13 ALKALMAZÁSARCHITEKTÚRÁK

TÉMAKÖRÖK

A fejezet célja az alkalmazói szoftverrendszerek bizonyos osztályai architekturális modelljeinek bemutatása. Mire a fejezet végére érünk:

- meg fogjuk ismerni az üzleti rendszerek két alapvető architekturális szervezési módját, a kötegelt és a tranzakciófeldolgozást;
- meg fogjuk érteni az információ- és erőforrás-kezelő rendszerek absztrakt architektúráját;
- meg fogjuk tudni, hogy a parancsvezérelt rendszerek (például szerkesztők) hogyan szervezhetők eseményfeldolgozó rendszerekké;
- meg fogjuk ismerni a nyelvfeldolgozó rendszerek szerkezetét és felépítését.

TARTALOM

Ahogyan arról a 11. fejezetben írtam, a rendszer-architektúrákat több szemszögből vizsgálhatjuk. Eddig a 11. és 12. fejezet rendszer-architektúrákról szóló fejtegetései a felépítésre, valamint a vezérlés, elosztás és rendszerstrukturálás kérdéseire összpontosítottak. Ebben a fejezetben azonban új szemszögből, az alkalmazások szemüvegén keresztül vizsgáljuk az architektúrákat.

Az alkalmazások valamilyen üzleti vagy szervezeti szükséglet kielégítésére szolgálnak. Az üzleti részek sok közös jellemzővel rendelkeznek – felvesznek embereket, számlákat bocsátanak ki, folyószámlájuk van stb. –, ami különösen igaz az egyazon üzleti szektorban tevékenykedő cégekre. Ezért az általános üzleti funkciók mellett minden telefonszolgáltatást nyújtó cégnek kell hívás-összekapcsolást, hálózat-karbantartást, számlakibocsátást stb. végeznie. Következésképp az ilyen cégek által használt alkalmazásokban igen sok közös van.

Általában az azonos típusú rendszerek hasonló architektúrával rendelkeznek, és a rendszerek közötti különbségek csak a részletes funkcionalitásban mutatkoznak meg. Ezt az SAP R/3-hoz (Appelrath és Ritter, 2000) hasonló üzleti

erőforrás-tervező rendszerek (Enterprise Resource Planning, ERP) és az ezekhez fejlesztett vertikális szoftvercsomagok térhódítása is alátámasztja. Ezekben a rendszerekben – amelyeket a 18. fejezetben röviden tárgyalok – egy általános rendszert konfigurálnak és dolgoznak át egy speciális üzleti alkalmazás elkészítésekor. Például egy ellátáslánc-kezelő rendszer különféle ellátókra, javakra és szerződési feltételekre átszabható.

Az alkalmazásarchitektúrák itteni tárgyalásakor bemutatom néhány alkalmazásfajta általános szerkezeti modelljét. Szót ejtek ezen alkalmazásfajták alapvető felépítéséről, és – ahol lehetséges – a magas szintű architektúra „lerombolásával" megmutatom azokat az alrendszereket, amelyek ténylegesen bekerülnek az alkalmazásokba.

Szoftvertervezőként ezeket az általános alkalmazásarchitektúrákat a következő módokon tudjuk felhasználni:

1. *Az architekturális tervezés folyamatának kiindulópontja.* Ha ismeretlen számunkra az adott alkalmazásfajta, kezdeti terveinket alapozhatjuk az általános architektúrákra. Természetesen ezt konkrét rendszerekre majd specializálni kell, de jó kiindulási alapot nyújtanak a terv elkészítéséhez.
2. *Tervezési gyorslista.* Ha már elkészült a rendszer architektúrájának a terve, ezt az általános architektúrával összevetve leellenőrizhetjük, hogy nem hagytunk-e ki valamilyen fontos komponenst.
3. *A fejlesztői csapat munkájának szervezése.* Az alkalmazásarchitektúrák a rendszer-architektúrák stabil szerkezeti jellemzőit azonosítják, és sok esetben ezek párhuzamosan is fejleszthetők. Az architektúra különböző alrendszereinek megvalósítását rábízhatjuk a csoport tagjaira.
4. *Komponensek újrafelhasználásának felmérési módja.* Ha vannak olyan komponenseink, amit esetleg újrafelhasználhatóvá kellene tennünk, akkor összevethetjük ezt az általános szerkezetekkel, hogy kiderüljön, valószínű-e, hogy azokat újra fel tudnák használni az alkalmazásunkban.
5. *Az alkalmazásfajták fogalomszótára.* Ha egy adott alkalmazásról beszélünk, vagy több, ugyanolyan fajta alkalmazás összehasonlítását végezzük, akkor az általános architektúra fogalmainak segítségével fejezhetjük ki magunkat.

Sokfajta alkalmazás létezik, és ezek a felszínen nagyon különbözőnek is tűnhetnek. Azonban ha az alkalmazások architekturális felépítését vizsgáljuk, azt állapíthatjuk meg, hogy sok ilyen felületesen egymásra nem hasonlító alkalmazás mégis közös elvek alapján építkezik. Ezt négy alkalmazásfajta bemutatásával illusztrálom:

1. *Adatfeldolgozó alkalmazások.* Ebbe a csoportba az adatvezérelt alkalmazások tartoznak. Az adatokat kötegekben dolgozzák fel anélkül, hogy közben explicit felhasználói közbeavatkozás történne. Az alkalmazás által végrehajtandó speciális tevékenységek a feldolgozott adatoktól függenek. A kötegelt feldolgozórendszereket általában olyan üzleti alkalmazásokban alkalmazzunk, amelyekben hasonló műveleteket kell nagy mennyiségű adaton elvégezni.

Sok adminisztratív funkció tartozik ide, például fizetési jegyzék készítése, számlázás, hirdetés.

2. *Tranzakciófeldolgozó rendszerek.* Ezek olyan adatbázis-központú alkalmazások, ahol a felhasználók kéréseit feldolgozva információkat frissítünk egy adatbázisban. A leggyakoribb fajtáját az interaktív üzleti rendszerek jelentik. Ezeket úgy szervezik, hogy a felhasználói tevékenységek ne zavarják egymást, és az adatbázis integritása ne sérüljön. Az interaktív banki rendszerek, e-kereskedelmi rendszerek, információs rendszerek és helyfoglaló rendszerek tartoznak ide.
3. *Eseményfeldolgozó rendszerek.* Ez az alkalmazásoknak egy nagyon nagy csoportját takarja, ahol a rendszer tevékenységei a rendszerkörnyezet eseményeinek az értelmezésétől függnek. Ilyen esemény lehet egy felhasználó által kiadott utasítás vagy a rendszer által figyelt változó értékének a megváltozása. Sok PC-re írt alkalmazás, beleértve a játékokat és az olyan szerkesztőrendszereket is, mint amilyenek a szövegszerkesztők, táblázatkezelők, képszerkesztők és prezentációkészítők, ebbe a kategóriába tartozik. Idetartoznak a 15. fejezetben tárgyalt valós idejű rendszerek is.
4. *Nyelvfeldolgozó rendszerek.* Ezekben a rendszerekben a felhasználó szándékát egy formális nyelven (például Javában) fejezi ki. A nyelvfeldolgozó rendszer ezt feldolgozva egy belső alakra hozza, majd ezt a belső reprezentációt értelmezi. A leginkább ismert nyelvfeldolgozó rendszerek a fordítóprogramok, amelyek a magas szintű programozási nyelven leírt programokat gépi kóddá alakítják. A nyelvfeldolgozó rendszereket azonban az adatbázisok parancsnyelvének és a strukturált adatleírásra gyakran használt jelölőnyelveknek (például XML) az értelmezésére is használják.

Azért választottam a rendszereknek ezen csoportjait, mert a manapság használt rendszerek nagy többsége beletartozik valamelyik fenti kategóriába. Az üzleti rendszerek jellemzően vagy adat-, vagy tranzakciófeldolgozó rendszerek, a legtöbb, személyi számítógépre készült szoftver pedig egy eseményfeldolgozó architektúra alapján épül fel. A valós idejű rendszerek szintén eseményfeldolgozók. Ez utóbbiakkal részletesebben a 15. fejezetben foglalkozom. A szoftverfejlesztés teljes egészében a nyelvfeldolgozó rendszerek közé sorolható.

