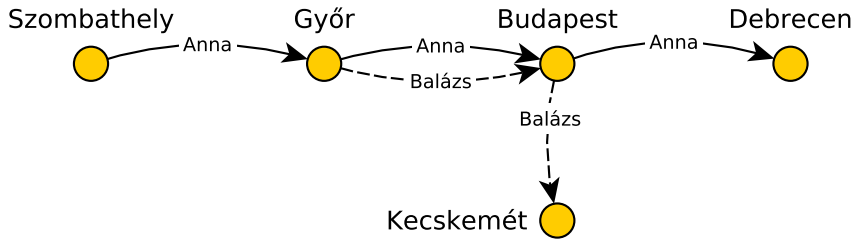
1. gyakorlat – Strukturális modellezés – Megoldások

Közösségi fuvarszolgáltatást tervezünk, ahol bárki meghirdetheti a közeljövőben tervezett autós utazásait; mások pedig a rendszerünkön keresztül értesülhetnek erről, és utasként csatlakozhatnak (akár csak egy rövidebb szakaszon is), ha beszállnak az üzemanyagköltségbe. Az egyes fuvarokat különböző fuvarszakaszokra osztjuk. Nem feltétlenül végig az autó gazdája fog vezetni, ez megbeszélhető, azonban minden fuvarszakasz során jelen kell lennie.
A szolgáltatásunk eddig zárt tesztüzemben futott, a fuvarokat ad-hoc szerveztük, és az adatokat nem rögzítettük szisztematikus módon. Hamarosan szeretnénk nyilvánosan is elindítani a szolgáltatást. A webes felületen az utazásszervezéssel kapcsolatos információkat elérhetővé kell tenni, ezért valamilyen módon nyilván kell ezeket tartanunk.

1. Struktúra modellezése gráffal

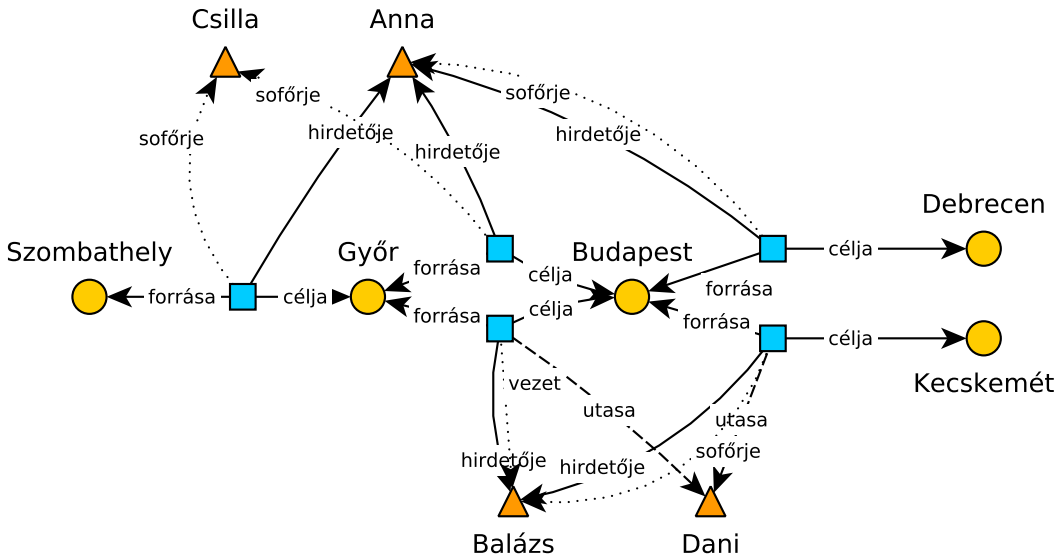
Szeretnénk a rendszer mögöttes adatmodelljét megtervezni. Ehhez az eddigi fuvarok tapasztalatai alapján összeállítottunk néhány tipikus forgatókönyvet.

