordinátája (pl.: 46.2701916)

`stop_lon`: a megállóhely hosszúsági koordinátája (pl.: 20.1640239)

routes.txt – a viszonylatokra vonatkozó információkat írja le

`route_id`: a viszonylat azonosítója, minden rekordban különböző

`agency_id`: a viszonylatot üzemeltető szolgáltató azonosítója (az `agency.txt` fájlban létezőt kell megadni)

`route_short_name`: a viszonylat száma (pl.: 1, 1V, 3F, 84, 84A, 90, 90H, stb.)

`route_long_name`: a viszonylat jellege (pl.: körjárat, villamospótló, céljárat), és mivel a `route_type` mezőben a trolibuszokra nincs jelölés, ezért ezeknél a viszonylatoknál „trolibusz” értéket vesz föl

`route_desc`: a viszonylathoz kapcsolódó hosszabb megjegyzéseket tartalmazza, pl. forgalmi terelésekkel, jegyváltással, korlátozásokkal kapcsolatban.

`route_type`: a viszonylat járműve (0 - villamos, 2 - vonat, 3 - autóbusz)

`route_url`: a viszonylat menetrendi weblapjának a címét tárolhatja

`route_color`: a viszonylat színe, a viszonylatszám háttérszíne, Szegeden pl.: 0000FF (kék) – autóbuszvonalak, FFFF00 (sárga) – villamosvonalak, 00CC00 (zöld) – trolibuszvonalak

`route_text_color`: a viszonylatszám betűszíne, amely színvilágában igazodik a `route_color` mezőben kiválasztott színhez

trips.txt – a járatokra vonatkozó információkat írja le. A járat szó alatt az adott viszonylaton adott időpontban induló, adott útvonalon, adott menetidővel és közlekedési rend szerint közlekedő járművet kell érteni. A GTFS itt nem ennyire szigorú, nem követeli meg az adott indulás időpont szerinti szétválasztást. Én a magyar terminológia szerint, egyesével listáztam a járatokat, ez későbbi fejlesztés (pl. alacsonypadlós járatok jelölése) során járhat előnyökkel.

`route_id`: a járat viszonylatának azonosítója (a `routes.txt` fájlban létezőt kell megadni)

`service_id`: a járat közlekedési rendjének azonosítója (a `calendar.txt` fájlban létezőt kell megadni)

`trip_id`: a járat azonosítója, minden rekordban különböző

`trip_headsign`: a járat célállomása. Ez nem mindig egyezik meg az utolsó megállóhely nevével, hanem gyakran a járművek kijelzőin is megjelenített városrész neve, ahol a járat végállomása van.

`direction_id`: a járat irányát jelöli (0 – városközpontból vagy decentrumból kifelé, 1 – visszafelé)

`shape_id`: a járat útvonalának azonosítója (a `shapes.txt` fájlban létezőt kell megadni)

stop_times.txt – a megállóhelyi indulási időket tartalmazza. Általában ez a legnagyobb terjedelmű fájl, hiszen minden egyes járat indulása minden egyes megállóhelyről itt kerül rögzítésre.

`trip_id`: a járat azonosítója, amely megáll a megállóhelyen (a `trips.txt` fájlban létezőt kell megadni)

`arrival_time`: az adott járat érkezési ideje, azonban helyi közlekedésben ez megegyezik az indulási idővel, de a specifikáció miatt kötelező megadni

`departure_time`: az adott járat indulási ideje a megállóból (pl. 8:10:00). Az éjfélt átlépő járatok esetében az éjfélt utáni időpontokat folyamatosan, pl. 24:03:00 formátumban kell megadni.

`stop_id`: a megállóhely azonosítója (a `stops.txt` fájlban létezőt kell megadni)

`stop_sequence`: a megállóhely sorszáma az adott járat útvonalán. Ez gyakorlatilag megegyezik az adott járat menetidejével a megállóhelyig.

`pickup_type`: 1 – csak leszálló utasok részére (pl. feltételes megállóhelyen, végállomáson), 0 – nincs megkötés

`drop_off_type`: 1 – csak felszálló utasok részére (pl. végállomáson), 0 – nincs megkötés

`shape_dist_traveled`: az adott járat útvonalán a megállóig megtett távolság. Ez gyakorlatilag a járat útvonalát jelképező törtvonal (`shapes.txt`) pontjai közül a megállóhelynél található pontnak a sorszáma. Ebből következik, hogy az útvonalakat a megállóknál mindig meg kell törni. A beírt számnak meg kell egyeznie az útvonal pontokat tároló `shapes.txt` fájl ugyanezen mezőjéhez tartozó értékkel.

calendar.txt – a közlekedési rendekre vonatkozó információkat írja le

`service_id`: a közlekedési rend azonosítója, minden rekordban különböző

`monday`: közlekedik-e hétfőnként az adott közlekedési rend? (értéke: 0/1)

`tuesday`: közlekedik-e keddenként az adott közlekedési rend? (értéke: 0/1)

`wednesday`: közlekedik-e szerdánként az adott közlekedési rend? (értéke: 0/1)

`thursday`: közlekedik-e csütörtökönként az adott közlekedési rend? (értéke: 0/1)

`friday`: közlekedik-e péntekenként az adott közlekedési rend? (értéke: 0/1)

`saturday`: közlekedik-e szombatonként az adott közlekedési rend? (értéke: 0/1)

`sunday`: közlekedik-e vasárnaponként az adott közlekedési rend? (értéke: 0/1)

`start_date`: a közlekedési rend érvényességének kezdete (pl. 20110901)

`end_date`: a közlekedési rend érvényességének vége (pl. 20111231)

calendar_dates.txt – a közlekedési rendektől való eltéréseket írja le tanítási szünetek, nemzeti ünnepek, munkanap áthelyezések, stb. esetén. Ezeken a napokon a szokásos közlekedési rend helyett egy másikat alkalmaznak. Például munkanapi helyett munkaszüneti napi menetrendek vannak érvényben. A kivételt mindkét közlekedési rendhez be kell jegyezni!

`service_id`: a közlekedési rend azonosítója, amelyre vonatkozik az eltérés

`date`: a dátum, amelyik napra vonatkozik az eltérés (pl. 20111101)

`exception_type`: 1 – a közlekedési rend érvényes lesz a megjelölt napon, 2 – a közlekedési rend nem lesz érvényes a megjelölt napon

shapes.txt – a járatok útvonalait alkotó pontokra vonatkozó információkat írja le

`shape_id`: az útvonal azonosítója

`shape_pt_lat`: az adott útvonal egy pontjának szélességi koordinátája (pl. 46.2664451)

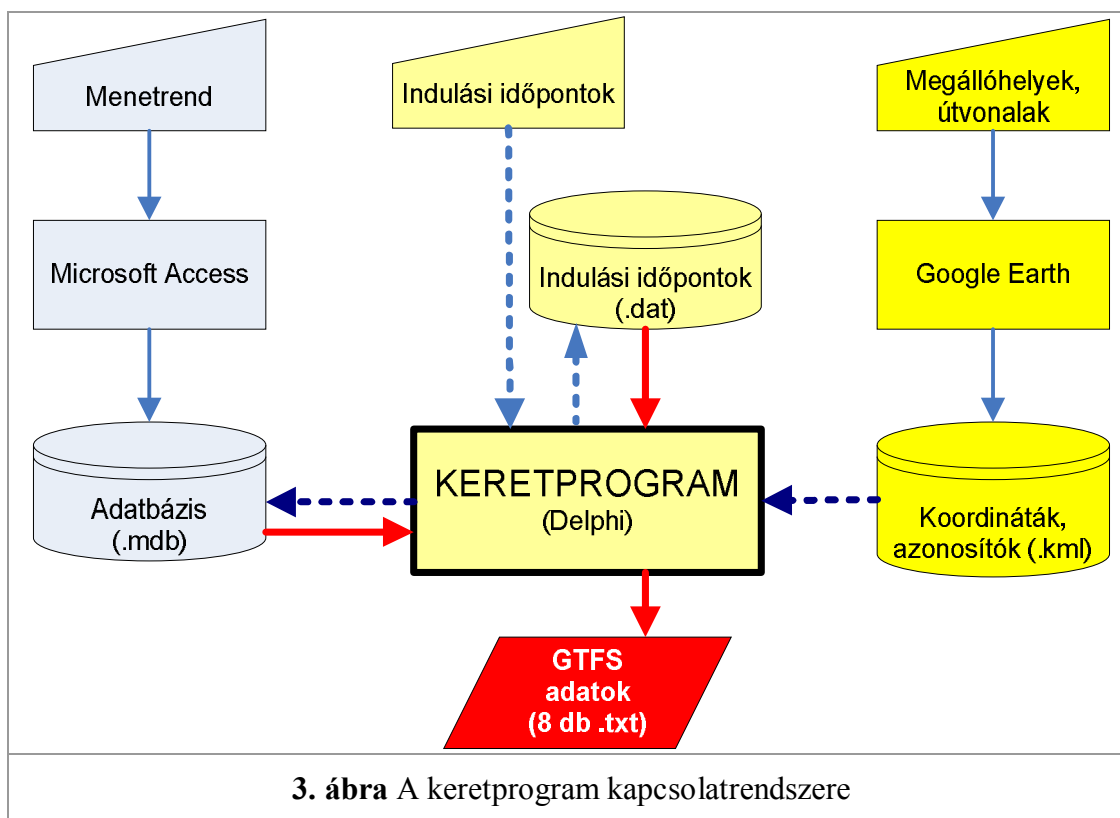
`shape_pt_lon`: az adott útvonal egy pontjának hosszúsági koordinátája (pl. 20.1266586)

`shape_pt_sequence`: az adott pont sorszáma az adott útvonalon

`shape_dist_traveled`: az adott útvonalon az adott pontig megtett távolság. Ez gyakorlatilag a járat útvonalát jelentő törtvonal adott pontjának a sorszáma, tehát a *shape_pt_sequence* mezőhöz tartozó számmal megegyezik. Ha az adott pont egy megállóhelynél található, akkor a mező értéke meg kell, hogy egyezzen a megállóhoz tartozó *shape_dist_traveled* értékkel (lásd `stop_times.txt`).