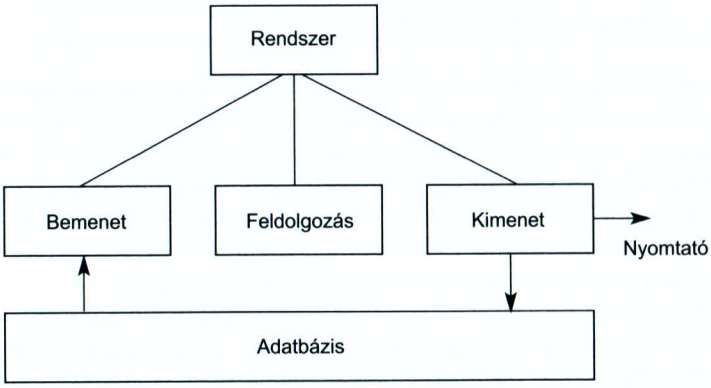
A kötegelt és tranzakciófeldolgozást végző rendszerek egyaránt adatbázis-központúak. Mivel az adatoknak központi szerepük van, gyakran előfordul, hogy különbözőféle alkalmazások ugyanazt az adatbázist használják. Például egy banki kimutatásokat nyomtató üzleti adatfeldolgozó rendszer ugyanazt az ügyfélszámla adatbázist használja, mint az a tranzakciófeldolgozó rendszer, amely webalapú hozzáférést biztosít a számlainformációkhoz.

Természetesen, mint ahogyan azt a 11. fejezetben is írtam, a bonyolult alkalmazások ritkán követnek egyetlen, egyszerű architekturális modellt, ehelyett hibrid architektúrával dolgoznak, amikor is az alkalmazás különböző részei különféleképpen vannak strukturálva. Az ilyen rendszerek tervezésekor ezért figyelembe kell vennünk az önálló alrendszerek architektúráit, és meg kell vizsgálnunk, hogy miként vannak ezek a teljes rendszer-architektúrában integrálva.

13.1. ADATFELDOLGOZÓ RENDSZEREK

Az adatfeldolgozó rendszereken alapuló rendszereket az olyan üzleti tevékenységek támogatására alkalmazzák, mint amilyen a fizetések kifizetése, a számlák kiszámítása és nyomtatása, a folyószámlák karbantartása vagy akár a biztosítások megújítása. Ahogyan az a nevükből is következik, ezek a rendszerek általában az adatokat helyezik a középpontba. Az ezen adatokat tartalmazó adatbázisok általában nagyságrendekkel nagyobbak magánál az adatfeldolgozó rendszernél. Az adatfeldolgozó rendszerek olyan kötegelt feldolgozást végeznek, ahol az adatok be- és kimenetét állományok vagy adatbázisok segítségével valósították meg ahelyett, hogy egy felhasználói terminál szolgálna be- és kimenetként. Ezek a rendszerek a bejövő rekordokból adatokat válogatnak le, azután az értékektől függően végrehajtanak valamilyen tevékenységet, majd a számítás eredményét visszaírhatják az adatbázisba, és nyomtatás céljából megformázhatják a bemenő és a kiszámított kimenő értékeket.

A kötegelt feldolgozó rendszerek architektúrája három fő komponensből áll, mint ahogy az a 13.1. ábrán is látható. A bemeneti komponens összegyűjti az akár több különböző adatforrásból érkező bemeneteket, a feldolgozó komponens ezen bemenetek alapján számításokat végez, a kimeneti komponens pedig generálja az adatbázisba visszaírandó és kinyomtatandó kimeneteket. Például egy telefonszámla-készítő rendszer veszi az ügyfelek adatait, valamint a telefonközpontból a beszélgetésekre vonatkozó adatokat (bemenetek), majd kiszámítja az egyes ügyfelek költségét (feldolgozás), végül kinyomtatja a számlákat (kimenetek).



13.1. ábra. Egy adatfeldolgozó rendszer bemenet-feldolgozás-kimenet modellje

A bemeneti, feldolgozó és kimeneti komponensek a bemenet-feldolgozás-kimenet szerkezet alapján tovább bonthatók. Például:

1. A bemeneti komponens adatokat (bemenet) olvashat be egy állományból vagy adatbázisból, ellenőrizheti az adatok érvényességét és kijavíthat bizonyos hibákat (feldolgozás), majd az érvényes adatokat (kimenet) egy sorba helyezheti.

2. A feldolgozó komponens kivehet egy ilyen ellenőrzött adatot a sorból (bemenet), ez alapján bizonyos számításokat végezhet, és létrehozhat egy új rekordot, amely a számítás eredményét tartalmazza (feldolgozás), majd ezt az új rekordot nyomtatás céljából egy sorba helyezi (kimenet). A feldolgozás történhet az adatbázisban is, illetve külön program is végezheti.
3. A kimeneti komponens a sorból kiolvashatja a rekordokat (bemenet), megformázhatja azt a kimeneti igényeknek megfelelően (feldolgozás), majd elküldi egy nyomtatónak vagy visszaírja az új rekordot az adatbázisba (kimenet).

Az adatfeldolgozó rendszerek természete szerint a rekordok és a tranzakciók sorban kerülnek feldolgozásra, és nincs szükség az állapotok tranzakciók közötti megtartására, vagyis ezek a rendszerek természetüknél fogva nem objektum-, hanem függvényorientáltak. A függvények olyan komponensek, amelyek két hívás között nem tartják meg belső állapotuk információit. A 8. fejezetben bemutatott adatfolyam-diagramok alkalmasak az üzleti adatfeldolgozó rendszerek architektúrájának leírására.

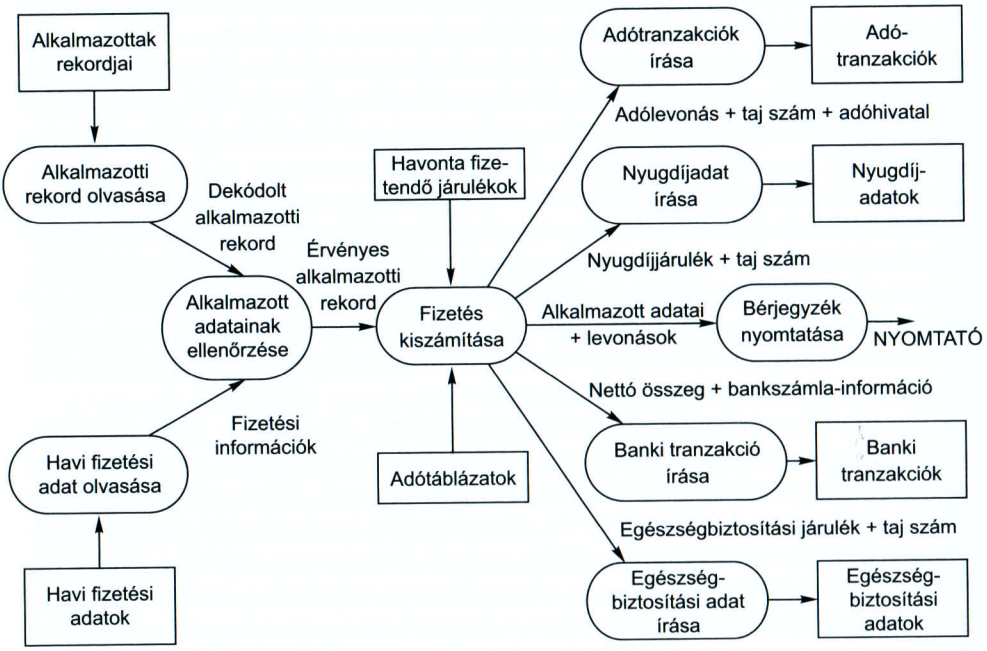
Az adatfolyam-diagramok a függvényorientált rendszerek egyik ábrázolási lehetőségét nyújtják, ahol minden lekerekített sarkú téglalap egy valamilyen adattranszformációt megvalósító függvényt reprezentál, míg a nyilak a függvény által feldolgozott adatokat jelzik. Az állományokat és adattárakat téglalapok jelzik. Az adatfolyam-diagramok előnye, hogy a feldolgozást elejétől a végéig bemutatják, azaz láthatjuk, hogy az adatok hogyan terjednek a rendszerben, és a rajtuk végrehajtott összes függvényt is leolvashatjuk. Az alapvető adatfolyam szerkezet egy bemeneti, egy feldolgozó és egy kimeneti függvényből áll.

