Az Egerfood információs rendszer

doktori (Ph.D.) értekezés

Radványi Tibor Zsolt

Témavezető: Dr. Kormos János

Debreceni Egyetem
Természettudományi Doktori Tanács
Informatika Doktori Iskola
Debrecen, 2010

Ezen értekezést a Debreceni Egyetem Természettudományi Doktori Tanács Informatika Doktori Iskola programja keretében készítettem a Debreceni Egyetem informatika doktori (PhD) fokozatának elnyerése céljából.

Debrecen, 2010. április 20.

.....
Radványi Tibor Zsolt
jelölt

Tanúsítom, hogy Radványi Tibor Zsolt doktorjelölt 2008 – 2010 között a fent megnevezett Doktori Iskola Informatika programjának keretében irányításommal végezte munkáját. Az értekezésben foglalt eredményekhez a jelölt önálló alkotó tevékenységével meghatározóan hozzájárult. Az értekezés elfogadását javaslom.

Debrecen, 2010. április 20.

.....
Dr. Kormos János
témavezető

Az Egerfood információs rendszer

értekezés a doktori (PhD) fokozat megszerzése érdekében az informatika tudományágban

Írta: Radványi Tibor Zsolt, okleveles matematika-fizika-informatika szakos tanár.

Készült a Debreceni Egyetem Informatika Doktori Iskolája (Informatika programja) keretében.

Témavezető: Dr. Kormos János

A doktori szigorlati bizottság:

elnök: Dr. ....
tagok: Dr. ....
Dr. ....

A doktori szigorlat időpontja: 2008. július 15.

Az értekezés bírálói:

Dr. ....
Dr. ....
Dr. ....

A bíráló bizottság:

elnök: Dr. ....
tagok: Dr. ....
Dr. ....
Dr. ....
Dr. ....

Az értekezés védésének időpontja: 200. .....

Köszönetnyilvánítás

Ezúton fejezem ki köszönetemet témavezetőmnek Dr. Kormos Jánosnak, aki dékáni teendői mellett is tudott időt szakítani arra, hogy folyamatosan irányítson és segítsen a PhD tanulmányaim végzésében és a doktori disszertációm írásában.

Köszönöm Dr. Liptai Kálmánnak és Dr. Kovács Emődnek a Matematikai és Informatikai Intézet vezetőinek, hogy tanulmányaim során folyamatosan támogattak jó tanácsaikkal, és biztosították a szükséges időt a kutatások elvégzésére. Köszönöm Dr. Kusper Gábornak, akivel sok órát törtük a fejünket, hogy ez a rendszer megfeleljen a követelményeknek, és aki mindenben segítette a munkámat

Köszönetemet fejezem ki Dr. Kiss Attilának, az Egerfood Regionális Tudásközpont igazgatójának, hogy részt vehettem a projekt munkáiban, és ezzel lehetővé tette a disszertáció megszületését.

Ajánlom feleségemnek, Valikának ...

Tartalomjegyzék

Ábrák jegyzéke

Táblázatok jegyzéke

1. fejezet

Bevezetés

1.1. A témaválasztás indoklása, motivációim

1993 óta dolgozom pedagógusként. Pályám első 10 évében matematika-fizika és informatika szakos tanárként oktattam a diákokat egy középiskolában. Az ezt követő 7 évben jelenlegi munkahelyemen, az Eszterházy Károly Főiskola Számítástudományi tanszékén tanítottam.

Már az egyetemi éveim alatt is érdekelt a kutatás, az új felfedezése. Akkor még a fizika tudomány, neutronfizika területén dolgoztam. A munka eredményeként született meg egy referált cikk neutronfizikából. Mindig vonzódtam a méréshez, az empírikus megtapasztaláshoz.

Már a középiskolai tanításom során is sokszor kerültem olyan helyzetbe, hogy szoftverfejlesztéses projektben kellett részt vennem, esetleg irányítani azt. Sikerült végigkövetni a fejlesztőeszközök fejlődését egészen a HT-1080Z Basic interpreterjétől kiindulva a Commodore korszakon keresztül, a IBM-PC kompatibilis gépekig. A Pascal, a Delphi nyelv fejlődését követhettem mint tanár, és mint fejlesztő is. Később a Visual Studio fejlődésével és a C# nyelv megjelenésével és térhódításával, ezzel az eszközzel is megismerkedtem. Mindig törekedtem arra, hogy a fejlesztéskor a felhasználói igények maradjanak az előtérben. A tanítás során pedig arra próbáltam nevelni a leendő informatikusokat, hogy olyan termékeket készítsenek, melyek mind használhatóságban, mind hatékonyságban a lehető legjobban kihasználják a fejlesztő eszközök lehetőségeit.