a) Anna Szombathelyről autóutat tervez Győr és Budapest érintésével Debrecenbe. Balázs Győrből indít fuvart Budapestre, majd onnan tovább Kecskemétre. Alkossunk gráfmodellt a szövegben megadott viszonyok alapján!
Megoldás
Itt egyszerűen (példány)gráfot kell építeni a szöveges leírás alapján. Ezt sokféleképpen lehet, például lehetnek a városok a csomópontok és a közöttük futó irányított élek a fuvarszakaszok (a kocsi gazdájával címkézve). Természetesen máshogy is ábrázolható ugyanez – de erről majd egy későbbi pontban.



b) Dani győri, és nincs kocsija. Milyen gráfelméleti művelet ad választ arra, hogy a felajánlott fuvarokba utasként becsatlakozva mely városokba lehet Győrből eljutni? (Feltehetjük, hogy az autósok az utazás időpontját tekintve rugalmasak.)
Megoldás
A Győrből irányított utakon elérhető csomópontokat keressük. Ha az átszállásokat ki akarjuk zárni (ott ugye figyelembe kellene venni az érkezés és indulás időzítését, amely egyelőre nem része a modellünknek), akkor a csupa egyforma élcímkéjű irányított utakra van szükségünk. Ezeket Győrből induló gráfbejárással határozhatjuk meg, amelyre ismertek algoritmusok.

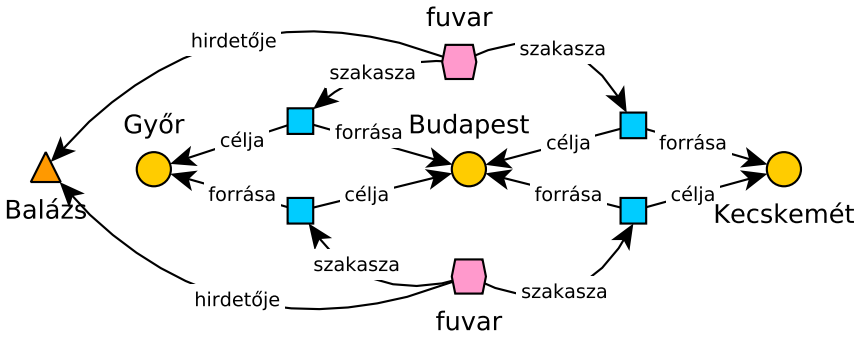
c) Csilla úgy döntött, hogy Anna autójával fog utazni Szombathelytől Budapestig; Dani Győrből Kecskemétre kért fuvart. Mivel Anna előző este sokáig dolgozott, az indulás után aludna, ezért úgy beszélték meg, hogy Budapestig Csilla vezet. Balázs végig maga vezet. Alakítsuk át a gráfot olyan módon, hogy kifejezze ezt a tudást!
Megoldás
A fuvarszakasz kiindulópontja, célja, sofőre és utasai között többes viszony áll fent. Ezen viszonyokat úgy tudjuk gráfként kezelni, ha bevezetünk egy személyt (▲) és egy fuvarszakaszt reprezentáló csompontot (■), amelyhez utána élekkel köthető a többi elem. Így tehát a meglévő éleket csomóponttal és több új éllel helyettesítjük; élet csomóponttá transzformáltunk. Fontos észrevétel, hogy személy és fuvarszakasz között lehet „kínál”, „vezet” és „utazik” jelentésű él is, ezeket legcélszerűbben címkével tudjuk megkülönböztetni. A „vezet” él implicit azt is jelenti, hogy az illető az adott fuvarszakaszon utazik.



d) Balázsnak nem ez az első közösségi fuvarja; korábban Kecskemétről hirdetett utazást Győrbe Budapesten keresztül. Egészítsük ki a gráfot olyan módon, hogy megjelenjen benne ez az információ is!

Megoldás

Most szigorúan csak bővítjük a gráfot. Megjelenik a fuvarszakaszokat összefoglaló fuvar csomópont (⬠), amely a hirdetővel van összekötve. Ez például arra is jó, hogy lássuk, hogy Budapest-Győr-Budapest fuvar például nincs: ez a fuvar fogalom nélkül nem lenne egyértelmű.



e) Milyen művelettel kaphatunk a teljes tudást reprezentáló gráfból egy egyszerűsített nézetet, amelyik csak a hirdetőket, az utazásaikat, és az utazásokat alkotó szakaszokat mutatja? Milyen jellegű lesz az így kapott nézeti gráf?