A 13.2. ábrán az látható, hogy hogyan használhatjuk az adatfolyam-diagramokat az adatfeldolgozó rendszerek architektúrájának részletes megjelenítésére. Az ábrán egy fizetéskifizető rendszer terve látható. A rendszer beolvassa a szervezet alkalmazottainak információit, kiszámítja a havi fizetést és a levonásokat, majd elvégzi a kifizetést. Látható, hogy a rendszer az alapvető bemenet-feldolgozás-kimenet szerkezetet követi:

1. A diagram bal oldalán látható Alkalmazotti rekordok olvasása, Havi fizetési adatok olvasása és Alkalmazott adatainak ellenőrzése függvények betöltik az alkalmazottak adatait, és ellenőrzik azokat.
2. A Fizetés kiszámítása függvény kiszámolja az alkalmazottak bruttó fizetését, majd ebből meghatározza a levonásokat, végül ezekből kiszámítja a havi nettó fizetést.
3. A kimeneti függvények különböző állományokba írják a kiszámolt levonásokat és a fizetést. Ezeket majd más programok fogják feldolgozni, ha már minden alkalmazott fizetésének kiszámítása elkészült. A nettó összeget és a levonásokat is tartalmazó bérjegyzék kinyomtatásra kerül.

Az adatfeldolgozó programok architekturális modellje viszonylag egyszerű. Ilyen rendszerek esetén azonban az alkalmazás bonyolultsága gyakran a feldolgozandó adatokban tükröződik. A rendszer-architektúra tervezésekor ezért

nemcsak a program, hanem az adatok architektúráját is meg kell tervezni (Brackett, 1994). Az adatarchitektúrák tervezésének kérdésköre azonban meghaladja e könyv kereteit.



13.2. ábra. Egy bérszámfejtő rendszer adatfolyam-diagramja

13.2. TRANZAKCIÓFELDOLGOZÓ RENDSZEREK

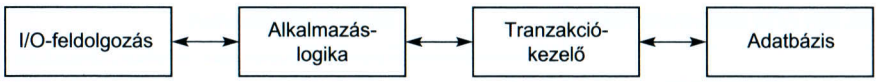
A tranzakciókezelő rendszerek arra szolgálnak, hogy feldolgozzák a felhasználók kéréseit az információ adatbázisból történő kinyerésére és az adatbázis frissítésére vonatkozóan (Lewis *és mások*, 2003). Technikailag egy adatbázis-tranzakció nem más, mint műveleteknek egyetlen egységként (atomi egységként) tekintett sorozata. Mielőtt a változtatások véglegesítődnének, a tranzakció minden műveletének be kell fejeződnie. Ez azt jelenti, hogy a tranzakción belüli esetlegesen hibás műveletek nem veszélyeztetik az adatbázis konzisztenciáját.

Tranzakcióra példa, amikor az ügyfél kérelmezi egy banki ATM-nél a bankszámlájáról történő pénzlevételt. Ehhez le kell kérni az ügyfél bankszámlájának adatait, ellenőrizni kell az egyenleget, le kell vonni belőle a levenni kívánt összeget, és utasítani kell az ATM-et, hogy adja ki a pénzt. Míg a lépések mindegyike végbe nem megy, a tranzakciónak nincs vége, és az ügyfél számláján nem történik semmiféle változás.

A felhasználó szempontjából egy tranzakció egy valamilyen célt kielégítő koherens műveletsorozat, mint például „keresd meg a Londonból Párizsba tartó

járatok indulási idejét". Még ha a felhasználói tranzakció nem is igényli az adatbázis módosítását, akkor is érdemes ezeket a műveleteket technikailag egy adatbázis-tranzakcióban elhelyezni.

A tranzakciófeldolgozó rendszerek általában olyan interaktív rendszerek, ahol a felhasználók aszinkron módon szolgáltatásokat kérnek. A 13.3. ábra az ilyen alkalmazások magas szintű architekturális szerkezetét ábrázolja. Először a felhasználó egy I/O-feldolgozást végző komponensen keresztül küld egy kérést. A kérés feldolgozása egy alkalmazásspecifikus logika alapján megy végbe. Létrejön egy tranzakció, ami egy tranzakciókezelőhöz kerül. Ez általában egy adatbázis-kezelő rendszerbe van beépítve. Miután a tranzakciókezelő biztosítja, hogy a tranzakció rendben végrehajtódott, jelzi az alkalmazásnak, hogy a feldolgozás befejeződött.



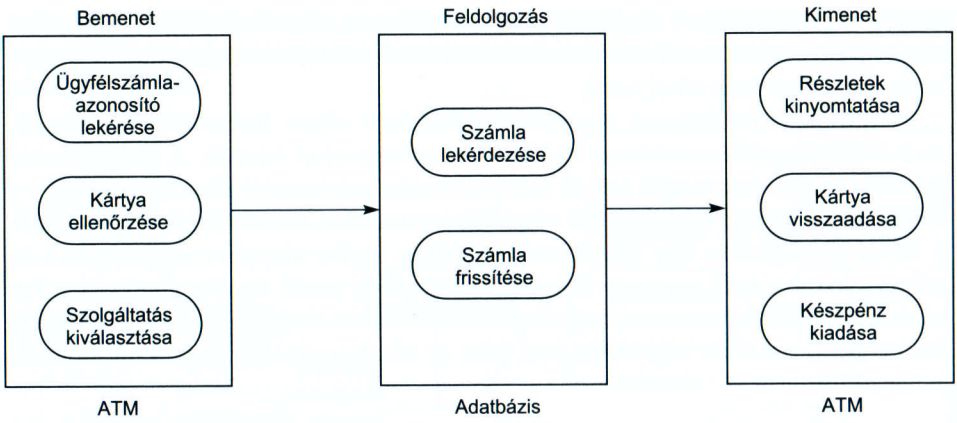
13.3. ábra. Tranzakciófeldolgozó alkalmazás szerkezete

Az adatfeldolgozó rendszereknél megismert bemenet-feldolgozás-kimenet szerkezet a legtöbb tranzakciófeldolgozó rendszerre is ráillik. Ezek némelyike kötegelt feldolgozórendszerek interaktív verziója. Például régebben a bankok az ügyfelek tranzakcióit (például utalásokat) offline gyűjtötték, és az éjszaka folyamán egy kötegelt feldolgozás során végezték a bankszámlák módosítását. Ezt a megközelítést mára már teljesen lecserélték az interaktív, tranzakciófeldolgozó rendszerek, ahol a bankszámlák módosítása valós időben zajlik.

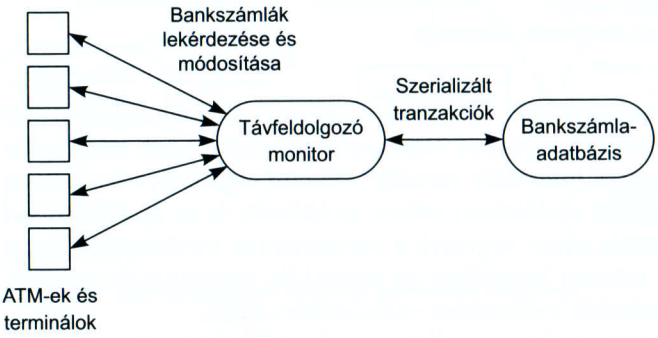
Egy tranzakciófeldolgozó rendszerre példa egy olyan banki rendszer, amely lehetővé teszi az ügyfeleknek, hogy egy ATM segítségével lekérdezzék bankszámlájuk egyenlegét, és készpénzt vegyenek fel. A rendszer két együttműködő alrendszerből áll, az ATM szoftveréből és a bank adatbázisszerverében működő bankszámla-feldolgozó szoftverből. A bemeneti és kimeneti alrendszereket az ATM szoftverében kell implementálni, míg a feldolgozó alrendszert a bank adatbázisszerverében. A rendszer architektúrája a 13.4. ábrán látható. Az alap bemenet-feldolgozás-kimenet diagramot egy kicsit kibővítettem, hogy látni lehessen, hogy milyen komponensekből állhat össze a bemeneti, a feldolgozó és a kimeneti tevékenység. Szándékosan nem ábrázoltam, hogy ezek a komponensek hogyan kommunikálnak, mert a műveletek sorrendje gépről gépre eltérhet.