Amikor a főiskolára kerültem, akkor a pedagógiai előéletem nagy segítséget nyújtott, és felvetődött, hogy módszertani irányokban kezdek kutatásokba. Ennek nyomai mind a mai napig megvannak, hiszen ez egy olyan terület, melyre a felsőfokú oktatásban is nagy szükség van. Nem is szeretnék

szakítani vele a későbbiek során sem.

A másik irány az informatika tudományon belül való munkálkodás. Ebben nem volt akkora tapasztalatom, hiszen korábbi munkám során ilyen jellegű feladatokat kevésbé kellett teljesíteni. Több terület felvetődött, végül az adatbáziskezelés, az adatbázis-kezelő rendszerek vizsgálata és elválaszthatatlanul a szoftveres technológiák használata és vizsgálata maradt meg kutatási vonulatként.

A főiskolai tanításom során az ehhez az irányhoz tartozó tárgyak oktatása is a feladatommá vált. Így volt sok olyan megállapítás, amit akár a hallgatók is kétkedve fogadtak, amikor különböző adatbáziskezelési technikák alkalmazásáról volt szó. Meggyőzni őket azzal lehetett, ha megmértük, hogy az állítás igaz, vagy hamis. Természetesen ez módszertanilag is kiváló, hiszen közben programozunk, megismerjük az egyes rendszerek tulajdonságait, és saját tapasztalat útján sajátítja el a hallgató az ismereteket. Ennél hatékonyabb tanulási-tanítási módszer nem létezik. A hatékonyságmérések eredményeit később felhasználtuk a munkánk során, és az értekezésemben kifejtett rendszer fejlesztése közben is.

Amikor a PhD tanulmányaimat elkezdtem, a főiskolánkon, a természettudományi karon a biológus kollégák egy pályázatot nyertek, amivel Regionális Tudásközpontot hoztak létre. A RET célja az élelmiszerbiztonsági kutatások folytatása.

A kutatási-szolgáltatási tevékenységek középpontjában jelenleg környezetvédelmi és élelmiszeranalitikai munkák állnak, melyek közül kutatás-fejlesztési, valamint gazdasági és társadalmi szempontból az élelmiszeranalitikával és élelmiszerbiztonsággal kapcsolatos tevékenységek [1][2] a legjelentősebbek. Ezért a létesített élelmiszerbiztonsági és analitikai vizsgálati centrum az eddigi tevékenységek logikus folytatásának, bizonyos új súlypontok kialakításának és a gazdaságilag legrelevánsabb kutatási témák kiterjesztésének tekinthető.

Szükségük volt egy információs rendszerre, melynek segítségével nyomonkövethetik a konzorciumi partnereik által előállított élelmiszerek életútját.

Komoly elhatározásunk az, hogy az Észak-Magyarországi Innovációs Stratégiával összhangban a K+F és innovációs képességek fejlesztésével, valamint a hazai élelmiszerbiztonsági kutatási tevékenységek összehangolásával és kiterjesztésével a gazdasági szféra szereplői számára is értékes eredmények szolgáltatásával járuljunk hozzá a hazánkban előállított élelmiszerek versenyképességének növeléséhez.

Összeállt egy kis fejlesztő csoport, melynek tagja lettem. Feladatom az adatbázis rendszer tervezése, kialakítása volt, illetve a szoftverfejlesztés tá-

mogatása. Elsősorban olyan megközelítést vártak tőlem, mely szakít a hagyományos adatbázis felépítéssel. Relációs adatbázist használ, de egy olyan, eddig még nem megvalósított új koncepció létrehozásával, mely lehetővé teszi a gazdasági szereplők, és az általuk előállított termékek felvitelét a rendszerbe minden későbbi változtatás nélkül. Mivel a cégek és a termékeik előállításának folyamata akár gyökeresen is eltérhetnek egymástól, így a hagyományos tervezési módszertannal szakítani kellett, és új szempontból kellett megközelíteni és felépíteni a rendszert. Ez tükröződik mind az adatbázis logikai szerkezetében, mind a szoftverrendszert kiszolgáló segédszoftver alapgondolatában is.