3 A keretprogram rendszere

A GTFS adatok előállításához három programra volt szükségem. Ezek közül egy, a Delphi, az tulajdonképpen egy fejlesztői környezet, amiben elkészítettem azt a keretprogramot, ami a másik két alkalmazást, a Google Earthöt és a MS Accesst köti össze. A kapcsolat felépítését a 3. ábra mutatja.



A Google Earthben lehet a megállókat és az útvonalakat bejelölni, majd a kész munkát egy XML-szerű formátumba elmenteni. Ebben a fájlban több tulajdonság mellett a számunkra fontos azonosítók és koordináták is szerepelnek. Mivel ezekkel a végleges GTFS szöveges állományok generálása előtt még egyéb műveleteket kell végezni (pl. az útvonalak pontjainak sorszámozása), ezért a földrajzi pozíciókat a keretprogram segítségével továbbítjuk a MS Access adatbázisba, közben megkapják a sorszámuakat, és így tároljuk azokat. Az Accessben űrlapokon keresztül az adatbázis többi részét is fel lehet tölteni a rendelkezésre álló információk alapján. Egyetlen kivételt a járatok indulási időpontjai jelentenek, mivel ezeket nem az időpontok egyenkénti begépelésével adjuk meg, hanem a helyi menetrendeknél szokásos formátumban, azaz egy adott órához tartozóan a percekkel együtt, vesszővel elválasztva. Ahhoz, hogy ezt az egyszerű beviteli formát biztosítsam, a keretprogramnak része egy indulási jegyzék, ahol az

adatbázisban már létrehozott járatcsoportok indulási időpontjait lehet bevinni, módosítani, vagy akár kitörölni. Ezeket viszont (történeti okokból) nem írtam bele az adatbázisba, hanem külön fájlban tárolom. A keretprogram fő feladata viszont a GTFS szerinti közösségi közlekedési adatok előállítása. Ebben a folyamatban felhasználja az adatbázist, kiegészítve az indulási jegyzéket tartalmazó állománnyal. Az így elkészült GTFS szövegfájlokat már csak össze kell tömöríteni, és végül az interneten publikálni.

3.1 A Google Earth szerepe

Ez a program azért lesz fontos számunkra, mert innen származnak a megállóhelyek és a járat útvonalak földrajzi pozíciói. A műholdfelvételek alapján, megfelelő nagyításban igen nagy pontossággal lehet szabadkézzel is dolgozni. A munkamenetről készült pillanatképet a 4. ábra mutatja.



4. ábra A Google Earth

Két objektumtípust kell használni: a helyjelzőt a megállókhöz (az ábrán a sárga rajzszög) és az útvonalat a járatokhoz (az ábrán a fehér és sárga vonalak). Előbbieket mindig az úttestre kell kitűzni (ez a későbbiekben számítani fog), és lehetőleg a megállótábla vonalában. Természetesen ilyen pontos információkkal nem rendelkeztem a város több mint 500 megállóhelyéről, így (GPS készülék híján) a bejelölésben a Norc [6] háromdimenziós utcaképei is segítettek. Azonban ezeket sem tudtam minden esetben fölhasználni, ugyanis a szolgáltatás a peremkerületekben már nem elérhető, illetve a fényképek néha már elavultak voltak. Ilyenkor személyesen mértem föl a terepet, fényképeztem, és kerestem olyan jellegzetes pontokat (pl. épületek, telekhatárok, bazaltbeton burkolat, stb.), amelyek a műholdfelvételeken kivehetők, és

ezekhez vonatkoztatva már be tudtam jelölni a megállóhelyeket. A helyjelzőnek egyetlen tulajdonságát, a név mezőt kell csak kitölteni. A műholdfelvételen történő bejelölés előtt a megállót az adatbázisban föl kell venni, és az ott kapott azonosító számot kell a mezőbe beírni. Egy megálló áthelyezésekor a kijelölt helyjelzőt az egérrel arrébb kell csak vonszolni.

Az útvonalak kijelölése földrajzi szempontból már egyszerűbb volt. Általában az utcák a Google Earthön jól kivehetők, több helyen még a sávok is látszódnak. Itt szembesültem azonban azzal a problémával, hogy a városról készült műholdképeket pontatlanul illesztették egymáshoz (kb. útszélességnyi elcsúszások adódnak néhol), illetve a képekre illesztett vektorgrafikus térkép sincs pontos fedésben az utcákkal. Erre első körben, a Google Earthön belül megoldást nem találtam, az 5. fejezetben, a továbbfejlesztések témakörében szerepel az áttérés az OpenStreetMap térképekre.

Az útvonal objektumok törtvonalakból állnak, ezeket a töréspontokat piros téglalapok jelzik (4. ábra). Fontos, hogy a megállóhelyeknél a kitűzött helyjelzőhöz is töréspontot kell tenni, ugyanis később így lehet majd a megállót az adott útvonalhoz hozzárendelni. Az útvonalak könnyen megváltoztathatók, jobb egérgattintással egy tetszőleges töréspont törölhető, illetve az útvonal közbenső módosítása, kibővítése is lehetséges. Az útvonal egyetlen fontos tulajdonsága is a név mező lesz. Ide egy tetszőleges nevet lehet beírni, de ugyanennek a névnek az adatbázisban is szerepelnie kell. (Utólag is fölvehető.) Az útvonal színe, illetve vonalvastagsága tetszés szerint beállítható. Ez a GTFS adatokban nem jelenik meg, csupán az útvonalak egymástól való vizuális elkülönítését szolgálja. Emiatt azonban inkább elrejtjük az útvonalakat, hogy csak azok látszódjanak, amelyekkel éppen dolgozunk.

Az elkészült munkát KML formátumba kell elmenteni. Ez egy XML alapú fájl, amiben minden útvonal és megállóhely adatai egy `<Placemark>` és egy `</Placemark>` sor között kerülnek kifejtésre. Egyes tulajdonságok közősek, mások csak az adott objektumnál fedezhetők fel. A feldolgozás során csak a nevek és a koordináták fognak szerepet játszani.

3.2 A MS Access szerepe

A KML fájlban tárolt földrajzi pozíciók a keretprogramban elindított művelettel átkerülnek egy Access adatbázisba. A legtöbb menetrendi információt (viszonylatokról,

járatokról, megállókról, közlekedési rendekről) ezzel az alkalmazással kell fölvenni, amelyeket a koordináták mellett szintén itt tárolunk.

3.2.1 Az adatbázis felépítése

Az adatbázis a GTFS fájlok (mint egyed típusok) szerinti felosztását követi. Ezek alapján a nyolc szövegfájlnak megfelelően nyolc normál tábla, illetve két kódtábla és két kapcsolótábla alkotja az adatbázist. A táblák közötti kapcsolatot a Melléklet mutatja.

A két kódtábla feladata az, hogy a GTFS szerinti kódolt adatokat a felhasználó könnyedén megadhassa. A *colors* tábla a színek nevét és a hozzájuk tartozó hat karakteres, hexadecimális számkódot tartalmazza, és ezzel a kóddal csatlakozik a *routes* tábla *route_color* és *route_text_color* mezőjéhez. A *tipus* tábla pedig a viszonylatokon használt járműfajtákat, és az ezekhez rendelt GTFS kódszámokat tárolja, amikkel a *routes* tábla *route_type* mezőjéhez kapcsolódik.

Az egyed típusokat megtestesítő táblák mezői szinte teljesen megegyeznek a GTFS fájlok mezőivel. Néhány kivétel azonban akad. A *shapes* táblából hiányzik a *shape_dist_traveled* mező, de ez megegyezik a *shape_pt_sequence* mezővel, tehát fölösleges is lett volna létrehozni. Ugyanezen okból maradt ki ez a mező a *stop_times* táblából is, a többször felhasználható adatokat csak egy helyen tároljuk. Az előbb említett két táblában egy-egy *id* mezőt is találunk. Ez azért szükséges, mert bár a táblák rekordjai egyébként is különböznenek (hiszen pl. a *shapes* táblában nem létezhet egy adott útvonalon két, azonos sorszámmal és azonos koordinátákkal jellemezhető pont), mégis kell elsődleges kulcsot definiálni. A *stop_times* és a *routes* táblákban is találunk egy-egy idegen kulcsot, a *route_stops_id* mezőt. Ez a két tábla összekötését szolgálja. A GTFS szerint ugyanis a megállóhelyi indulásokhoz (*stop_times*) rendeljük hozzá a járatokat (*trips*), ami egyszerű, egy-a-többhöz (1:N) kapcsolatot eredményezne. Viszont Szegeden (de általában országos szinten is) a menetrendi információk esetében inkább fordítva, a járatokhoz rendeljük hozzá a megállóhelyi indulásokat. A *stops* táblában a *stop_megj* mezőben a megállóhelyek egyértelmű azonosítására szolgáló megjegyzéseket tároljuk. Ezek nem a GTFS-hez kötődő információk, hanem a programot használó személy részére egyértelműsíti, hogy a több azonos elnevezésű megálló közül melyiket szeretné kiválasztani. Bár a GTFS lehetőséget nyújt a megállónevek kiegészítésére egyéb mezők felhasználásával, az általam használt megjegyzések egyrészt nem

hivatalosak (ilyen megállóhelyeket körülíró információkat nem is használnak a szolgáltatók), másrészt ezek rövidítések, éppen csak az egyedi azonosítást szolgálják – ezért nem kerülnek bele a GTFS szövegfájlokba.