A bank ügyfélszámlarendszeréhez hasonló rendszerek esetén többféle módon is kapcsolatot tudunk teremteni a rendszerrel. Sokan az ATM-eken keresztül lépnek majd kapcsolatba vele, mások a banki terminálon keresztül, és lesznek, akik webböngészőn keresztül férnek majd hozzá bankszámlájuk adataihoz.

A különböző terminálok kommunikációs protokolljainak kezelését leegyszerűsítendő, a nagyméretű tranzakciófeldolgozó rendszerek általában tartalmaznak egy köztes szoftvert, amely a különféle terminálfajtákkal kommunikál, szerializálja az onnan érkező adatokat, és elküldi azokat feldolgozásra. Ezt a köztes szoftvert, amelyről a 12. fejezetben már röviden szóltam, távfeldolgozó monitor-



13.4. ábra. Egy ATM szoftverarchitektúrája



13.5. ábra. Tranzakciókezelés köztes szoftvere

nak vagy tranzakciókezelő rendszernek nevezhetjük. Az IBM CICS (Horswill és Miller, 2000) rendszere egy igen széles körben használt ilyen rendszer.

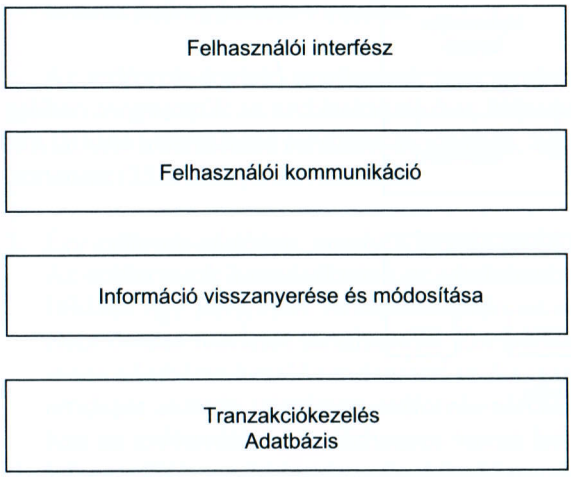
A 13.5. ábrán az ATM-ből vagy a banki terminálról érkező tranzakciókat kezelő ügyfélszámlarendszer architektúrájának egy másik nézete látható. A távfeldolgozó monitor kezeli a bemenetet és szerializálja a tranzakciókat, amelyeket adatbázis-lekérdezésekké alakít. A lekérdezésfeldolgozó az adatbázis-kezelő rendszerben helyezkedik el. Az eredmények a távfeldolgozó monitorhoz kerülnek vissza, amely nyomon követi, hogy melyik terminál küldte a kérést. Ez a rendszer olyan formátumra hozza az adatokat, amit a terminál szoftvere is megért, majd visszaadja neki a tranzakció eredményeit.

13.2.1. Információ- és erőforrás-kezelő rendszerek

Minden olyan rendszert, amely egy megosztott adatbázissal kapcsolatba lép, tranzakcióalapú információs rendszernek tekintünk. Információs rendszerként ellenőrzött hozzáférést biztosít nagy mennyiségű adathoz, mint például könyv-

tári katalógusok, repülőgépjáratok menetrendje, kórházi betegek rekordjai. A www fejlődése nyomán rengeteg speciális szervezeti információs rendszerből vált teljes körűen elérhető, általános célú rendszer.

A 13.6. ábra egy információs rendszer nagyon általános architektúráját mutatja be. A rendszert a rétegzett vagy absztrakt gép megközelítéssel (11.2.3. alfejezet) modelleztük, ahol a legfelső réteg a felhasználói felületet, a legalsó pedig az adatbázist jelenti. A felhasználói kommunikáció réteg a felhasználói felületről érkező és oda menő bemenetekkel és kimenetekkel foglalkozik, az információlekérő réteg pedig az adatbázis-elérés alkalmazásfüggő logikáját tartalmazza. Ahogy majd azt a későbbiekben látni fogjuk, a modell rétegei közvetlenül egy internetalapú rendszer szervereire képezhetők le.



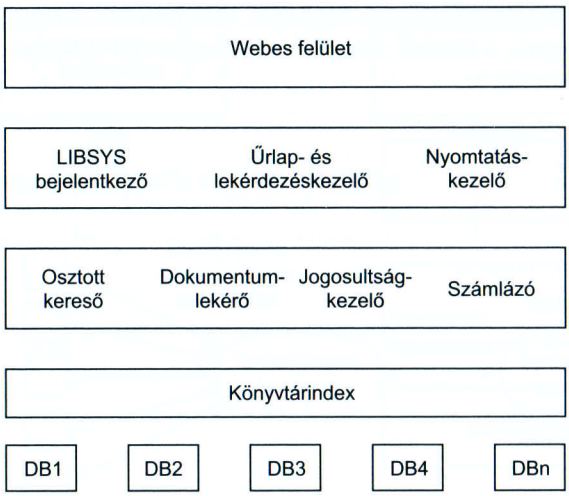
13.6. ábra. Egy információs rendszer rétegzett modellje

Ennek a rétegzett modellnek egy példányaként tekintsük a LIBSYS-rendszer 13.7. ábrán látható architektúráját. Emlékeztetőül megjegyzem, hogy ez a rendszer távoli könyvtárak dokumentumaihoz történő hozzáférést biztosít, amelyeket aztán kinyomtatás céljából le is lehet tölteni. Minden rétegbe beleírtam, hogy mely komponensek támogatják a kommunikációt, illetve az információelérést. Megjegyzendő, hogy az adatbázis osztott. A felhasználók mindig a dokumentumot biztosító könyvtár adatbázisához kapcsolódnak.

A felhasználói kommunikáció rétegének a 13.7. ábrán látható módon három komponense van:

1. A *LIBSYS bejelentkező komponens* azonosítja és hitelesíti a felhasználót. Minden olyan információs rendszernek, amely meg szeretné szabni, hogy kik vehessék igénybe, a felhasználói kommunikáció rétegben tartalmaznia kell felhasználói hitelesítést. Ez akár személyes is lehet, vagy e-kereskedelmi rendszerekben hitelkártyaadatok megadásával is történhet.

2. Az *űrlap- és lekérdezéskezelő komponens* a felhasználó számára megjelenített űrlapokat kezeli, és olyan lekérdezési lehetőségeket biztosít, amelyek lehetővé teszik, hogy a felhasználó információt kérjen a rendszertől. Ilyen komponenst minden információs rendszernek tartalmaznia kell.
3. A *nyomtatáskezelő komponens* LIBSYS-specifikus. A szerzői jogok miatt csak korlátozottan nyomtatható dokumentumok nyomtatását vezérli. Például bizonyos dokumentumokat csak egyszer szabad kinyomtatni.



13.7. ábra. A LIBSYS-rendszer architektúrája

A LIBSYS információlekérő és -módosító rétege a rendszer funkcionalitását megvalósító alkalmazásspecifikus komponenseket tartalmazza. Ezek az alábbiak:

1. *Osztott kereső.* A felhasználó által kezdeményezett keresést a rendszernél bejegyzett összes könyvtárnál elvégzi. A bejegyzések a könyvtárindexben vannak karbantartva.
2. *Dokumentumlekérő.* A felhasználó által igényelt dokumentumot vagy dokumentumokat áttölti a LIBSYS-rendszert futtató gépre.
3. *Jogosultságkezelő.* Ez a komponens kezeli a digitális és szerzői jogokat. Nyomon követi, hogy ki kérte le a dokumentumokat, és például azt is biztosítja, hogy ugyanaz a személy ne kérhesse le ugyanazt a dokumentumot többször.
4. *Számlázó.* Ez a komponens minden kérést naplóz, és kezeli a esetlegesen díjköteles dokumentumok kapcsán a fizetést. Ezen túlmenően jelentéseket készít a rendszerről.