1.2. Helyzetelemzés, tézisek

A létesített élelmiszerbiztonsági kutatással, nyomonkövetési rendszerek fejlesztésével foglalkozó centrum az integrált kutatások, valamint technológiai K+F révén nemzetközi tekintetben is innovatív tevékenységet végez, számos új magyar terméket, know-how-t és szabadalmat hoz. Létrehoztuk a vizsgált élelmiszer termékpályák egységes, új szempontú, komplex nyomonkövetési modellrendszerének prototípusait is. A multidiszciplináris megközelítés eredményeképpen feltárul az összes technológiai állomás és termelési lépcső minden egyes kockázati tényezője, és mód nyílik a kritikus technológiai eljárások fejlesztésére. A nyomonkövetési vizsgálatok a következő termékpályákra terjednek ki: Egri Bikavér bor, Detki Háztartási keksz, Hűtött friss réteslap, Tóth csípős kolbász, csiperke gombakonzerv, halkonzerv.

A projektben az informatika platform felé több feladatot fogalmaztak meg. Ezekből több érintette a kommunikációt a rendszeren belül, és azon kívül. [56, 63] A projekt belső kommunikációját működtető web rendszer elkészítése és folyamatos működtetése. A célja a projektben dolgozók közötti információáramlás biztosítása. Megoldásra került a dokumentumok tárolása és kezelése. Gondoskodtunk a mindennapi információcsere eszközéül szolgáló elektronikus levelezés archiválásról, a munkacsoportok számára levelezési listákat, fórumokat hoztunk létre. Ezeket a feladatokat a Novell cég által fejlesztett GroupWise rendszer oldotta meg.

Megoldandó feladat: Lehetséges-e olyan általános szerkezetű, mégis relációs adatbázis tervezése, mely segítségével a cégek és a termékeik előállításának folyamatai későbbi bővítések, vagy partícionált adatbázis használata nélkül tárolhatóak? A felvetés azért fontos, mert az adatbázis tervezési gyakorlat nem az általános megoldásokat keresi. A probléma fontos, mert egy-egy új termék gyártási folyamatának a bevezetésének ideje néhány hétről 2-3 napra csökkenhet. Így az előzetes elemzés során az a döntés született, hogy a több kutatással és tervezési munkával járó út lesz bejárva.

Fontos feladat a nyomkövetési adatbázis szerkezetének meghatározása, az adatátvitel hardveres és szoftveres kialakítása. Az informatikai rendszer gerincét a nyomkövetési rendszer adatbázisa képezi. A begyűjtött adatokat, követelményeket elemeztük, és ezek alapján megalkottuk az információs rendszer adatmodelljét. A formalizált modellt ellenőrzéseknek vetettük alá, majd a validált modell alapján megterveztük az adatbázis szerkezetét.

1. tézis. *Az adatbázis szerkezetének kialakításánál egy általánosan használható, általános felépítésű, sémamódosítás nélkül bővíthető rendszert tervezzünk, mely alkalmas a kijelölt feladatok megoldására, az elosztott adatbázis tárolására és a migrációs feladatok teljesítésére.*

Megoldandó feladat: Milyen üzleti vagy ingyenes adatbázis-szervert használjunk? El lehet ezt dönteni empirikus eszközökkel? Lehet-e mérési stratégiát kidolgozni arra a problémára, hogy az adott feladat elvégzéséhez a legalkalmasabb adatbázis szervert ki tudjuk választani.

2. tézis. *Hatékonysági mérések segítségével el lehet dönteni, hogy az adott feladatra milyen SQL-szervert érdemes használni.*

Megoldandó feladat: A szerver kiválasztása után a kérdés, hogy egy adott szerver esetén melyik adatelérési protokoll használata eredményezi a leghatékonyabb és legbiztonságosabb adatkezelést. Eldönthető-e méréssel az MSSQL szerver esetén, hogy milyen módszerrel a leghatékonyabb az adatkezelés?

3. tézis. *Az MSSQL szerver esetén a tárolt eljárások alkalmazása 2,5-3-szoros sebességnövekedést eredményez tranzakcióvezérelt adatfelvitel esetén.*

A nyomkövetési rendszer hosszú távú fejlesztési stratégiájának figyelembe vételével kidolgoztunk egy termékazonosításra alkalmas algoritmust és kódrendszert, amellyel született kódot a terméken a nyomkövetési rendszerben való szereplést igazoló garanciajegy hordoz. Fontos hogy az adatok megfelelő kriptográfiai eljárással titkosítva kerüljenek tárolásra és mozgatásra.