Megoldás

Csomópontokra és élcímkére szűrjük a részgráfot.

2. Tulajdonságmodellezés

Az eddigi fuvarok során összegyűjtöttünk néhány adatot. Ezeket az alábbi táblázat tartalmazza.

ért. száma	ért. összege	kategória	név	jelszó	rendszám	dohányos	A/C	díjfizetés	jog.sz.
6	24	kocsi			`ABC-123`		nincs		
17	71	személy	Anna	qwe		nem			`KL2048`
16	49	személy	Balázs	pass		igen			`MN4096`
14	45	személy	Csilla	12345		igen		kártya	
1	5	személy	Dani	barát		nem		utalás	
0	0	kocsi			`DEF-456`		van		
7	31	személy	Eszter	2501		nem		kártya	
2	8	személy	Feri	almafa		nem		Bitcoin	

Egy-egy fuvar után 1–5 skálán lehet értékelni, hogy az útitársak és az autó mennyire járultak hozzá a kellemes utazáshoz. A fenti táblázat többek között a begyűjtött értékeléseket is mutatja.

a) Amikor valaki a lehetséges fuvarok közül válogat, az útitársak és az autó értékelése mellett a légkondicionáló megléte, ill. a dohányzás is fontos szempont lehet. A döntéshez azonban nem kell, és nem is szabad a felhasználóknak megismernie a sofőrök jelszavát és jogosítványszámát, valamint hogy a többi útitárs a fuvar megegyezéses díját milyen módszerrel rendezi a cég felé. Milyen (tulajdonságmodellezésnél megismert) művelettel kapható meg az ehhez a nézethez szükséges információ?
Megoldás
Ez egy *vetítés* művelet, amely a három megnevezett jellemzőt eldobja, a többit megtartja.

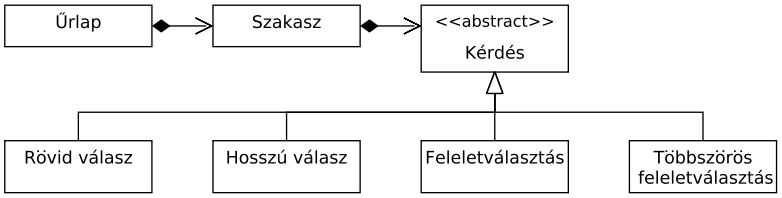
b) A rangsorolás az értékelések pontszámának nem az összege, hanem az átlaga alapján történik. Milyen művelettel bővíthető ki a tulajdonságmodell ezen számítás eredményével?
Megoldás
Vegyük fel egy új „értékelések átlaga” oszlopot (jellemzőt), amely *származtatott tulajdonságként* az alábbi képlettel számítható (ha legalább egy értékelés van már):

$$\textit{értékelések átlaga} = \frac{\textit{értékelések összege}}{\textit{értékelések száma}}$$

c) Eszter és Feri most keres fuvart. Eszter lehetőleg 4 pont feletti értékelésű sofőrt szeretne. Feri csak légkondicionálóval rendelkező kocsik közül hajlandó válogatni. Milyen műveletekkel kaphatóak meg a választási lehetőségeik?
Megoldás
Ezek *szűrés* műveletek lesznek, amelyek csak azokat a sorokat tartják meg, amelyekben a „értékelések átlaga” jellemző értéke egy adott tartományba esik, illetve a „légkondicionáló” jellemző értéke „van”.