Ugyanazt a négyrétegű általános szerkezetet látjuk, mint egy másik fajta információs rendszer, az erőforrások foglalását támogató rendszer esetében. Az erőforrás-foglaló rendszerek egy bizonyos erőforrásból kezelnek egy fix mennyiséget, például koncert- vagy focimeccsjegyek. Ezeket a kérelmezőkhöz rendeli

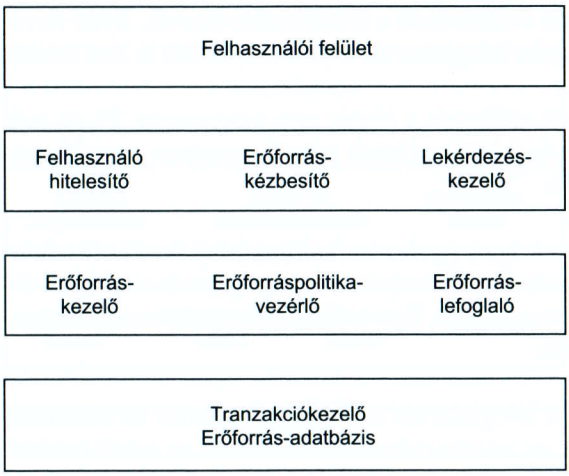
hozzá. A jegyfoglaló rendszerek nyilvánvaló példái egy ilyen erőforrás-foglaló rendszernek, de sok, látszólag nem hasonló rendszer is ilyen, például:

1. *Órarendi rendszerek,* amelyekben az órarend bejegyzéseihez órákat rendelünk. A lefoglalt erőforrás itt egy időintervallum, és általában rengeteg megszorítás tartozik minden erőforrásigényhez.
2. *Könyvtári rendszerek,* amelyek a könyvek kikölcsönzését és visszavitelét kezelik. Ebben az esetben a lefoglalt erőforrások a kölcsönzött elemek. Ilyen rendszerek esetében nem elég csupán lefoglalni az erőforrást, de fel is kell szabadítani.
3. *Légi irányító rendszerek,* ahol az erőforrás a légtér egy szegmense. Ez az erőforrás dinamikus lefoglalását és felszabadítását jelenti, azonban az erőforrás itt nem fizikai, hanem virtuális.

Az erőforrás-foglaló rendszerek igen gyakori alkalmazásfajták. Ha részletesebben megnézzük az architektúrájukat, láthatjuk, hogyan igazodnak a 13.6. ábrán látható információs rendszer modellhez. Egy erőforrás-kezelő rendszer komponensei (13.8. ábra) a következők:

1. *Egy erőforrás-adatbázis,* amely a lefoglalandó erőforrások adatait tartalmazza. Az erőforrások hozzáadhatók az adatbázishoz és törölhetők az adatbázisból. Például egy könyvtári rendszer esetén az adatbázis minden kölcsönözhető elem összes részletét tartalmazza. Ezt tranzakciófeldolgozó rendszert tartalmazó adatbázis-kezelő rendszerrel szokás megvalósítani. Az adatbázis-kezelő rendszer szintén tartalmaz erőforrás-zárolási lehetőségeket, hogy ugyanahhoz az erőforráshoz ne férhessen hozzá két párhuzamosan kérelmet indító felhasználó.
2. *Egy szabályhalmaz,* amely az erőforrások lefoglalásának szabályait írják le. Például egy könyvtári rendszer a beiratkozott felhasználókra korlátozza az erőforrások lefoglalásának lehetőségét, de korlátozhatja a kölcsönzés maximális időtartamát, a maximálisan kölcsönözhető elemek számát stb.
3. *Egy erőforrás-kezelő komponens,* amely az erőforrást biztosító számára teszi lehetővé azok hozzáadását, szerkesztését, törlését.
4. *Egy erőforrás-lefoglaló komponens,* amely az erőforrások kiosztásakor frissíti az erőforrás-adatbázist, és összekapcsolja az erőforrást és a kérelmező adatait.
5. *Egy felhasználóhitelesítési modul,* amely lehetővé teszi, hogy a rendszer ellenőrizze, hogy a felhasználó jogosult-e az erőforrás igénybevételére. A könyvtári rendszerben ez egy gép által leolvasható kártya lehet, a jegyfoglaló rendszer esetén pedig hitelkártya-információk, amely alapján ellenőrizhető, hogy a vásárló ki tudja-e fizetni az erőforrás árát.
6. *Egy lekérdezéskezelő modul,* amelynek segítségével a felhasználók felfedezhetik, hogy milyen erőforrások állnak rendelkezésre. A könyvtári rendszerben ez bizonyos elemekre vonatkozó kérdéseket jelenthet, míg a jegyfoglaló rendszerben grafikusan megjelenhet, hogy hová szólnak még lefoglalható jegyek.

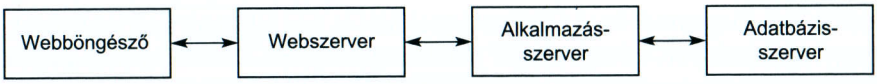
7. *Egy erőforrás-kézbesítő komponens,* amely előkészíti az erőforrást a kérelmezőhöz történő eljuttatásra. A jegyfoglaló rendszerben ez például lehet egy megerősítést kérő e-mail elküldése.
8. *Egy felhasználói felület komponens* (gyakran egy webböngésző), amely a rendszeren kívül áll, és lehetővé teszi, hogy az erőforrás kérelmezője kérdéseket tegyen fel, és kérelmezze bizonyos erőforrások lefoglalását.



13.8. ábra. Egy erőforrás-foglaló rendszer rétegzett architektúrája

Ez a rétegzett architektúra sokféleképpen megvalósítható, például úgy, hogy minden réteg egy külön szerveren futó nagyméretű komponens lesz. Minden réteg definiálja saját külső interfészeit, és minden kommunikáció ezeken keresztül zajlik. Ennek alternatívájaként, ha a teljes információs rendszer egyetlen számítógépen fut, akkor általában egyszerű programokként megvalósított köztes termékek kommunikálnak az adatbázissal, annak API-ján keresztül. Egy harmadik alternatívaként finom szemcsézettségű komponenseket hozhatunk létre különálló webszolgáltatásokként (12. fejezet), amelyeket a felhasználó kérelme esetén dinamikusan összeállítunk.

Az információs és erőforrás-kezelő rendszerek implementációjaként általában az internetprotokollokat alkalmazzuk, a felhasználói felület pedig webböngészőn keresztül jelenik meg. Egy ilyen rendszer esetén a szerverek kialakítása a 13.6. ábrán látható négyrétegű, általános modellt tükrözi. Ezeket a rendszereket általában többrétegű kliens–szerver architektúrák segítségével valósítják meg, erről a 12. fejezet ír. A rendszer felépítése a 13.9. ábrán látható. A felhasználói kommunikációért a webböngésző felel, az alkalmazásszerver feladata, hogy implementálja az alkalmazáslogikát, valamint az információlekérési és információtárolásra vonatkozó kéréseket, az adatbázisszerver pedig az információk adatbázisból és adatbázisba való mozgatásáért felel. Több szerver beállításával növelhetjük a keesztmetszetet, percenkénti több száz tranzakció kiszolgálására is lehetőségünk van.