Megoldandó feladat: Létre kell hozni felhasználói interfészeket a különböző adatgyűjtő és lekérdezési tevékenységekhez. Oly módon, hogy egy új cég, vagy egy meglévő cég új termékének rendszerbe integrálásakor a lehető legrövidebb időn belül működhessen a rendszer, és a kliensprogram is.

4. tézis. *Létre lehet hozni egy munkafolyamat gráfnak nevezett objektumot, mely segítségével a kliensprogram felülete is generálható a termék előállításának folyamata alapján.*

Megoldandó feladat: Létrehozható-e egy megfelelő titkosítási biztonsági rendszer, mely az elosztott információs rendszer minden szegmensét képes védeni, hatékony, és megfelelően erős az ipari titkok védelmére?

5. tézis. *Lehetséges az, hogy olyan többrétegű kriptográfiai rendszert hozzunk létre, mely megfelelő biztonsági szintet biztosít az ipari titkok tárolására, és védelmet biztosít a kommunikáció közbeni lehallgatás ellen. Ez a rendszer az AES-128 - WCF* [1] *- VPN* [2] *hármas lesz.*

Kialakítottuk a fogyasztókkal WAP-on, ill. interneten való kommunikáció tartalmi szempontjait. Megtörtént az általános élelmiszerbiztonsági információk különböző részletességű platformokon való megjelenítése. Megterveztük és megvalósítottuk a teljes információs rendszer biztonsági követelményeit és a biztonsági eljárásokat.

1.3. Irodalmi előzmények

A munka elkezdésekor három nagy csomópont köré csoportosítottuk a kutatás irányát. Az első az adatbázissal kapcsolatos eredmények, melyek felhasználásával, és továbbfejlesztésével kívántunk eredményeket elérni. Ezekben

[1] Windows Communication Foundation
[2] Virtual Privat Network

fontos volt az adatbázisok kialakításának lehetőségei és a hatékonyság mérésének és értékelésének eredményei.

Másodsorban az adatok titkosítása, a kommunikáció az egyes telephelyek között. Azokat az eredményeket kerestük, melyek ebben a munkában segítenek.

A harmadik csoport a szotverfejlesztés, a felhasználói interfészek kialakítása, az objektum orientált programozási módszerek használata volt. Fontos szempont volt hogy a lehető legmodernebb eszközöket használjuk, valamint a fejlesztett rendszer egy összehangolt, egymásnak legjobban megfelelő fejlesztő eszközökkel legyen kialakítva. Így esett a választás a Microsoft által kiadott Visual Studió eszközre, mely tartalmazza a C# [3] programozási nyelvet, és olyan eszközöket, melyek segítették a munkánkat. Ekkor a framework 3.0 keretrendszer volt a legújabb, azóta a 3.5 is megjelent, így ennek osztályait tudtuk használni.

1.3.1. Az adatbázis fejlesztéshez tartozó előzmények

Nagyon fontos kérdés volt az adatbázis-kezelő rendszer vizsgálatánál, hogy milyen lehetőségek vannak. A DBMS [4] értékesítők milyen kínálatot biztosítanak a felhasználóknak, és milyen feltételekkel. A [16] cikkben a szerző megvizsgálta a kínálatot, és a piacon tapasztalható versenyhelyzetet. Azt lehetett tapasztalni, hogy a DBMS fejlesztők hatékonyan támaszkodnak az operációs rendszer szolgáltatásokra. Különösen, ha a DBMS és az operációs rendszer gyártója ugyanaz a cég. Ekkor egymást kiegészítő belső eszközökkel ruházzák fel a rendszereket. AZ ERP [5] rendszerek fejlesztésekor fontos tényező a helyes DBMS kiválasztása.

A [17] cikk áttekinti, hogy milyen lehetőségek vannak az ipari adatbázisrendszerek használatára elosztott ellenőrzési rendszerekben. Megtalálhatjuk benne, hogy milyen fontos szoftverfejlesztési és adatbázis tervezési kérdések merülnek fel a fejlesztés során. Áttekinti a lehetőségek széles tárházát annak, hogy a korszerű, cégen belüli ellenőrző rendszerek mögött milyen adatbáziskezelő rendszerek állhatnak. Hogyan tudják a mobilitást és a rugalmasságot biztosítani a modularitás segítségével. Megtaláljuk annak az elemzését, hogy milyen elterjedt ipari platformok vannak. Segítséget nyújtott a cikk az előttünk álló feladatok és lehetőségek pontosabb körülhatárolásában. A lehetőségeink felmérésében.