A táblák közötti kapcsolatok elemzését követően két helyen külön kapcsolótáblákat kellett használni. Itt fontos megjegyezni, hogy a *trips* tábla nem a járat, hanem az úgynevezett járatcsoport egyed típus elemeit tartalmazza. A járatcsoport egy adott viszonylaton, azonos útirányon, azonos menetidővel, azonos közlekedési rend szerint induló járatok összességét jelenti. A kódtáblák és a normál táblák közötti kapcsolat egy-a-többhöz (1:N), hiszen egy rekordba csak egyetlen kódot lehet választani, egy kód viszont több rekordban is szerepelhet. Ezt a logikát követve egy viszonylathoz (*routes*) egy szolgáltató (*agency*), míg egy szolgáltatóhoz több viszonylat tartozik. Egy járatcsoport (*trips*) egyetlen viszonylathoz (*routes*), de egy viszonylathoz több járatcsoport is hozzárendelhető. Egy járatcsoport (*trips*) egyetlen közlekedési rend (*calendar*) szerint, azonban egy adott közlekedési rend szerint több járatcsoport is közlekedhet. Egy kivétel (*calendar_dates*) csak egyetlen közlekedési rendet (*calendar*) érint, ám egy közlekedési rendben több kivétel is lehet. Egy megállóhelyi indulás (*stop_times*) csak egyetlen megállóban (*stops*) értelmezhető, de egy megállóból nyilván több indulás is feltételezhető. A felsorolt esetek mind 1:N kapcsolatot jelentenek.

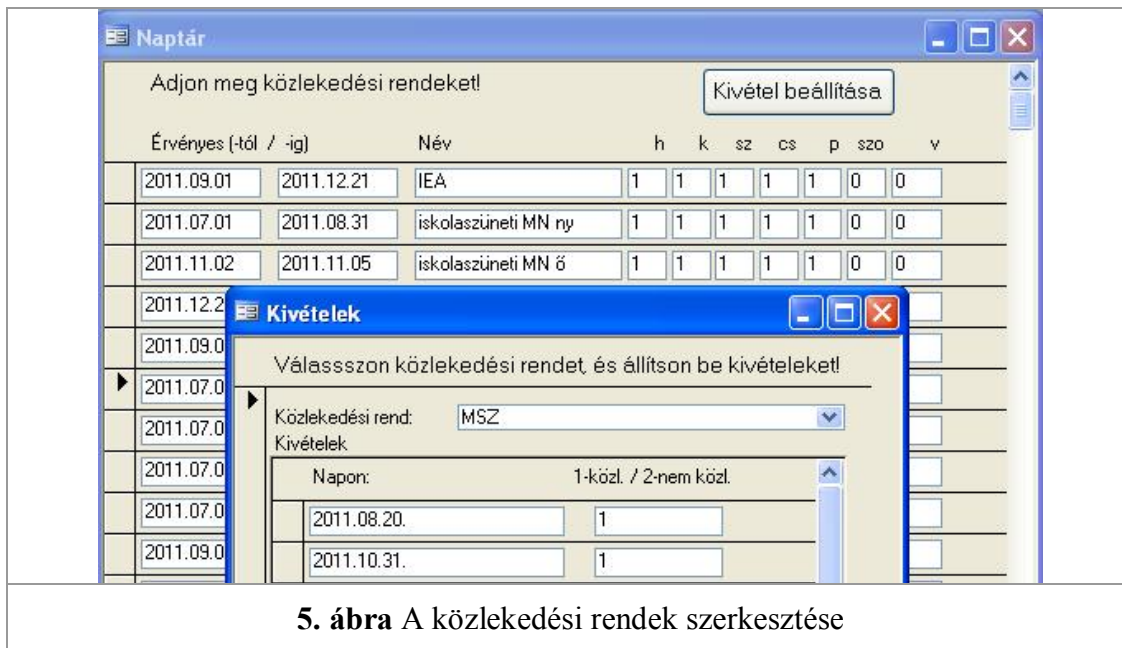
Egy járatcsoporthoz (*trips*) nyilván több útvonal pont (*shapes*), de egy ilyen útvonalpont szintén több járatcsoporthoz tartozhat. Ennek oka, hogy a járatcsoportokat például eltérő közlekedési rend alapján hozzuk létre, ettől viszont a két járatcsoportnak még ugyanaz lesz az útvonala. Hasonló a helyzet a járatcsoportok (*trips*) és a megállóhelyi időpontok (*stop_times*) esetében is. Ezt a két több-több (N:M) kapcsolatot kellett a kapcsolótáblákkal fölbontani két 1:N kapcsolatra. Az útvonal pontok (*shapes*) így már egyértelműen csak egy útvonalhoz (*shape*) tartozhatnak, és egy adott járatcsoport (*trips*) is csak egyetlen útvonalat (*shape*) használ. Ugyanígy a megállóhelyi indulásokat (*stop_times*) is egyetlen megállócsoporthoz (*route_stops* – bár ez az elnevezés nem éppen pontos ebben a formában) rendeljük hozzá, és a járatokat is egy megállócsoport jellemzi.

A mezők adattípusa szinte minden esetben szöveg (string), hiszen a GTFS fájlokba is szöveggént kerülnek az adatok, így a keretprogram típuskonverzióval keveset

foglalkozik. Másrészt a bevitt tömegközlekedési információ is szöveg (még a járatszám is), illetve a földrajzi pozíciókat is így érdemes feldolgozni, mert a koordináták tizedesponttal vannak elválasztva, és ezek átalakítása a tárolás kedvéért tizedesvesszőre értelmetlen, mivel a GTFS fájlokban is tizedespont az elválasztó. Egyedül néhány azonosító és idegen kulcs formátuma szám (hosszú egész – integer), illetve a *stop_times* tábla *stop_sequence* értéke, ami a sorba rendezéshez volt nélkülözhetetlen.

3.2.2 Az adatbázis használata

Az adatbázis megnyitása után rögtön egy menü fogad, ahol ki lehet választani, hogy milyen művelettel szeretnénk foglalkozni. A *Megállókódok* gomb mögött egy lekérdezés található, ami kilistázza azokat a megállókat (azonosítóval, névvel, megjegyzéssel együtt), amelyeknek hiányzik a koordinátája. Ez azt jelenti, hogy ezek a megállók a Google Earthben nincsenek még fölvéve, vagy onnan nem lettek az adatbázisba importálva. Ugyanakkor nagyobb számú megállóhely együttes felvételét is lehet a lekérdezésben bonyolítani, ekkor látható az összes új megálló azonosítója, így nem kell egyesével visszakeresni a kódokat, ha a Google Earthben csoportosan jelöljük be a megállókat. Ez főleg a kezdeti, tömeges adatbevitelnél jelentett számomra könnyebbséget, a későbbi, fenntartási időszakban, amikor már kevés új megállóhelyet kell fölvenni, ezt más módon is megtehetjük.



5. ábra A közlekedési rendek szerkesztése

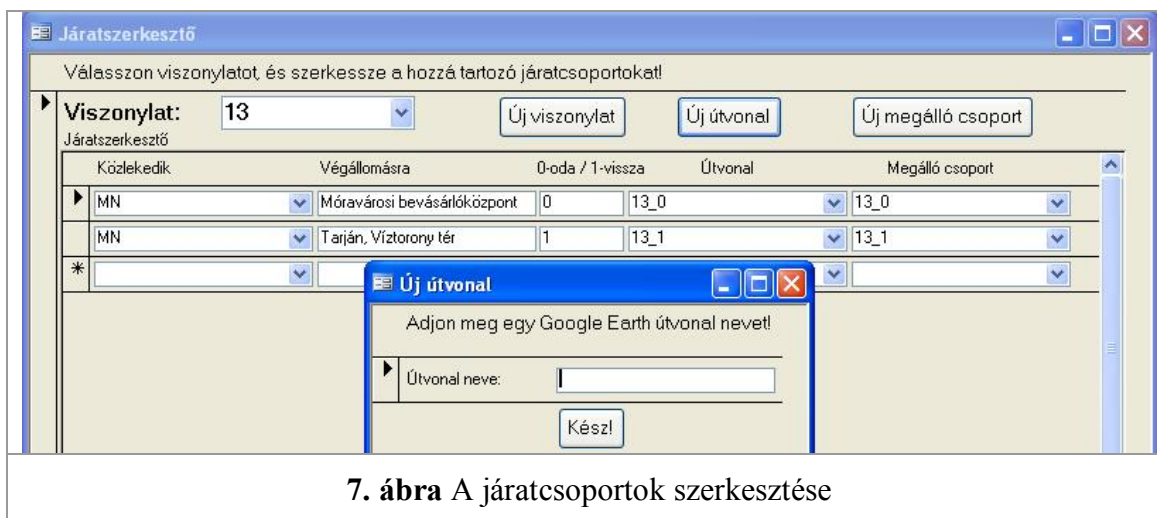
A tömegközlekedési adatok másik alapját jelentik a különféle közlekedési rendek. A főmenü *Menetrendi naptár* gombjára kattintva ezek az adatok is szerkeszthetők (5.

ábra). A hátsó, inaktív ablakban látható listában szerepelnek az egyes közlekedési rendek, amelyek érvényességi idejét beviteli maszkon keresztül adjuk meg (ÉÉÉÉHHNN) formátumban. A név elsődleges kulcs, minden sorban különbözőt kell megadni, de egyébként nincs más megkötés. Szeged esetében a szokásos elnevezéseket és rövidítéseket használtam. A rekordban be kell állítani, hogy a közlekedési rend általában (az érvényességi időszakon belül) mely napokon közlekedik. Az ünnepek, munkanap áthelyezések, stb. körüli forgalmirend-változásokat a *Kivételek* gombra kattintva lehet beállítani. Először a kombinált listából kiválasztjuk a módosítandó közlekedési rendet, és a megjelenő listába beírjuk a változás dátumát (szintén beviteli maszkon keresztül), illetve azt, hogy az adott napon a közlekedési rend a megszokottól eltérően érvényes lesz-e, vagy sem.