3. Űrlap

Az alábbi metamodell alapján készítsünk egy űrlapot, amelynek segítségével az utasok egy-egy utazást követően visszajelzést adhatnak a sofőrről. Nem szeretnék túl sok időt elvenni az utastól, ezért a legtöbb információt eldöntendő, illetve feleletválasztós kérdések formájában gyűjtjük be. Az utasnak lehetősége van arra is, hogy saját szavaival összefoglalja tapasztalatait egy rövid szöveges vélemény keretében.



a) Milyen információra van szükségünk a sofőr azonosításához?
Megoldás
A sofőrt azonosíthatjuk a személyi vagy a jogosítványszáma alapján, de mindkettő adatvédelmi kérdéseket vet fel. Általános esetben az a megoldás, hogy a rendszerünk utazás azonosítót generál, amely amely arra szolgál, hogy az utazás adatokat (indulás, cél, sofőr, utasok stb.) egy (nem publikus) adatbázisban tárolja, így az utasnak elég ezt megadnia.

b) Gyűjtsünk össze pár kérdést, csoportosítsuk őket, majd adjuk meg az elkészült űrlapnak egy modelljét. (Ez már példánymodell lesz a későbbiekben.)
Megoldás
Lehetséges kérdések (a teljesség igénye nélkül):
- Utazás azonosító (Rövid válasz)
- Szolgáltatás értékelése (1-től 5-ig) (Feleletválasztás)
- Minek kapcsán tapasztalt esetlegesen hiányosságokat: pontos indulás és/vagy érkezés, megállások gyakorisága, ülés kényelmessége, autó felszereltsége (Többszörös feleletválasztás)
- Szöveges vélemény, egyéb tapasztalatok (Hosszú válasz)

c) Top-down vagy bottom-up tervezést alkalmaztunk?
Megoldás
Ebben az esetben bottom-up tervezés volt (a kérdésektől a szekciókon át az űrlap megalkotásáig),

míg a metamodellt top-down terveztük (az űrlaptól a szekciókon át eljutottunk a kérdésekig, majd összegyűjtöttük a lehetséges típusokat).

d) Ha az utas az ötös skálán való értékelésnél hármasnál rosszabbat ad a sofőrre, akkor mindenképpen szeretnénk szöveges véleményt. Hogyan tudnánk ezt a modellben megfogalmazni (és melyikben)?
Megoldás
Jólformáltsági kényszerrel jelezhetjük a metamodellben, amely alapján a példánymodell maga már validálható.

4. Típusok modellezése

A szolgáltatáshoz szeretnénk adatbázist tervezni. Ehhez fontos, hogy megkülönböztessük a típusokat a rendszerünkben és keressünk validációs szabályokat.

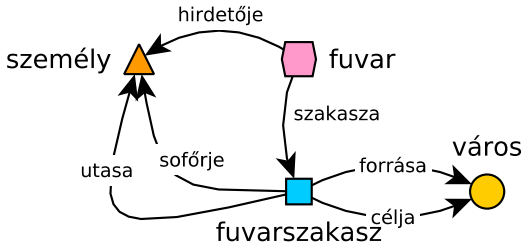
a) Milyen alapvető elem- és kapcsolattípusokat sugallnak a gráfmodellben látható megadott viszonyok? Ábrázoljuk típusgráffal!
Megoldás
Észrevehetjük, hogy „sofőrje” címkéjű él mindig fuvarszakasz jelentésű csomópontból megy egy személyt jelentő csomópontba: lehet, hogy itt rögtön két csomóponttípust és egy éltípust fogtunk! A csomóponttípusok és éltípusok megválasztása elég egyértelmű, de persze az élek elnevezése, irányítása egyedi is lehet; például

$$(\text{fuvarszakasz}) - [\text{sofőrje}] \rightarrow (\text{személy})$$

él helyett lehet

$$(\text{személy}) - [\text{vezet}] \rightarrow (\text{fuvarszakasz})$$

él is, akár összevissza is. Érdemes időt szentelni arra, hogy ezekben az esetekben konzisztensen egyféle nevezéktant használjunk.
Ez alapján egy lehetséges típusgráf:

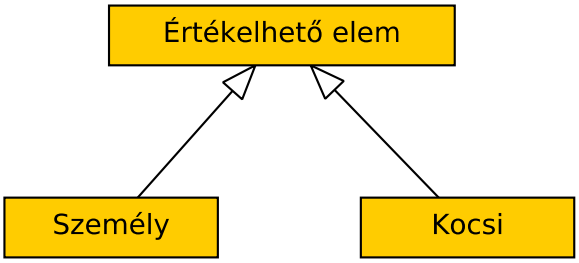


(Megjegyzés: a gráfban nem jelent meg az autó, ezzel a típusgráfban most nem foglalkoztunk.)