[3] C-Sharp

[4] Database Management System: Adatbázis-kezelő rendszer

[5] Enterprise Resource Planning: Vállalat irányítási rendszer

Hasonló vizsgálatot végeztek Malajziában. Ennek eredményét írja le a [18] cikk. A kis- és középvállalatok szintjén fontos annak a megvilágítása, hogy az informatikai eszközök, köztük az adatbázis-alkalmazások, illetve adatázis-kezelők technológiai szintjének növelése előnyös a profittermelésben, és az üzletmenet javításában. Az adatbázis-alkalmazások (mint például adatbázis-kezelő rendszer, adattárház és az adatbányászat) alkalmasak az információk hatékony eszközökkel való tárolására, kezelésére, így hasznos információkat nyújt az üzleti tevékenységről, mint a vásárlói számlák, szállítói kapcsolatok, mozgás leltár, beszerzés, marketing tervezés és egyéb üzleti tevékenység. Felismerve a potenciális lehetőséget az adatbázis technológiában, a cégek már most kihasználhatják ezeket az információkat annak érdekében, hogy hatékonyan kezeljék a beszállítói lánc tevékenységét.

A tanulmány kimutatta, hogy ha a gyártó pozitív hozzáállást mutatott az adatbázis technológia iránt, akkor az ellátási láncban részt vevő cégek tevékenységét is befolyásolni tudta. Bár általában kis-közepes vállalatok között a rendszerek elfogadása korai szakaszában jár, az eredmények azt mutatják, hogy a malajziai gyártók lelkesednek az adatbázis-technológia előnyeit iránt. Röviden, a tendencia az, hogy a cégek dinamikus adatbázist használnak az adatok kezelésének javítására a stratégiai tervezésben és az ellátási láncban.

Fontos kérdés volt az adatbázis logikai tervezése. Nagy relációs adatbázisok tervezést valamely adatbázis tervezési módszertan segítségével oldhatjuk meg [48]. Először is az ER[6] modellt használhatjuk a követelményelemzéskor felderített rendszer megjelenítésére. Majd transzformálhatjuk ezt a kiterjesztett EER modellé, és végül normalizálhatjuk a logikai tervet. A normalizálás elkerülhetetlen, az adatintegritás fentartása céljából.

A tervezés során definiáljuk az egységeket (entity), az attribútumokat, és az egyedekek közötti kapcsolatokat, azok típusával (1:1, 1:n, n:m). Hogy különböztetjük meg az egyedeket az attribútumoktól? Milyen irányelveket vegyünk figyelembe? Ahogy Teorey javasolja:

1. Az entitásoknak vannak leíró információik, míg az tulajdonságok nem rendelkeznek ilyennel.

2. A többértékű tulajdonságokat az entitások közé kell sorolni.

3. Olyan tulajdonságokat definiáljunk az egyedekben, melyek közvetlenül jellemzik azt.

4. Kerüljük el az összetett azonosítók használatát, amennyire csak lehetséges.

[6]Entity-Relationship: egység-kapcsolat modell

5. Normalizáljuk az egyedeket.

Meg kellett vizsgálni, hogy milyen adatbázis-modell alkalmazásától várhatjuk a legjobb eredményeket. [19] Figyelembe véve azt is, hogy várhatóan elosztott adatbázisrendszert kell kialakítani, és ennek a replikációját is meg kell oldani [20, 21]. Ez mutatta meg, hogy alkalmazható a pesszimista, az optimista illeteve a félig optimista megközelítése a prolémának. Mivel a rendszerünkben nem feltétlenül elsődleges szempont a replikációkor a gyors válaszadás, ezért a konkurens hozzáférést többszörös próbálkozással javíthatjuk. Választhatjuk az optimista megközelítést is.

A rendszer komponensei, a puffer szerverek, a központi adattárház, és a kliensek, ezek egymással szabványos és biztonságos módon kell hogy kommunikáljanak. A szabványt az XML formátum fogja szolgáltatni [23]. Hiszen ez az a forma, amiben a relációs adatbázisban tárolt adatok könnyen átvihetőek [22]. Tudjuk biztosítani a konkurens hozzáférést, a megfelelő zárak alkalmazásával. A tranzakciókezelés megvalósítható az XML adatbázisok használata esetén is. Erre szükségünk is volt a rendszer fejlesztésénél, hiszen OLTP[7] adatbázis modellt alkalmaztunk.