6. ábra A viszonylatok szerkesztése

A megálló és a menetrendi naptár birtokában következhet a viszonylatszerkesztés. Ez az ablak is a főmenüből indítható, képét a 6. ábra mutatja. Egy kombinált listából választható ki a szerkesztendő viszonylat, illetve a mellette található gombbal új viszonylatot is létrehozhatunk. A művelet ablaka ezzel megegyező, csak a viszonylatszámot egy egyszerű beviteli mezőben kell kitölteni. A jármű és a viszonylatot üzemeltető cég kombinált listából választható, ennek ismeretében az Access felkínálja a szegedi szokásoknak megfelelő színezést is. Ha ez nem tetszik,

akkor a legördülő listákban lévő színekből lehet válogatni. A többi mező értelemszerűen kitöltendő a GTFS értelmezése szerint (lásd 2.3 fejezet).



7. ábra A járatcsoportok szerkesztése

A viszonylat után nézzük meg a járatcsoportok szerkesztését (7. ábra). Ha eddig nem vettük volna föl a viszonylatot, akkor ez még itt is lehetséges az *Új viszonylat* gombra kattintva. A viszonylat létrehozása után egyből kiválasztható is lesz a legördülő listából. A járatcsoportok elkészítését a segédúrlapon végezhetjük, kiválasztva a közlekedési rendet a legördülő listából, valamint beírva a célállomást és a járat irányát a GTFS értelmezésnek megfelelően. Az útvonal és a megállócsoporthoz is legördülő listából választható, amennyiben még nincs fölveve, újat is létre lehet hozni a felső gombokra kattintva. Egyszerűen csak az új útvonal vagy megállócsoporthoz kell megadni a nevet. Itt még az sem probléma, ha az adott csoport nincs a Google Earthben megrajzolva vagy a megállók hiányoznak a csoportból.

A megállócsoporthoz szerkesztését a 8. ábra mutatja be. A már meglévő, vagy kattintás után (*Új csoport*) létrehozott megállócsoporthoz a segédúrlapon szerkeszthető. A menetidők, föl- és leszállást korlátozó számok egyszerűen begépelhetők, mi több a korlátozások alapértelmezett értéke 0, vagyis „nincs korlátozás”. A megálló legördülő listából választható ki, azonban ha a mezőbe elkezdjük a nevet gépelni, akkor az Alt+↓ billentyűkombinációval a megállók listája lenyílik, és az addig begépelte névrészletnek megfelelő megállókhoz ugrik. Látható, hogy a lista tartalmazza a megállónévhez fűzött megjegyzéseket is, amik segítségével gyorsan, egyértelműen beazonosítható, hogy melyik megállóra is van szükségünk. Egy már meglévő megállócsoporthoz a lista alján tovább bővíthető, a megállót a lista megjelenítésekor az Access automatikusan a

menetidő szerint fogja növekvő sorrendbe állítani. Újonnan létesített megállót az *Új megálló* gombra kattintva lehet fölvenni. Ekkor egy kis ablak ugrik föl, ahol ki lehet tölteni a megálló nevét és a magyarázó megjegyzést, valamint szürke (nem engedélyezett) mezőben látszódik a kapott megállókód is.



8. ábra A megállócsoportok szerkesztése

Egyedül a szolgáltatók (*agency*) táblájához nem készült űrlap, ugyanis ezek az egyedek ritkán bővülnek vagy szűkülnek. A táblaszerkesztőben viszont minden további nélkül bármelyik rekord tetszés szerint módosítható.

3.3 A keretprogram feladatai

A következő alfejezetekben ismertetésre kerülő három funkció közül mindegyikben közös, hogy a MS Access programmal valamilyen kapcsolatot kell létesíteni. A táblák szerkeszthetőségét, a lekérdezések keretprogramból történő indítását meg kellett oldani. Ehhez a Delphi adatbázis-kezelő részei közül az ADO és a BDE komponenseket használtam, amint keresztül a keretprogram jelenleg egy fixen beállított adatbázishoz van hozzákötve.

3.3.1 Importálás a Google Earthből

A *stops* és *shapes* tábláknál láthattuk, hogy a földrajzi pozíciókat tartalmazó mezőket nem töltöttük ki. Ezt a keretprogram fogja megtenni a Google Earthben fölvevett megállók és útvonalak alapján, az elmentett KML kiterjesztésű fájlból. A keretprogramban (9. ábra) a Google Earth → Importálás menüpontra kattintva megnyílik egy ablak (OpenDialog objektum), ahol ki lehet választani az importálandó állományt. Ezután

megkezdődik az adatátvitel a KML fájlból az adatbázisba. Mindezek előtt persze a kiválasztott KML állományt a keretprogramban szövegfájlként meg kell nyitni, az adatbázis kapcsolatot aktiválni, a *shapes* táblában lévő (régi) adatokat pedig törölni kell. Ezután következhet a KML fájl olvasása, amihez a következő kereső eljárást használtam:

```
keres:='<name>';           // keres:String
i:=1;                      // i:Byte;
repeat
  Read(g,x);               // g:TextFile; x:Char
  if x=keres[i] then inc(i)
  else i:=1;
until i=Length(keres)+1;
```

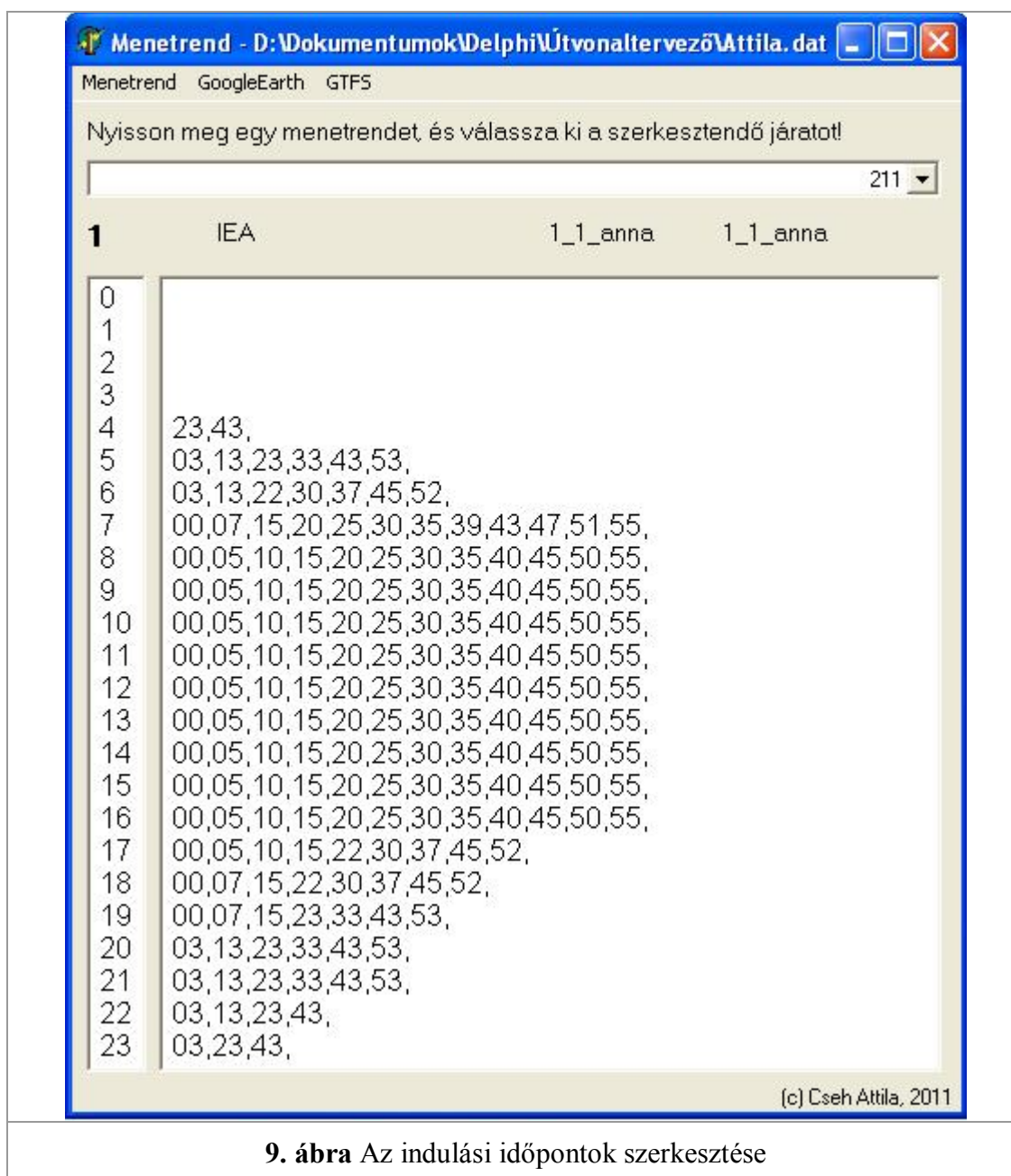
A fenti algoritmus a szövegfájlban a kurzort az első kódsorban aposztrófok közé írt szó mögé helyezi. Ezzel az eljárással először egy helyjelzőt (<Placemark>) vagy annak hiányában a fájl végét (</Document>) keresi meg a program. Amennyiben helyjelzőt talál, akkor a keresés folytatódik a <name> tulajdonságig, majd beolvassa az utána következő kifejezést. Ez lesz a megállóhely vagy az útvonal azonosítója. Ezután a szövegfájlt tovább olvasva a program eldönti (szintén a fenti kereső algoritmussal), hogy megállót, vagy útvonalat jelöl-e a <Placemark> kifejezés. Ennek tükrében más-más szóra rákeresve, de mindkét esetben megkeresi a földrajzi pozíciókat tartalmazó szövegrészt, és kiolvassa a szélességi és hosszúsági koordinátákat. Útvonal esetében ez addig tart, amíg a vonal összes töréspontját be nem olvassa. Ezekhez ráadásul olvasás közben a program hozzáfűz egy globális sorszámot, ami a *shapes* táblában elsődleges kulcsként funkcionál, illetve egy útvonal sorszámot, amit a *shape_pt_sequence* mezőbe kerül.