b) Milyen típusokba sorolhatóak a táblázatban szereplő elemek a rajtuk értelmezett jellemzőik köre és a kapcsolataik alapján?
Megoldás
Észrevehetjük, hogy a jellemzők egy köre (pl. „rendszám”) csak az értékelhető elemek egy részén, míg más jellemzők (pl. „jogosítványszám”) csak a többi értékelhető dolgon értelmezettek. Hasonló megfigyelésre juthatunk a gráfban tapasztalt kapcsolattípusok alapján is. A két csoportot épp a „kategória” jellemző értéke alapján különíthető el. Vegyük észre, hogy egy értékelhető elemet alapvetően jellemez, hogy személy vagy jármű-e, és ez nem fog később megváltozni. Ez tehát két típusnak tekinthető.

c) Definiáljunk egy típushierarchiát a problémára!
Megoldás

Vegyük észre, hogy az „értékelhető elem” mégis közös általánosítása a kocsiknak és személyeknek, hiszen bizonyos aspektusból hasonló a kezelésük. Tehát a „személy” és a „kocsi” az „értékelhető elem” két altípusának tekinthető.



Kézenfekvőnek tűnhet, hogy megkülönböztessük a sofőr (autóval és jogsival rendelkező) felhasználókat az utasoktól (akik csak betársulnak és fizetnek ezért). Ez azonban nem jó típusbesorolás, mivel az idő folyamán változhat (pl. Feri idővel jogosítványt szerezhet és autót vehet; Balázs legközelebb dönthet úgy, hogy más autójával utazik), és az is lehet, hogy nem az autó gazdája vezet egy adott szakaszon. Általában a *típus* szót (szemben pl. a *fogalom* szóval) csak olyan kategorizálásnál használjuk, amikor egy elem besorolása nem változik az időben, ezért nem javasolt típusrendszert alapozni arra, ki sofőr és ki utas. Azonban egy-egy fuvar(szakasz) esetén vannak olyan *szerepek*, hogy kik ülnek az autóban, ki közülük a kocsi gazdája, és ki vezet. Tanulság: a szerep és a típus nem ugyanaz.

d) (*Kiegészítő feladat*) A típusgráf, a típushierarchia és a jellemzők értelmezési tartománya alapján rajzoljunk metamodellt! Milyen további megkötésekkel (jólformáltsági kényszerekkel) egészíthetjük ki?

Megoldás

Jólformáltsági kényszerre példák:

- Minden fuvarnak legyen legalább egy szakasza.
- A fuvar szakaszai egymáshoz csatlakozzanak, tehát a harmadik szakasz ott végződjön, ahonnan a negyedik indul

Kiegészítő feladat: megvalósítás programmal

a) Készítsünk olyan adatstruktúrát (tetszőleges programozási nyelven), amely egy ilyen fuvarszervező információtartalmának reprezentálására szolgál!

b) Egészítsük ki olyan eljárással (metódussal) a programot, amely képes felsorolni, hogy egy megadott városból hova juthatunk el (átszállás nélkül) a meghirdetett fuvarokkal!

c) Készítsük el az előző eljárás okosabb változatát, amelyik igény szerint elkerüli azokat a fuvarokat, ahol legalább egy szakaszon dohányzó útitársunk lenne!

Megoldás

Kiindulási ötlet: készítsünk egy-egy C struktúrát a Fuvar és a Fuvarszakasz jellemzőivel, valamint hivatkozásokkal egymásra:

```
1  typedef struct {
2    char* felhasznaloi_nev;
3    //...
4    BOOL dohanyzik_e;
5  } Szemely;
6
7  typedef struct {
8    char* kiindulasi_telepules;
9    char* cel_telepules;
10   time_t datum;
11   //...
12   Szemely* utasok_tombje;
13 } Fuvarszakasz;
14
15 typedef struct {
16   Fuvarszakasz* szakaszok;
17   //...
18 } Fuvar;
```