13.9. ábra. Egy többrétegű internetes tranzakciókezelő rendszer

Az e-kereskedelmi rendszerek olyan internetalapú erőforrás-kezelő rendszerek, amelyek bizonyos javak vagy szolgáltatások elektronikus úton történő rendelhetőségének céljából jöttek létre. Manapság már számos ilyen rendszer létezik az autókölcsönzéstől mint szolgáltatástól egészen az olyan kézzelfogható javak megvásárlásáig, mint amilyenek a könyvek vagy a fűszerek. Az e-kereskedelmi rendszerekben az alkalmazásspecifikus réteg egy pluszfunkcionalitást is tartalmaz, ez pedig a „bevásárlókocsi", amelybe a felhasználók különböző tranzakciókban belepakolhatnak dolgokat, de fizetni már egyetlen tranzakcióban fognak.

13.3. ESEMÉNYFELDOLGOZÓ RENDSZEREK

Az eseményfeldolgozó rendszerek a rendszer környezetében vagy a felhasználói felületen bekövetkező eseményekre reagálnak. Ahogyan a 11. fejezetben említettem, az eseményfeldolgozó rendszerek fő jellemzője, hogy az események bekövetkezésének ideje megjósolhatatlan, de a rendszernek kezdenie kell tudni valamit az adott eseménnyel, ha már bekövetkezett, és lehetőleg akkor, amikor bekövetkezett.

Ilyen eseményalapú rendszereket rendszeresen használunk, például saját számítógépünk igénybevétele esetén. A szövegszerkesztők, prezentációkészítők és a játékok, mind-mind a felhasználói felületről érkezett eseményeket dolgozzák fel. A felhasználói felület események implicit utasításokat adnak a rendszernek, amely valamilyen tevékenységet végrehajtva engedelmeskedik a parancsnak. Például egy szövegszerkesztő esetén, ha duplán kattintunk egy szóra, az azt jelenti, hogy „jelöld ki a szót".

A valós idejű rendszerek, amelyek külső ingerekre „valós idejű" választ adnak, szintén eseményfeldolgozó rendszerek. Azonban a valós idejű rendszerek esetén az események általában nem felhasználói felület események, hanem a rendszer érzékelőivel és működtetőivel kapcsolatban álló események. A megjósolhatatlan eseményekre adandó valós idejű válasz szükséglete miatt ezek a rendszerek általában együttműködő folyamatokból épülnek fel. A valós idejű rendszerek általános architektúrájáról a 15. fejezet szól.

Ebben az alfejezetben a szerkesztőrendszerek általános architektúrájáról lesz szó. A szerkesztőrendszerek olyan programok, amelyek egy PC-n vagy munkaállomáson futnak, és lehetővé teszik, hogy a felhasználók különféle dokumentumokat szerkesszenek, Ezek lehetnek például szöveges dokumentumok, diagramok vagy akár képek is. Néhány szerkesztő egyetlen dokumentumfajta szerkesztését végzi csak, míg mások – ideértve a legtöbb szövegszerkesztőt – különféle dokumentumok szerkesztését is megengedik, például a szöveg mellé elhe-

lyezhetünk egy diagramot is. A táblázatkezelőket is tekinthetjük szerkesztőnek, amely egy munkalapon elhelyezkedő cellákat szerkeszt. A táblázatkezelőknek ezenfelül természetesen vannak további funkciói, például különböző számításokat végezhet.

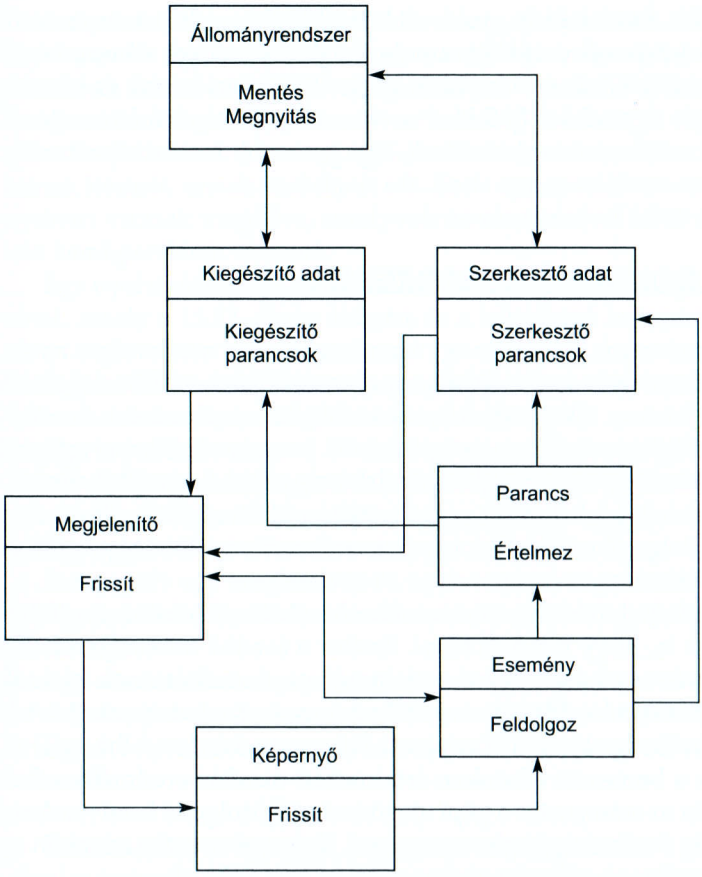
A szerkesztőrendszereknek több olyan jellemzőjük is van, amelyek megkülönböztetik azokat egyéb rendszerektől, és befolyásolják architekturális tervezésüket:

1. Általában egyfelhasználós rendszerek. Éppen ezért nem kell törődniük a konkurens hozzáféréssel, és egyszerűbb adatkezelő résszel rendelkeznek, mint a tranzakcióalapú rendszerek. Még ha vannak is osztott adatok, akkor sem jellemző rájuk a tranzakciók használata, mert azok hosszú időt vesznek igénybe, így az adatintegritást inkább más módon biztosítják.
2. Gyors visszajelzéssel kell szolgálniuk a felhasználók tevékenységeire (például „kiválasztás", „törlés"). Ez azt vonja magával, hogy a memóriában tárolt adatokon operál, nem pedig a lemezen. Ezért rendszerhiba esetén elveszhetnek az adatok, tehát gondoskodniuk kell a helyreállításról.
3. A szerkesztési munkamenetek rendszerint sokkal hosszabbak az online rendelés vagy más tranzakciók idejénél. Ez szintén azt jelenti, hogy nagyobb az adatvesztés kockázata, ha valamilyen probléma történik. Ezért az ilyen rendszerek tartalmazni szoktak egy helyreállítási lehetőséget, amely egy rendszerhiba esetén automatikusan elment minden folyamatban lévő munkát, ami alapján a helyreállítás elvégezhető.

Egy szerkesztőrendszer általános architektúrája a 13.10. ábrán látható. Ez együttműködő objektumok összességét takarja. A rendszer objektumai aktívak, nem pedig passzívak (14. fejezet), önállóan és konkurensen képesek működni. Lényegében a képernyőn bekövetkező események feldolgozása és parancsként értelmezése történik. Ez módosítja az adatszerkezetet, amely aztán újra megjelenítésre kerül a képernyőn.

A 13.10. ábrán látható architekturális komponensek felelősségei a következők:

1. Képernyő. Ez az objektum figyeli a képernyő memóriaszegmensét, és felderíti, hogy milyen események következtek be, majd átadja ezeket az eseményfeldolgozó objektumnak a képernyő-koordinátákkal együtt.
2. Esemény. Ez az objektum fogadja a Képernyő-től érkező eseményeket. Azt az ismeretet felhasználva, hogy mi van megjelenítve a képernyőn, értelmezi az eseményt, és lefordítja a megfelelő szerkesztő parancsra. Ez a parancs ezután a parancs értelmezéséért felelős objektumnak adódik át. Nagyon gyakori események (például egérkattintás, billentyűlenyomás) esetén az eseményobjektum közvetlenül az adatszerkezettel kommunikál, ami lehetővé teszi annak gyorsabb frissítését.
3. Parancs. Ez az objektum feldolgoz egy, az eseményobjektumtól érkező parancsot, és meghívja a Szerkesztő adat objektum megfelelő módszerét.