Egy-egy koordináta pár beolvasása után a begyűjtött (és az útvonal pontok esetében kiegészített) adatokat az adatbázis megfelelő sorába továbbítja a program. A *shapes* táblában folyamatosan töltődnek a rekordok, a *stops* táblában viszont a *FindKey()* függvénnyel meg kell keresni azt az egy rekordot, amire a koordináták vonatkoznak. A keresés természetesen a beolvasott azonosító alapján történik. Ezért fontos, hogy a Google Earthben a megállóhelyek és az útvonalak elnevezései megegyezzenek az adatbázisban tárolt megállókódokkal és útvonal nevekkal.

A sikeres importálást egy felugró ablak jelzi, illetve azt is, ha valahol az azonosítók nem egyeztek meg, így nem lehetett hova beilleszteni a földrajzi pozíciókat.

3.3.2 Az indulási időpontok szerkesztése

A keretprogram egyik leglátványosabb része a menetrendszerkesztő, a megjelenítés erre a funkcióra koncentrál (9. ábra). A két fő komponens az adatbázishoz kapcsolt legördülő lista, ami tartalmazza a már fölvetett járatcsoportokat, és a Memo típusú beviteli mező, ahová az ábrán látható módon, egymástól vesszővel elválasztva kell a megfelelő óra sorába az indulási időpontok perceit begépelni.



9. ábra Az indulási időpontok szerkesztése

A bevitt adatokat egy típus nélküli állományban (DAT kiterjesztéssel) tárolja a program. Az alkalmazott fájlstruktúrának köszönhetően az összes indulási időpont

(körülbelül 12000 db) elfért egy 100 kB nagyságú fájlban. A fájl csak a legszükségesebb, de annál fontosabb adatokat tartalmazza. 4 bájt foglal el a járatcsoport azonosítója (integer típus), és 1 bájt helyigényű (byte típus) az a szám (n), ami az adott járatcsoporthoz tartozó indulások számát jelzi. Itt föltételeztem, hogy legfőljebb 255 db indulás lehetséges egy nap. Végül következnek a TDateTime típusú (8 bájtos) indulási időpontok, amelyek összesen $n \times 8$ bájtot foglalnak le. Ennyiből áll tehát egy járatcsoport, maga a DAT fájl pedig az összes olyan járatcsoportot tartalmazza, amelyet az adatbázisban fölvettünk, és megadtuk az indulási időpontjait.

A keretprogram elindításakor a kapcsolat az adatbázissal rögtön felépül. A Menetrend → Megnyitás menüpontra kattintva egy (OpenDialog) ablakban ki kell választani az időpontok tárhelyéül szolgáló fájlt, vagy ha még nem létezik, akkor itt kell megadni a nevét. Ezután a kapcsolat fölláll a DAT állománnyal is, a legördülő menü aktívva válik, és benne megtalálható az összes járatcsoport azonosítója. Legördítéskor, és a kiválasztás után a járatcsoport kódszáma mellett megjelenik az azt jellemző viszonylatnév, közlekedési rend, valamint az útvonal és a megállócsoport azonosítója is, hogy a beazonosítás minél könnyebb legyen, és szerkesztés közben is tudjuk, éppen melyik járatcsoport időpontjaival dolgozunk. A legördülő menüből kiválasztva egy járatcsoportot vagy megjelennek a hozzá tartozó időpontok, vagy egy felugró ablak közli, hogy ezek még hiányoznak.

Amennyiben már léteznek a járatcsoporthoz időpontok, úgy a megjelenítés az alábbi módon zajlik. A program elkezdi olvasni a DAT fájlt. Az első 4 bájtból megállapítja az azonosítót, az 5. bájtból pedig, hogy hány időpont következik (n). Amennyiben az azonosító nem egyezik meg a legördülő listából választottal, úgy a program $n \times 8$ bájtot átugrik, és a következő járatcsoport azonosítójához ér, és a folyamat kezdődik előlről – amíg a fájl végét el nem érjük. Ha a járatcsoport azonosítók egyeznek, akkor a program elkezdi beolvasni a TDateTime típusú, 8 bájtos adatokat egy tömbbe. Így jön létre egy *menetrend* típusú rekord, amit az alábbiak szerint deklaráltam:

```

type menetrend=record
    trip_id:Integer;
    meret:Byte;
    ido:array[0..255] of TDateTime;
end;

```

Innen konvertálja egyesével szöveggé, és választja szét az óra és perc részekre az időpontokat. 0 órától kezdve a számolást, amíg az időpontok óra adatai ezzel a számmal megegyeznek, addig a perc értékek egymás után, vesszővel elválasztva halmozódnak. Amint az időpontok órát ugranak, az eddig összegyűjtött percek bekerülnek a Memo ablak aktuális sorába, és a program is számol egyet. Innentől kezdve a folyamat ismétlődik, és egészen addig tart, amíg a program 23 óráig el nem számol. Ennek a végén az utolsó sor is kitöltésre kerül a vesszőkkel elválasztott perc értékekkel.

Az időpontok szerkesztésének befejezésekor a Menetrend → Mentés menüpontra kattintva végezhető el az aktuális járatcsoport mentése. Ekkor az előző bekezdéshez hasonlóan a program elkezd olvasni a DAT fájlt, de annak teljes tartalmát járatcsoportonként egy *menetrend* típusú tömb rekeszeibe helyezi (puffer tömb). Az egyedüli kivételt az aktuálisan szerkesztett járatcsoport jelenti, ez természetesen nem kerül a pufferbe. Ellenben a Memo ablakban található perc értékeket a program elkezd soronként beolvasni, majd a vesszők segítségével perc értékekre bontani, végül hozzátéve az óra értéket is (ami megegyezik a sornak a sorszámaival) TDateTime típusú adatokat képez. Ezeket összeszámolja, hozzájuk teszi a darabszámot és a járatcsoport azonosítót, és ezzel elkészül egy *menetrend* típusú adat. A mentés végén pedig a DAT fájlt újraírva (Rewrite) a pufferből a járatcsoportok visszakerülnek a fájlba, legvégére pedig az aktuálisan elmentett járatcsoport kerül.

Ha az aktuálisan megnyitott járatcsoportot ki szeretnénk törölni, akkor a Menetrend → Törölés menüpontra kattintva az előbbi folyamat játszódik le azzal a különbséggel, hogy a megnyitott járatcsoportból nem képez a program *menetrend* típusú rekordot, és csak a pufferben lévő járatcsoportok kerülnek vissza a fájlba (ami ugye nem tartalmazza az aktuális járatcsoportot).

3.3.3 A GTFS alapú adatok generálása

A keretprogram legfontosabb feladata, hogy az előző fejezetekben föl vitt és eltárolt információk kombinációiból előállítsa a nyolc GTFS szerinti szövegfájlt. Az *agency.txt*, *stops.txt*, *routes.txt*, *shapes.txt*, *calendar.txt* és *calendar_dates.txt* állományok esetében

ez egyszerű, hiszen az adatbázis azonos nevű táblái már a GTFS-nek megfelelő mezőkkel lettek megalkotva. A keretprogram rekordonként kiolvassa a szükséges mezőket, és beleírja a fájlba. Ellenőrzésképpen azoknál a mezőknél, amelyeknél az adatbázis-kezelőben nem lett beállítva a kötelező kitöltés, ott a megvizsgálja, hogy üres-e az adott rekord. Ha igen, akkor nem is kéri le az adatot, hanem átugrik az adott rekord következő mezőjére.

A *trips.txt* generálásánál meg kellett oldani, hogy a járatcsoportonként tárolt információk a szövegfájlba már egyesével, járatonként kerüljenek kifejtésre. Ezért itt a folyamat az indulási időpontokat tartalmazó DAT állomány megnyitásával kezdődik. A beolvasott járatcsoport azonosító alapján egy egyszerű lekérdezéssel az adatbázisban a program megkeresi a járatcsoportot. Innentől kezdve minden egyes indulási időpont (azaz tulajdonképpen járat) beolvasása után a járatcsoporthoz tartozó mezőket beírja a szövegfájlba. Egyedül a járatcsoport azonosítót egészíti ki egy számmal, mégpedig az adott indulási időpont (járat) sorszámaival a járatcsoporton belül. Ez a ciklus addig tart, amíg a program a DAT fájl végére nem ér. Így lesz minden járatcsoporthoz tartozó járatnak egyedi azonosítója.

Ez a tulajdonság a *stop_times.txt* előállításánál használható ki, ami egyben a legösszetettebb módon szerkesztett szövegfájl a GTFS állományban. A keretprogram ebben az esetben is először a járatcsoport azonosítót (*betolt.trip_id*) nyeri ki a DAT fájlból. Ennek birtokában a járatcsoporthoz tartozó rekordokat lekérdezi az adatbázis *stop_times* táblájából:

```
SELECT stop_times.stop_id, stop_times.stop_sequence,  
stop_times.pickup_type, stop_times.drop_off_type, trips.trip_id  
FROM (route_stops INNER JOIN stop_times ON  
route_stops.route_stops_id = stop_times.route_stops_id)  
INNER JOIN trips ON route_stops.route_stops_id =  
trips.route_stops_id  
WHERE (((trips.trip_id)=IntToStr(betolt.trip_id)))
```

Ezzel a lekérdezéssel megkapjuk az adott járatcsoporthoz tartozó megállóhely azonosítókat, menetidőket, fel- és leszállási korlátozásokat. A *stop_times.txt* fájlba viszont szükség van még a *shape_dist_traveled* értékekre is, ami alapján a járat útvonalához hozzákapcsoljuk a megállóhelyeket. Ezért a keretprogram még két, az előbbihez hasonló lekérdezést küld az adatbázis *stops* és *shapes* tábláihoz: az aktuális járatcsoport megállóhelyeinek és útvonalainak a földrajzi pozícióira, illetve az útvonal

koordináták sorszámaira (*shape_pt_sequence* = *shape_dist_traveled*) vonatkozóan. Ez utóbbit kapják meg a járat útvonalának töréspontjainál fekvő megállóhelyek is a *stop_times.txt* fájlban.