Az adatbázis rendszer kialakításakor gondolni kellett az adatbiztonság kiterjesztésére is. A biztonságos kommunikációt a szoftverrendszer, és az információs rendszer részeként a hálózati elemek biztosítják. Az adatbázisban használt kriptográfiai eljárások egy végső védelmi vonalat biztosítanak, arra az esetre, ha az adatbázist éri külső támadás. Ekkor a tárolt adatokat csak a belső kódolás védheti meg. Ez megtehető, több lehetőség is kínálkozik erre.[24, 25, 26].

A [27] cikkben elemzik a szerzők, hogy milyen kérdések merülnek fel az adattárolás és adatbiztonság területén. Milyen kompromisszumokra kényszerülünk ha az adatbiztonságot és a hatékonyságot állítjuk mérlegre. Javaslatot tesznek egy hatékony kulcskezelési protokollra, mely lehetővé teszi a biztonságos adattárolást.

1.3.2. A kriptográfiai előzmények

A biztonság a számítógép és a számítógépes kommunikációs alapú információs rendszerek elengedhetetlen része és feltétele. Az ipari vagy részben ipari környezetben működő rendszerekkel szemben kiemelten magas a biztonsági elvárások szintje. Hiszen a hálózati kommunikációban ipari titoknak minősülő adatok vesznek részt. Így a gazdasági résztvevők alapvető elvárása, hogy az adataik megfelelő szinten védve legyenek.

[7] OnLine Transaction Processing

Az EGERFOOD rendszerben 6 élelmiszeripari cég vesz részt, melyek megkövetelik a rendszertől a magas biztonsági fokozatot.

Ezek a kérdések nem csak tisztán technikai jellegű elvárások. Kiderült, hogy az emberi tényező legalább annyira fontos, mint a megfelelő kriptográfiai eljárások alkalmazása. [28] A cikk ebből a szempontból vizsgálja a problémát, ad egy lehetséges megközelítést, melyet mind a fejlesztők, mind a cégvezetők követhetnek.

A technikai tényezők is összetettek. A biztonsági megfontolások során figyelmebe kell venni az információs rendszer minden részterületét, szakaszát. Kezdve ott, ahol az adatok keletkeznek, áthaladva az átviteli közegeken és csatornákon, kiterjesztve a biztonsági módszereket az adatok tárolásának helyeire is.

A [32] cikkben a szerzők egy fontos kérdést járnak körbe. Megvizsgálják, hogy a wireless hálózatokon belül milyen hatékonysággal lehet alkalmazni a különböző kriptográfiai eljárásokat. [33] Ezt teszik mind a 2,4GHz, mind az 5GHz hálózati frekvenciákon. Fontos a vizsgálat, hiszen ezek a kriptográfiai eljárások jelentős erőforrást fogyasztanak, úgy mint CPU időt, memóriát, mobil eszközöknél akkumulátor töltést. A cikkben a szerzők összehasonlítanak 6 különböző titkosítási algoritmust, úgy mint AES (Rijndael), DES , 3DES, RC2, Blowfish, és RC6 [35, 34, 36, 37]. A vizsgálatok azt mutatták, hogy több esetben a Blowfish algoritmus teljesített a legjobban, de a változó adatmennyiséget és a változtatható kommunikációs protokollt figyelembe véve nem volt jelentős eltérés az egyes algoritmusok között. Amennyiben a jelerősség csökken a hálózaton belül, megnőhet a szükséges idő. A szerzők javasolnak egy újabb megközelítési módot is az algoritmusok és a protokollok kiválasztására. Ebben a megközelítésben a felhasznált energia minimalizálása az elsődleges szelekciós szempont. Ez fontos olyan rendszerek esetében, ahol a rendszer hardver elemei akkumulátoros energiaellátást használnak.

A fenti algoritmusok közül választunk mi is egy megfelelő, a fejlesztő eszköz által támogatott algoritmust, melynek a segítségével az adatokat már a keletkezésük helyén kódolhatjuk.

Az adatáramlás biztonságát jól tudja biztosítani a megfelelően kiépített VPN hálózat [40, 38]. Nem csak az oktatás a kutatás, hanem a nagyvállalti szférában is kielégítő adatbiztonságot tud biztosítani. A cikk bemutatja a VPN hálózatokat, a QoS [8] parancsokat és konfigurációt. A [39] cikkben leírja a szerző az IPSec értékelését, komoly kritikai megfogalmazásokat téve. Ezeket szem előtt kell tartani egy VPN hálózat tervezésekor. Túl bonyolultnak minősíti a részrendszert, nehezen kezelhető, és nem elég biztonságos. Lehe-

[8] Quality of Service

tőségként megfogalmazza a következtetésben, hogy az új AES algoritmus használatával egyszerűbbé és hatékonyabbá tehető a titkosítás. Ezt szem előtt tartottuk, amikor az információs rendszer kriptográfiai rendszere került tervezésre. Így az AES algoritmust használjuk, mintegy kiegészítésként a VPN hálózaton belül.