13.10. ábra. Egy szerkesztőrendszer architekturális modellje

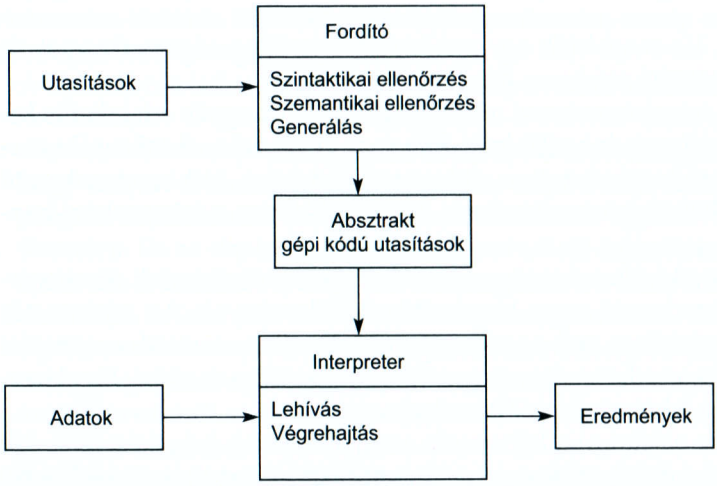
4. Szerkesztő adat. Ha meghívják egy metódusát, frissíti az adatszerkezetet, és meghívja a Megjelenítő objektum Frissít módszerét.
5. Kiegészítő adat. Az adatszerkezet mellett a szerkesztők egyéb adatokat is kezelnek, például stílusok és beállítások. Ebben az egyszerű architekturális modellben ezeket egyben belevettem ebbe az objektumba. Néhány szerkesztő parancs, például a helyesírás-ellenőrzés, is ezen objektum módszereként implementálható.
6. Állományrendszer. Az állományok megnyitását és mentését kezeli. Ezek egyaránt lehetnek szerkesztő vagy kiegészítő adatállományok. Az adatvesztés elkerülésének érdekében sok szerkesztő rendelkezik automatikus mentési funkcióval, amely az adatszerkezet mentését automatikusan végzi. Rendszerhibát követően ez lehet a helyreállítás alapja.
7. Megjelenítő. Ez az objektum a képernyőn megjelenítendő dolgokért felel. Ha a megjelenítendő adatok változtak, a Képernyő objektum Frissít módszerét meghíva a képernyő újrarajzolásra kerül.

Mivel a felhasználói utasításokra gyors választ szükséges adni, a szerkesztőrendszereknek nincs központi vezérlője, amely meghívná ezeket a komponenseket. Ehelyett a rendszer kritikus komponensei konkurensen futnak és közvetlenül kommunikálnak egymással (például az eseményfeldolgozó közvetlenül kommunikál a szerkesztő adatszerkezetével), így gyorsabb lesz a teljesítmény.

13.4. NYELVFELDOLGOZÓ RENDSZEREK

A nyelvfeldolgozó rendszerek bemenete egy természetes vagy mesterséges nyelv, kimenete pedig az adott nyelv egy másfajta reprezentációja. A szoftverfejlesztés során a legszélesebb körben alkalmazott nyelvfeldolgozó rendszerek a fordítóprogramok, amelyek egy mesterséges, magas szintű programozási nyelvet gépi kóddá fordítanak, de más nyelvfeldolgozó rendszerek például egy XML-formátumú leírást egy adatbázis lekérdezését végző utasításokká tudják fordítani, míg megint más nyelvfeldolgozó rendszerek egyik természetes nyelvet egy másikra próbálnak meg lefordítani.

A 13.11. ábrán egy nyelvfeldolgozó rendszer absztrakt architektúrája látható. Az utasítások írják le, hogy mit kell tenni. Ezeket a fordító valamilyen belső formátumra alakítja. Az utasítások egy absztrakt gép gépi utasításainak felelnek meg. Ezeket az utasításokat ezután egy másik komponens (az interpreter) értelmezi és végrehajtja, szükség esetén a környezetből kapott adatokat felhasználva. A folyamat kimenete a bemeneti adatokon értelmezett utasítás eredménye. Sok fordítóprogram esetén az interpreter a gépi utasításokat feldolgozó hardverelem, az absztrakt gép pedig a számítógép processzora. A Java nyelv esetén azonban az interpreter egy szoftverkomponens.



13.11. ábra. Egy nyelvfeldolgozó rendszer absztrakt architektúrája

A nyelvfeldolgozó rendszereket akkor alkalmazzuk, amikor egy problémát legkönnyebb úgy megoldani, hogy a megoldást algoritmusként vagy a rendszer adatainak leírásaként adjuk meg. Például a meta-CASE-eszközök olyan programgenerátorok, amelyek a szoftvertervezési módszerek segítésére CASE-eszközöket hoznak létre. A meta-CASE-eszközök tartalmazzák a módszer komponenseinek leírását, annak szabályait stb. Ezek egy speciálisan erre a célra kifejlesztett nyelven vannak megadva, amelynek az elemzésével lehet a generált CASE-eszköz konfigurálását végezni.

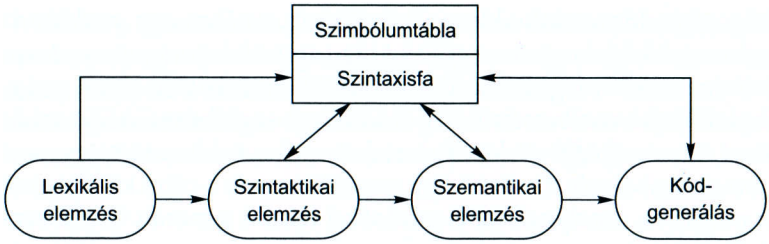
Egy nyelvfeldolgozó rendszer fordítói rendelkeznek egy általános architektúrával, amely a 13.12. ábrán látható, és a következő komponenseket tartalmazza:

1. Egy lexikális elemzőt, amely megkapja a bemeneti nyelv lexikális elemeit, és azokat valamilyen belső formára alakítja.
2. Egy, a lexikális elemző által felépített szimbólumtáblát, amely a fordítandó szövegben használt egyedek (változók, osztálynevek, objektumnevek stb.) neveiről tartalmaz információt.
3. Egy szintaktikai elemzőt, amely a fordítandó szöveg szintaktikáját ellenőrzi. Ezt a nyelv egy meghatározott nyelvtanával végzi és felépít egy szintaxisfát.
4. Egy szintaxisfát, amely a fordítandó programot egy belső szerkezetben reprezentálja.
5. Egy szemantikai elemzőt, amely a szintaxisfa és a szimbólumtábla információi alapján a bejövő nyelvi szöveg szemantikai helyességét ellenőrzi.
6. Egy kódgenerátort, amely a szintaxisfát bejárva absztrakt gépi kódot generál.

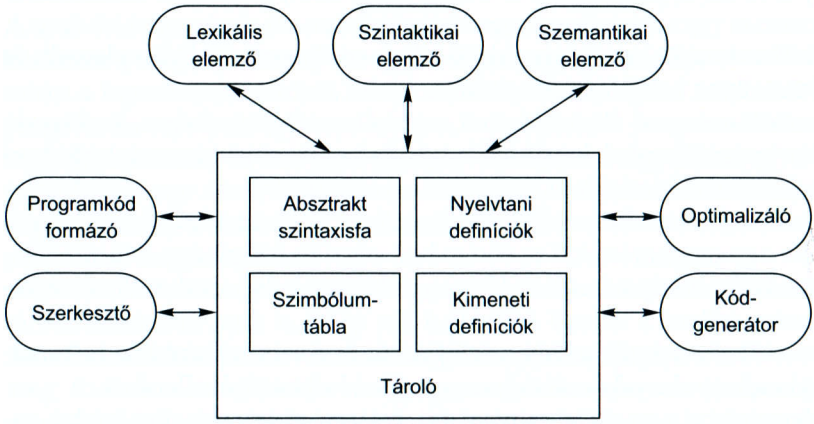
Tartalmazhat még egyéb komponenseket is, amelyek a szintaxisfa átalakításával a generált gépi kód hatékonyságát növelik, és redundanciáját csökkentik. A természetes nyelvi fordítók esetén a generált kód a bemeneti szöveg valamely más nyelvre lefordítva.