A birtokunkban lévő rengeteg információt a keretprogram fésüli össze. A DAT fájlból betöltött járatcsoport első időpontját beolvassa, majd a *stop_times* tábla lekérdezéséből az egyik megállóhely (rekord) menetidejét (*stop_sequence*) hozzáadja. Az így kapott megállóhelyi érkezési és indulási időpont kerül bele a szövegfájlba, amit az adott rekord többi mezője követ. A megállóhely földrajzi pozícióit a *stops* tábla lekérdezéséből keresi ki a keretprogram a megállóhely azonosító alapján, majd a *shapes* tábla lekérdezéséből az ehhez a megállóhelyi koordinátapárhoz tartozó útvonal pont *shape_dist_traveled* értékével tér vissza, ami így beírható az adott megállóhely *shape_dist_traveled* értékeként. A járat azonosítóját a *trips.txt* fájlhoz hasonlóan a járatcsoport azonosító és a hozzáírt sorszám együttesen alkotja.

Az előbbi folyamat során még csupán a *stop_times.txt* fájl egyetlen sorát hoztuk létre, azaz egy járat (indulási időpont) egy megállóhelyről történő indulását. A ciklikus működés során a program a kiválasztott járáshoz (indulási időponthoz) tartozó összes többi megállóhelyére elvégzi az előző bekezdésben leírtakat (a *stop_times* tábla lekérdezésének utolsó rekordjáig), majd a következő indulási időpontra ugrik. A járatcsoport összes indulási időpontjának (járatának) összes megállóhelye után a következő járatcsoport kerül feldolgozásra egészen addig, amíg a DAT fájl végére nem érünk. Ezek után nem meglepő, hogy a GTFS állománynak a *stop_times.txt* lesz a legterjedelmesebb fájlja, hiszen három ciklus fut egymásban: a járatcsoportok (a fájl végéig fut) → az indulási időpontok (a járatcsoport végéig fut) → a megállóhelyek (ahány megálló tartozik egy járáshoz).

A folyamathoz még hozzátartozik a koordináták lebegőpontos számmá konvertálása (az Accessben mindent szöveggként tárolunk!), a megállóhelyek és útvonalak kereső algoritmus, illetve annak a tőrésnek a biztosítása, amit a megállóhelyi koordináták és az útvonal töréspontok koordinátái között kell hagyni. Ezért fontos, hogy a Google Earth alkalmazásban minél pontosabban jelöljük be az útvonalakat, illetve megállóhelyeket.

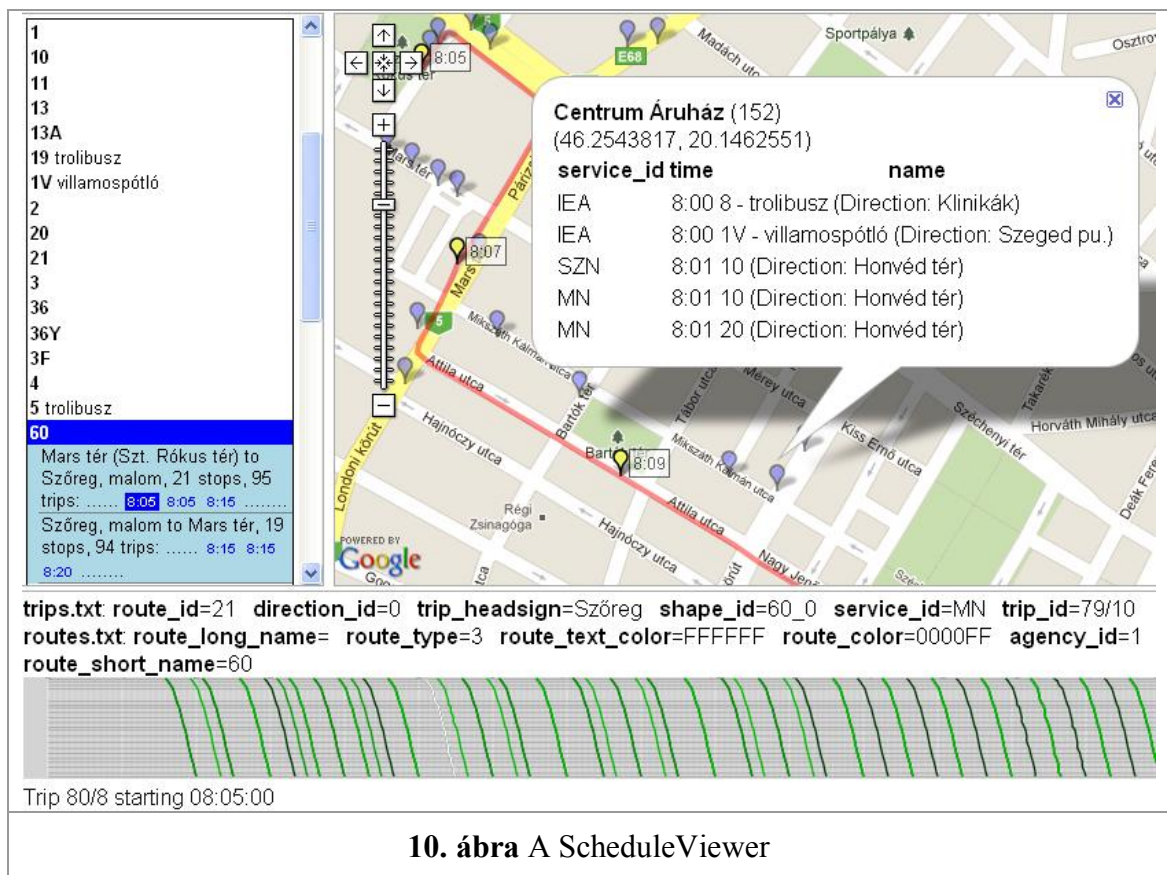
3.4 A publikálás

Ahhoz, hogy ezt a „legyártott” GTFS állományt a végfelhasználókhöz eljuttassuk, különböző feladatokra van még szükség. Mindenekelőtt a Google javaslata szerint [4] legalább két ellenőrzést érdemes elvégezni, mielőtt ténylegesen a publikálásra kerülne a sor. Ehhez két segédprogramot is rendelkezésünkre bocsát, amelyek a *FeedValidator* [7] és a *ScheduleViewer* [8]. Mindkettő egy parancssorban megnyíló alkalmazás, egyetlen teendőnk annyi, hogy a GTFS állományt egérrel a parancssorba vonszoljuk, ekkor elindul az ellenőrzés.

A *FeedValidator* a GTFS szabványnak való megfelelést vizsgálja formai és tartalmi szempontokból. Az ellenőrzés végén egy HTML fájlt nyit meg, amiben hibát jelez, vagy figyelmeztetést küld. Előbbi esetben az állományt nem érdemes publikálni, mert valószínűleg a felhasználó alkalmazások nem fognak helyesen működni. Ilyen probléma lehet, ha hiányzik valahonnan a mezőket elválasztó vessző, vagy üresen hagyunk kötelezően megadandó értéket, esetleg a megadott érték a GTFS szerinti értékkészleten kívülre esik, vagy értelmetlen. A szegedi adatok előállításában leggyakoribb hiba az volt, hogy amikor a megállóhelyi koordináták (*stops* tábla) és az útvonal koordináták (*shapes* tábla) nem találkoztak, akkor a megállóhelyi indulási időknél (*stop_times*) a *shape_dist_traveled* mező értéke üresen maradt. Ha csupán figyelmeztet a *FeedValidator*, akkor az adatok a GTFS-nek megfelelnek, azonban magukban a megadott információk tartalmi helyességében kételyek merülhetnek föl. Például a program jelez, ha egy megállóhelyet vagy útvonalat egyetlen járat sem használ. Ez gyakran előfordulhat forgalmi terelések esetén, természetesen az adatok ilyenkor helyesek, nyilván a járatok más útvonalat és más megállóhelyeket vesznek igénybe, a GTFS állomány minden további nélkül publikálható. A *FeedValidator* a fentiek mellett néhány statisztikai jellegű adatot is közöl a viszonylatokról, megállóhelyekről, járatokról, stb.

A *ScheduleViewer* egy egyszerű megjelenítő program. A futtatás végén szintén egy HTML oldalt kell megnyitni (ami a számítógép helyi kiszolgálójára hivatkozik). A megjelenő képet a 10. ábra mutatja. Alapállapotban a GTFS adataink lefedettségét területének térképe jelenik meg, ezt nagyítva az összes megálló kis kék léggömbként látszódik. Bal oldalon a viszonylatok sorakoznak, ezek közül valamelyiket kiválasztva megjelennek a hozzá tartozó járatok, azok közül pedig egy előzőleg megadott

időponthoz képest a leghamarabb induló ki is lesz jelölve. A kijelölés a térképen is követhető, az adott járat útvonala piros csíkként látható, míg az érintett megállók sárga színűvé változnak. Egyúttal a megállók jelzik is az adott járat indulási idejét. A térkép alatt további járat és viszonylat információk olvashatók, sőt még a viszonylat napi menetgrafikonja is kirajzolódik legalul. Egy megállóhelyet jelképező ikonra kattintva egy buborékban feltűnnek a megállóhely főbb tulajdonságai, alattuk pedig az adott időponthoz képest leghamarabb induló öt járat közlekedési renddel, indulási időponttal, járatszámmal és irányjelöléssel.