1.3.3. A kliensszoftverek fejlesztésének előzményei

A szoftverfejlesztéshez több irányból kellett közelíteni, hiszen egyrészt feladat volt a kliensszoftver megírása, a megfelelő fejlesztőeszköz kiválasztása, a fejlesztési stratégia eldöntése. Másrészt a kommunikációt kiszolgáló szerver oldali programok megírása, és tesztelése.

A [41] cikkben a szerző bemutatja a végeselem analízis problémakörében használható eszközöket az objektum-orientált fejlesztő eszközön. Részletesen tárgyalja a .NET keretrendszer lehetőségeit, és a szoftvertervezési szempontokat. A feladat nem egyezik a mi feladatunkkal, de az eszközök vizsgálata sokat segített az előrelépésben. A cikk kitér az elosztott problémák megvalósítására is, ami az informatikai rendszerünkben is lépten-nyomon fellelhető.

A szoftver fejlesztésekor sok feladat, és sok protokoll együttes kezelését, integrációját kellett megvalósítani. A [42] cikkben a szerzők az egyetemi infrastruktúra kialakításakor vizsgálják az integráció lehetőségeit, és tanulságait. Fontos megállapításuk, hogy az integrációs problémákat webservice [9] -ek alkalmazásával kívánják megoldani. A gondolat nagyban befolyásolta az Egerfood szoftver köztes rétegének kialakítását. A szerzők megvizsgálják a különböző adatbázis szervereket alkalmazó és különböző programozási nyelveken fejlesztett alkalmazások kommunikációs lehetőségeit. A vizsgálatba bevonták az Oracle, MS-SQL szerver, MySQL, DB2 adatbázis szervereket, valamint a programnyelvek közül többet, úgy mint C#, Delphi, Java, J2EE. A következtetés, hogy a webszolgáltatások segítségével sikerül összekapcsolni a részrendszereket. Ezt a gondolatot vettük át, azzal, hogy a szoftverünkre nem jellemző a többféle adatbázis szerver használata, bár a lehetőség megvan rá, mivel a cégek belső rendszere eltérhet a kommunikációs követelményeinktől, és a programnyelv is egységes volt.

A webszolgáltatások fejlesztéséhez kiváló segítséget nyújtott a [43] cikk. A szerző P2P hálózaton belül fejlesztett kommunikáció segítségével mutatja be a webszolgáltatások használatának metodikáját. Tanulságos forráskóddal bőven ellátott munka. A cikk segítségével betekintést kapunk a System.net,

[9] webszolgáltatás: szabványok és protokollok együttese, melynek feladata a hálózaton keresztük adatcsere megvalósítása a különböző alkalmazások között.

a System.Web.services névterek használatába.

A szerzők a [44] cikkben a .NET framework keretrendszert abból a szempontból vizsgálják, hogy megfelelő eszköz lehet-e a rendszer szintű modellezéshez és szimulációhoz. A hagyományosabbnak mondható C++ nyelvvel vetik össze. A .NET rendszer felett fejlesztett ESys.NET rendszert használják az összevetéshez, melyet C# nyelven fejlesztettek, mely egy erősen típusos objektum-orientált programozási nyelv. A SystemC-vel összehasonlítva a fejlesztett programjuk erősen támaszkodik a keretrendszer által nyújtott szolgáltatásokra. A legfontosabbak az XML kommunikáció, a webszolgáltatások használata, ahogy ezt ki is emelik a cikkben a szerzők. Ez irányvonalat mutatott nekünk arra, hogy melyek azok a .NET szolgáltatások, melyeket hatékonyan tudunk használni.