A nyelvfeldolgozó rendszereket felépítő komponensek különböző architekturális modellek szerint szervezhetők. Ahogyan Garlan és Shaw (Garlan és Shaw, 1993) rámutat, a fordítóprogramokat egy összetett modell alapján lehet implementálni. Ehhez az adatfolyam-architektúra használható, kiegészítve a megosztott adatok tárolójaként funkcionáló szimbólumtáblával. A lexikális, szintaktikai és szemantikai elemzés szakaszai a 13.12. ábrán látható módon, szekvenciálisan szervezettek.

A fordításnak ezt az adatfolyammodelljét széles körben használják. Igen hatékony olyan kötegelt környezetekben, ahol a programokat felhasználói beavatkozás nélkül fordítják és futtatják. Azonban kevésbé hatékony abban az esetben, amikor a fordítóprogramot más nyelvi feldolgozóeszközökkel – például strukturált szövegszerkesztővel, interaktív nyomkövetővel, belövővel, programkódformázóval stb. – kell integrálni. Az általános rendszerkomponenseket egy tárolóalapú modellbe szervezhetjük, mint ahogyan az a 13.13. ábrán látható.



13.12. ábra. Egy fordítóprogram adatfolyammodellje



13.13. ábra. Egy nyelvfeldolgozó rendszer tárolási modellje

Az ábrán az látható, hogy egy nyelvfeldolgozó rendszer hogyan lehet integrált programozástámogató eszközök része. Ebben a példában a központi információtároló szerepét a szimbólumtábla és a szintaxisfa tölti be. Az eszközök és eszközrészek ezen keresztül kommunikálnak. Az egyéb információk, mint a nyelvtannak és a program kimenő formátumának definíciója kikerülnek az eszközökből és bekerülnek a tárolóba. Ezért a szövegszerkesztő már gépelés közben ellenőrizni tudja, hogy a program szintaktikailag helyes-e, egy kódformázó pedig könnyen olvashatóvá tudja alakítani a programszöveget.

Kulcsfogalmak

- Az alkalmazásarchitektúrák általános modelljei segítenek megérteni az alkalmazások működését, összehasonlítani azonos alkalmazástípusba tartozó alkalmazásokat, és ellenőrizni az alkalmazás tervét.
- Sok alkalmazás vagy beleesik az általános alkalmazások négy kategóriájának valamelyikébe, vagy azok kombinációjaként írható le. Az itt tárgyalt négy általános alkalmazás közé az adatfeldolgozó, a tranzakciófeldolgozó, az eseményfeldolgozó és a nyelvfeldolgozó alkalmazások tartoznak.

- Az adatfeldolgozó rendszerek kötegelten működnek, és általában egy bemenet-feldolgozás-kimenet szerkezettel írhatók le. A rendszer bemenetén rekordok jelennek meg, majd az információ feldolgozásra kerül, és létrejönnek a kimenetek.
- A tranzakciófeldolgozó rendszerek olyan interaktív rendszerek, amelyek lehetővé teszik, hogy egy adatbázisban tárolt információt a felhasználók távolról elérjenek és módosítsanak. Ilyen rendszerek az információs rendszerek és az erőforrás-kezelő rendszerek.
- Az eseményfeldolgozó rendszerek közé a szerkesztő- és a valós idejű rendszerek tartoznak. Egy szerkesztőrendszerben a felhasználói felület eseményeket értelmezik, melynek hatására a memóriában tárolt adatszerkezet megváltozik. Ilyen rendszerek a szövegszerkesztő és prezentációkészítő rendszerek.
- A nyelvfeldolgozó rendszerek szövegek egyik nyelvből a másikba alakítását és a bemeneti nyelven megadott utasítások végrehajtását végzik. Egy fordítóból és egy absztrakt gépből állnak, mely utóbbi a generált nyelv végrehajtását végzi.

További irodalom

Az alkalmazásarchitektúrák témája nagyon elhanyagolt, a szoftverarchitektúrákról szóló könyvek és cikkek szerzői vagy az absztrakt elvekre, vagy a termékvonal-architektúrákra koncentrálnak.

Databases and Transaction Processing: An Application-oriented Approach. Ez a könyv valójában nem szoftverarchitektúráról szól, de a tranzakciófeldolgozó és adatközpontú alkalmazások alapjait tárgyalja. (Lewis, P. M. *és mások,* 2003, Addison-Wesley.)

Design and Use of Software Architectures. Ez a könyv a szoftverarchitektúrák termékvonal-megközelítésével foglalkozik, és így az alkalmazás szemszögéből tárgyalja az architektúrát. (Bosch, J., 2000, Addison-Wesley.)

Feladatok

13.1. Magyarázza el, hogy az itt bemutatott általános alkalmazásarchitektúrák hogyan segítik a tervező újrafelhasználással kapcsolatos döntéseit.

13.2. A fejezetben bemutatott négy alapvető alkalmazásfajta segítségével osztályozza az alábbi rendszereket, indokolást is adva:

- elárusítórendszer egy szupermarketben;
- újság-előfizetések fizetési határidejéről emlékeztetőket küldő rendszer;
- fotóalbum, amely régi fotók helyreállítására is lehetőséget biztosít;
- gyengén látók számára weblapokat felolvasó rendszer;
- interaktív játék, amelyben a szereplők mozognak, akadályokat kerülgetnek és kincseket gyűjtenek;

- leltári rendszer, amely figyelemmel kíséri, hogy milyen elemekből van raktáron, és ha valamiből egy meghatározott értéknél kevesebb van, akkor automatikusan rendelést generál.

13.3. A bemenet-feldolgoz-kimenet modell alapján bővítse ki a 13.2. ábra Fizetés kiszámítása függvényét, és rajzoljon egy olyan adatfolyam-diagramot, amely a függvény által végzett számításokat mutatja be. Ehhez az alábbi információk állnak a rendelkezésére:

- Az alkalmazott rekordja azonosítja az alkalmazott besorolását, majd ezt a besorolást kell keresni a fizetési sávok táblában.
- Egy bizonyos besorolás alatti alkalmazottak ugyanolyan árat kaphatnak a túlóráért, mint amilyen a normál órabérük. A fizetendő túlórák az alkalmazott rekordban jelölve vannak.
- A levonásra kerülő adó mennyisége függ az alkalmazott adókódjától (ez benne van a rekordban) és az éves fizetésétől. A havi levonások és a normál fizetés kódonként megtalálható az adótáblákban. Ezeket az aktuális és a normál fizetés közötti kapcsolattól függően skálázzák.

13.4. Magyarázza meg, hogy miért van szükség tranzakciókezelőre egy olyan rendszerben, ahol a felhasználói bemenetek az adatbázis megváltozását eredményezik.

13.5. Az információs rendszerek 13.6. ábrán látható alapmodellje segítségével mutassa be, hogy milyen komponensei vannak egy olyan információs rendszernek, amely lehetővé teszi az egy adott repülőtérről fel-, illetve oda leszálló járatok információinak megtekintését.

13.6. A 13.8. ábrán látható rétegzett architektúra segítségével mutassa be, hogy egy hotelszoba-foglalásokat kezelő erőforrás-kezelő rendszer milyen komponensekből áll.

13.7. Egy szerkesztőrendszerben a felhasználói felület események implicit vagy explicit parancsokká fordíthatók. Magyarázza meg, hogy akkor mégis miért kommunikál az Esemény objektum is közvetlenül a szerkesztő adatszerkezetével, nem csak a Parancs.

13.8. Módosítsa úgy a 13.10. ábrát, hogy az egy táblázatkezelő rendszer általános architektúráját mutassa. A terv alapjául az ön által használt táblázatkezelők jellemzői szolgáljanak.

13.9. Mi a szintaxisfa komponense szerepe a nyelvfeldolgozó rendszerekben?

13.10. A nyelvfeldolgozó rendszerek itt bemutatott általános modellje segítségével tervezze meg egy természetes nyelvű parancsokat elfogadó és azokat adatbázis-lekérdezésekké (például SQL-re) fordító rendszer architektúráját.