10. ábra A ScheduleViewer

A ScheduleViewer feladata befejezni azt, amit a FeedValidator esetleg nem tudott. Ehhez a felhasználó személy képességeit használja föl, hiszen némelyik tartalmi hibát az ember csak akkor veszi észre, amikor már szinte „élesben” működik a rendszer. Különösképpen fontos ez a saját megjelenítő, hiszen esetünkben az adatok előállítója nem tudja, hogy ki milyen alkalmazásban használja föl a GTFS állományt. A ScheduleViewerben böngészve viszont könnyen áttekintheti az adatokat.

Az ellenőrzés, és az esetleges korrekciók után következhet az adatok nyilvánosságra hozása. Ennek legegyszerűbb módja a tömörített ZIP fájl föltöltése egy nyilvános

hozzáférésű szerverre, és egy erre mutató link elhelyezése egy weblapon. A Google és a többi felhasználói alkalmazás is erről a címről fogja időnként az adatokat letölteni, úgyhogy a címet nem érdemes gyakran megváltoztatni. Megkönnyíthető a felhasználók dolga, ha a honlapon úgynevezett RSS hírcsatornát helyeznek el. Ez egy XML formátumú weboldal, amin keresztül a felhasználók folyamatosan értesülhetnek a legújabb változásokról. Amennyiben föliratkoznak az RSS hírcsatornára, úgy nem szükséges rendszeres időközönként letölteniük a GTFS adatokat, hanem csak akkor, ha az RSS jelzi, hogy frissült a szerveren tárolt állomány. Természetesen az RSS üzemeltetése ez esetben a GTFS adatokat publikáló személynek a feladata, azonban az XML formátumú weblapnak a fenntartása egyszerű szövegszerkesztő programmal is megoldható.

A TDK dolgozat keretében elkészült GTFS állományt a http://vonalhalozat.fw.hu/google_transit.zip címen helyeztem el. A <http://vonalhalozat.fw.hu/> weboldalon található a link a fájlhoz, illetve az RSS oldalhoz, ahol föl lehet iratkozni a hírcsatornára. Mindezt azonban ki lehet egészíteni azzal, hogy a világhálón létező GTFS gyűjtőoldalakon [9] [10] bejelentjük az adatokat az elérhetőségükkel együtt, illetve ha a Google Térképen is viszont szeretnénk látni az adatokat, akkor a Google-lel is föl kell venni a kapcsolatot.

4 Tapasztalatok a szegedi adatok feldolgozása közben

A keretprogram fejlesztése – főképp az utolsó időszakban – szorosan összefüggött Szeged város helyi közforgalmú közösségi közlekedési adatainak bevitelével a rendszerbe. Ennek során több furcsasággal is találkoztam, amelyeket a hétköznapi életben észre sem veszünk, vagy magunk is már beletörődtünk, azonban az adatbázis-kezelés szigorú szabályrendszerében rögtön kibuknak.

Ebből a szempontból a legsúlyosabb dolognak a kettős elnevezésű megállóhelyek tekinthetők. Ennek oka a két szolgáltató jelenlétében, és az általuk üzemeltetett viszonylatok különbözőségében keresendőek. A város főpályaudvarával „Szeged pályaudvar” és „Személypályaudvar” néven is találkozunk. A Mars téri központi buszállomáson a trolibuszok egyetlen pár megállót használnak („Mars tér” elnevezéssel), az autóbuszok viszont többet is (a trolibuszokét is). Mivel a tér minden oldalán van autóbusz megállóhely, így mindegyik kapott a „Mars tér” mellé kiegészítő elnevezést, például „Mars tér (Nagykörút)”, amit sajnos a trolibuszok nem vettek át. Bár a megállóhelyek neveiről a városüzemeltetés dönt, ezek egységesítése lassan halad, sőt, a legújabb viszonylatok is eltérő néven használják ugyanazt a megállóhelyet, amit a többi viszonylat évtizedek óta másképp hív. A GTFS adatbázisban jelenleg egy megálló, csak egyféleképpen szerepel. Ez nem okoz értelmezésbeli zavart, hiszen nagy elnevezésbeli különbségek szerencsére nincsenek.

Szintén a megállónevekhez kapcsolódik a folyamatos névmódosítás. A mára semmitmondó „Szűcs kovács” megállót előbb „Szőreg, Szűcs kovács”, majd újabban (ahogyan a helyközi menetrendben is szerepel) „Szeged (Szőreg), Szűcs kovács” néven „forgalmazza” a szolgáltató. Mivel a legújabb név is ugyanolyan semmitmondó (hiszen a kovácsműhely már nem létezik), mint elődei, és hivatalos értesítést nem adtak ki az átkeresztelésről, így a GTFS adatbázisban jelenleg az eredeti név szerepel.

A járatok utolsó megállóhelyeinek és célállomásainak elnevezése ugyancsak konfliktusforrás. Ahogy korábban már említettem, ezek helyes módon nem mindig egyeznek meg. Főleg külvárosi viszonylatoknál a célállomás lehet egyszerűen a peremkerület neve (ezeket szokták a viszonylatkijelzőkön is megjeleníteni). Sajnos a kiadott menetrendkönyvben az utolsó megálló neve gyakran mégis a peremkerület neve (pl. „Szentmihály”), holott az előző megállók is mind Szentmihály városrészben

találhatók. Szerencsére a viszonylatleírásokból kiderül, mi ezeknek az utolsó megállóknak a becsületes neve (pl. az előbbi esetben: „Harcos u.”), így a GTFS adatbázis ezeket használja. Néhány esetben a hivatalos elnevezések a földrajzi nevek helyesírásának ellentmondtak, így ezeket is javítani kellett.

A megállóhelyeken kívül a menetidők is kisebb problémaforrást jelentettek. A különböző, mégis közös szakaszon haladó viszonylatok menetideje gyakran eltér. Még olyan esetre is van példa, amikor még a viszonylatok szétválása előtt, a végállomástól két megállónyira három viszonylatnak háromféle menetidőt írnak elő ugyanarra az útra. Ezek valamilyen mértékben egységesítve lettek.

Nagyobb gondot okozott, hogy a Szegedi Közlekedési Kft. legújabb internetes menetrendjein nem tüntetik föl a menetidőket. Ennek egyik oka az lehet, hogy többféle menetidőt is használnak, erre azonban kétlem, hogy az a helyes reakció, hogy az interneten nem írnak ki még csak tájékoztató jellegű menetidőt sem. Így egyelőre a régebben érvényes általános menetidőt vettem figyelembe, holott a programom képes az eltérő menetidők feldolgozására.

Mindkét szolgáltatóra jellemző azonban, hogy a menetrendi hirdetményeket gyakran későn, az életbe lépésük előtt csupán egy-két nappal hozzák nyilvánosságra. Még ha a módosított GTFS adatok publikálása aznap meg is történik, akkor sincs garancia arra, hogy minden felhasználói alkalmazás frissíteni fogja egyből az adatbázisát. Így könnyen előfordulhat olyan eset, hogy az utazók helytelen tömegközlekedési adatok birtokába kerülnek.

Akadt viszont olyan dolog is, ahol a GTFS jelentette a szűk keresztmetszetet. Nevezetesen az alacsonypadlós járatok kérdése. Bár mindkét szolgáltató becsületesen meghirdeti, frissíti (és be is tartja) az alacsonypadlós járatok indulását, ezeknek a jelölését GTFS szerint egyelőre nem lehet megoldani. A fejlesztői fórumokon is fölmerült már ez a probléma, erre vonatkozóan várhatóan új mezőkkel fog bővülni a specifikáció.

5 Lehetséges alkalmazások, fejlesztési lehetőségek

Az előző fejezetben tapasztalt anomáliák miatt érdemes lehet fölvenni a kapcsolatot a szegedi tömegközlekedésért felelős városi szervezettel. Ez egyelőre nem történt meg, ugyanis a keretprogram még kiegészítésre szorul a frissített adatok kezelése, részben ehhez kapcsolódóan a jelenleg fix adatbázis rugalmas kiválasztása, illetve a tarifarendszer felvitelének terén.

A GTFS adatok legnyilvánvalóbb felhasználója, a Google Térkép viszont nem fogad akárhonnan fájlokat. Különböző okok miatt csak a szolgáltatók vagy az üzemeltetésért felelős hivatali szervek által publikált GTFS adatokat jelenítik meg térképükön. Innentől kezdve még fontosabbá válik az együttműködés annak érdekében, hogy Szeged fölkerülhessen Budapest mellé, mint a hivatalos adatokat közzétevő magyar város. A Google Térképen való megjelenés azonban kétségtelenül hatalmas előnyökkel jár, ezeket a 2.1 fejezetben már kifejtettem.

A másik ok, ami miatt a szolgáltató cégek bevonása fontos, az a menetrendi információk közlése. Jelenleg az elkészült keretprogram kézi adatbevitel alapján állítja elő a GTFS szerinti fájlokat. Ez egy Szeged méretű város esetében még-még elviselhető – bár így is viszonylag sok munkaórát jelentett. Napjainkban viszont már minden vállalat rendelkezik saját informatikai adatbázissal, amelynek outputja lehet akár a menetrendeknek valamilyen „olvasmányos” fájlba történő exportálása (pl. XML). Ezt már a Google Earth fájl olvasásához hasonlóan föl tudná dolgozni a keretprogram, és az adatbázis egy részét gyorsan, egyszerűen föl lehetne tölteni. Ugyanakkor mindent ez sem oldana meg egy csapásra, hiszen városunkban a két szolgáltató valószínűleg másfajta adatbázist használ (ráadásul az előző fejezetben leírt hibákkal tarkítva), így a tőlük kapott adatok közül, legnagyobb biztonsággal csak a járatok indulási időpontjait tudnám egyelőre átvenni (kiváltva a jelenlegi DAT fájlt).

Az elkészített GTFS állomány „nem hivatalos” státusza szerencsére nem jelent mindenhol akadályt. Arra már a 2.1 fejezetben is utaltam, hogy a Google Térképen kívül más alkalmazások is képesek a GTFS szerinti adatokat fogadni. Köztük léteznek nyílt forráskódú programok, amelyek aztán személyre szabhatók, igény szerint módosíthatók. Welker Zsombor sokat fáradozott ezek magyarosított példányainak a kifejlesztésével, illetve az ugyancsak nyílt forrású térkép, az *OpenStreetMap* (OSM)

adatainak összeházasításával a GTFS-sel. Az ő javaslatai alapján vázolom a keretprogramom fejlesztési irányát, és bemutatom a Zsombor által elkészített, és többek között a szegedi GTFS adatokkal feltöltött utastájékoztató alkalmazásokat is.

Az egyik lehetséges út a Google Earth leváltása az OpenStreetMapre. Ez technikailag nem jelent túlságosan nagy feladatot, hiszen az OSM-ből exportált fájl szintén XML-szerű, tehát beolvasása a keretprogrammal megoldható. Átalakításokra viszont szükség lenne, ugyanis míg a Google Earth fájl szinte csak a megállóhelyek és útvonalak adatait tartalmazta, addig az OSM fájl a letöltött térkép valamennyi objektumának földrajzi pozícióit és más attribútumait is magába foglalja. Ráadásul az adatok nem objektumonként szerepelnek az állományban, hanem adatbázisszerűen, tehát az alakzatok (hivatkozott) vonalakból, a vonalak (hivatkozott) pontokból állnak, és csak a pontoknál jelennek meg a koordináta tulajdonságok. Ez némiképp összetettebb feldolgozást igényel a keretprogramom részéről, illetve felhasználás előtt érdemes az objektumokat szűrni aszerint, hogy csak a tömegközlekedéssel kapcsolatosak maradjanak a fájlban.

Cserébe az OSM számtalan előnnyel kecsegtet a Google Earthhez képest. Mivel az OSM egy térképszerkesztő program, így a közösségi közlekedési útvonalakat a meglévő utcahálózaton jelöljük ki, így egy útvonaltervező, vagy vonalhálózatot megjelenítő alkalmazásban a járatok és az utcák egymással tökéletesen fedésben lesznek. A jelenleg használt Google Earth esetében az útvonalakat kézzel kell bejelölni, a térképi objektumok fölhasználása nem lehetséges, így kisebb-nagyobb pontatlanságok (főleg nagyított térképnézetben) előfordulnak. Az OSM szerkeszthetősége azzal is jár, hogy a változásokat (pl. új körforgalom építése) gyorsabban be lehet vezetni, mint a Google Térképen, így az OSM gyakran pontosabb információkat jelenít meg. Ez a különbség főleg a gyalogos infrastruktúránál jelentkezik, hiszen az OSM térképet bárki szerkesztheti, könnyen fölviheti a helyi viszonyoknak megfelelő közlekedési folyosókat.

A Zsombor által magyarosított, még nem végleges, de már működő egyik alkalmazás az *OpenTripPlanner* [11]. Ez egy útvonaltervező, amely az OSM térképet alapul véve az adatbázisát jelentő BKV, MÁV, Volán, szegedi és máshonnan származó adatok felhasználásával már országos szinten nyújt útvonalajánlatokat a tömegközlekedést

választóknak. Erre jelenleg nem sok másik internetes térkép képes – még a Google Térkép sem. Azonban tekintettel arra, hogy ez a fejlesztés nyílt forrású alapokon nyugszik, illetve magánemberek segítségével készül, így nem ódzkodnak nem hivatalos adatokat is fölhasználni. Az alkalmazás fejlesztése javában zajlik. Jelenleg az útvonaltervezés mellett működik a megállóhelyek térképes megjelenítése, a megállóból hamarosan induló járatok feltüntetése, a teljes megállóhelyi menetrend megjelenítése, illetve a viszonylatok kijelzése – mint vonalhálózati térkép.

Egy másik hasonló alkalmazáscsomag a *OneBusAway* [12]. Tudása a valamivel elmarad az OpenTripPlannertől, hiszen útvonalat nem tud tervezni. Viszont nem is erre a célra szánták, hanem arra, hogy ha a városban járva tudni szeretnénk valamelyik buszmegállóból induló buszok menetrendjét, akkor azt könnyen lekérdezhessük, akár mobiltelefonnal is. Ennek érdekében a OneBusAway alkalmazásnak létezik iPhone-os, androidos és Windows Phone 7-es változata is. Ezek mellett támogatja a régebbi telefonok hagyományos (wap) böngészőit is a tisztán szöveges kijelzéssel, illetve a telefonos, SMS alapú és természetesen a webes hozzáférést is.

Hasonló, hasznos alkalmazás Android telefonokra, a TinyAppWorks által fejlesztett, ingyenes *Budapesti menetrend* [13]. Ez az offline menetrend a BKK hivatalos GTFS adatbázisa alapján működik. A kezdőoldalon felsorolhatjuk a kedvenc járatainkat, megállókat vagy az adott megállóból az adott irányba induló járatokat. Böngészhetünk és kereshetünk az összes járat és megálló vagy az aktuálisan legközelebbi megállók listájában. Egy kiválasztott megállóban az alkalmazás megtalálja az összes onnan induló járat nevét és irányát. Egy kiválasztott járat esetén pedig megadható, hogy melyik megállóból melyik irányba közlekedő járművek indulási idejét szeretnénk listázni az aktuális időponttól vagy egy kiválasztott napon.

Legújabban az iPhone telefonokra letölthető, ingyenes *allSchedules* [14] nevű alkalmazás használta föl a szegedi GTFS adatokat. Erről a dolgozat írása közben értesültem, hiszen a program készítője megkért, hogy népszerűsítsem a termékét. Ennek legegyszerűbb módja a GTFS publikációs weboldalon való megjelentetés a többi felhasználás mellett. Ezért fontos a publikálás is, hiszen a GTFS adatokat felhasználók ezen keresztül léphetnek kapcsolatba az állomány előállítójával.

6 Összefoglalás

A fenti példák alapján látható, hogy hazánkban a GTFS alapú tömegközlekedési adatok fölhasználása még gyerekcipőben jár. Azonban már ez a viszonylag kisszámú publikált forrás is arra sarkallt egyeseket, hogy alkalmazásokat fejlesszenek az utasoknak. Munkájuk nem hiábavaló, hiszen az elkészült programok később is bármilyen GTFS formátumú adattal kompatibilisek lesznek, ilyen szempontból átalakítást nem igénylenek. A *Budapesti menetrend* alkalmazást több mint ötezren töltötték le, és a véleményezők hozzászólásait végigfutva a tömegközlekedési adatokra vonatkozó negatív megjegyzést nem találtam – szemben más hasonló fejlesztések véleményezésénél.

Kérdés, hogy a szolgáltatók mennyire ismerik föl az ebben rejlő lehetőségeket. Ennek a TDK-nak nem titkolt célja, hogy a nyilvánosságot kihasználva olyan emberekhez is eljusson a GTFS híre, akik tenni is tudnak a formátum térnyeréséért, minél szélesebb körű alkalmazásáért. Valahogy úgy, ahogy én is értesültem a nyáron a BKK fejlesztéséről, aminek itt az eredménye: az elkészült keretprogram, és a TDK dolgozat.

Bizakodva tekintek a jövőbe, hiszen a szolgáltató cégeknél olyan közlekedésmérnökök dolgoznak, mint én. Ezzel a képzettséggel, a Számítástechnika, a Közlekedési informatika című tárgyakban tanultak fölhasználásával megvalósítható egy ilyen keretprogram. Persze ezek mellett szükség volt jó pár kiegészítő ismeretre is, amelyeket tankönyvekből, és (főképp a GTFS újszerűsége miatt) a webről, internetes fórumokról tudtam megszerezni. Talán ezek alapján körvonalazható az is, hol a közlekedésmérnök helye egy közösségi közlekedési szolgáltató vállalat informatikai szervezetében.

Elnézve a folyamatosan fejlődő OpenTripPlanner alkalmazást, magam is büszke vagyok rá, hogy közreműködhettem abban, hogy Szeged város térképére nagyítva a tömegközlekedési információk megjelenhessenek a megfelelő tartalommal. Ehhez persze nem föltétlenül ez a keretprogram a legjobb megoldás, de egy lehetséges megoldás. A fejlesztés közbeni buktatókról, nehézségekről, illetve ezek kiküszöböléséről hasznos tapasztalatokat szereztem, Remélem, a befektetett munka gyümölcseit hamarosan az utazóközönség is élvezheti, illetve Budapest és Szeged példáját követve hazánk többi részéről is jelennek meg GTFS alapú közösségi közlekedési adatok.

Források

- [1] <http://code.google.com/p/googletransitdatafeed/wiki/FeedSpecHistory>
- [2] <http://transit.google.com>
- [3] <http://terkep.google.hu>
- [4] http://code.google.com/intl/en/transit/spec/transit_feed_specification.html
- [5] <http://www.transxchange.org.uk/>
- [6] <http://www.norc.hu>
- [7] <http://code.google.com/p/googletransitdatafeed/wiki/FeedValidator>
- [8] <http://code.google.com/p/googletransitdatafeed/wiki/ScheduleViewer>
- [9] <http://code.google.com/p/googletransitdatafeed/wiki/PublicFeeds>
- [10] <http://www.gtfs-data-exchange.com/>
- [11] <http://otp.flaktack.net>
- [12] <http://oba.flaktack.net>
- [13] http://www.androidzoom.com/android_applications/transportation/budapesti-menetrend_benut.html
- [14] <http://www.allschedulesapp.com/>

Melléklet

Az adatbázis felépítése