A [45] cikkben a szerzők egy információs rendszert mutatnak, melyet turisztikai célból hoztak létre. Elsősorban a WEB és a WAP technológiákat használták. Különböző hozzáférési jogosultságokat alakítottak ki, így mindkét technológia adatait mind statikusan, mind dinamikusan szolgáltatni tudták. Nagy hangsúly került a mobil interfészek használatára. Részletesen elemzik a szerzők a wap használat nehézségeit. A készülékek korlátozott tudását, az erősen szűkös sávszélességet, a relatív nagy költséget, mely a wapelérésel jár együtt. Hozzátehetjük, hogy az azóta eltelt 9 év (a fejlesztéskor ez 6-8) jelentősen átalakította a mobiltelefonok piacát. Ugrásszerűen fejlődtek a készülékek és a szolgáltatások. Ezzel párhuzamban a wap [46, 47] lehetőségeit is jobban ki lehet használni. Így bízhatunk benne, hogy a javasolt elérési protokolok nem tünnek majd megvalósíthatatlan igényeknek.

2. fejezet

Az adatbázis-modell kialakítása

2.1. Az adatbázis terv

Az adatbázis-modell kialakításánál több szempont figyelembe vétele vezérelt. Ezek az informatikai célok voltak: fogyasztó-központú modell kiépítésével az élelmiszerbiztonsággal kapcsolatos információ gyors, költséghatékony, megbízható továbbítása és feldolgozása és eljuttatása

- a fogyasztókhoz
- az élelmiszertermelőkhöz
- az érintett hatóságokhoz.

Intézményünk, az Eszterházy Károly Főiskolán létrehozott Regionális Tudásközpont, komoly elhatározása az, hogy az Észak-Magyarországi Innovációs Stratégiával összhangban a K+F és innovációs képességek fejlesztésével, valamint a hazai élelmiszerbiztonsági kutatási tevékenységek összehangolásával és kiterjesztésével a gazdasági szféra szereplői számára is értékes eredmények szolgáltatásával járuljon hozzá a hazánkban előállított élelmiszerek versenyképességének növeléséhez. A kidolgozásban részt vevő külső cégek a következő termékeikkel vesznek részt.

Vizsgált termékpályák: Egri Bikavér bor, Detki háztartási keksz, Hűtött friss réteslap, Tóth csípős kolbász, Csiperke gombakonzerv, Halkonzerv.

Az informatikai rendszer gerincét a nyomonkövetési rendszer adatbázisa képezi. A begyűjtött adatokat, követelmények elemezésre kerültek, és ezek

alapján meg lett alkotva az információs rendszer adatmodelljét. A nyomonkövetési rendszer hosszú távú fejlesztési stratégiájának figyelembe vételével ki lett dolgozva egy termékazonosításra alkalmas algoritmust és kódrendszert, amellyel született kódot a terméken a nyomkövetési rendszerben való szereplést igazoló garanciajegy hordoz. Fontos cél volt, hogy az adatok megfelelő kriptográfiai eljárással titkosítva kerüljenek tárolásra és mozgatásra.

2.1.1. Megvalósítási terveket befolyásoló követelmények

A központi adatbázis a központi adattárházban helyezkedik el. Minden EgerFood rendszert használó cégtől ide érkeznek be a naplók[1] adatai. Az adatbázis tervezése során az volt az elsődleges cél, hogy minden, a rendszerben szereplő cég összes adatát egységes rendszerben lehessen kezelni. Ez a követelmény egy hatékony, bár elég bonyolult és a hagyományokkal szakító adatbázis-szerkezetet eredményezett.

A központi tárház vállalja az adatok végleges tárolását és a lekérdező, megjelenítő modulok kiszolgálását.

Az első kérdés a termelési folyamatban kinyert adatok eljuttatása a tárházba. Erre több lehetőség is kínálkozik, melyek közül az első, hogy a kiépített hálózati rendszer segítségével online kapcsolatban lévő adatforrások folyamatosan szolgáltatják az adatokat közvetlenül a központ felé. Nézzük meg az adatforrásokat:

Két nagy területről érkeznek adatok: a főiskolai kutató laborok vizsgálati eszközeinek mért eredményei, illetve a távoli felhasználók, az ipari telephelyeken, a termelés során mért adatok. Ezek az adatok napi 24 órán keresztül, folyamatosan érkezhetnek, míg a laborok eredményei szakaszosan, a mérési kísérleteknek megfelelően.

A külső telephelyek hat különböző élelmiszeripari cég termelési helyei, ezeknek teljesen különböző a földrajzi elhelyezkedésük, az informatikai felszereltségük, és lehetőségeik. Az általuk előállított élelmiszeripari termékek is különbözőek, azaz gyártási technológiájuk eltéréséből adódóan a termékpályákban különböző a vizsgált adatok összetétele, és keletkezésük időintenzitása. Ezen eltérésekre a hálózatnak és az adatok tárolását megvalósító informatikai rendszernek fel kell készülni. Látható, hogy az adatok várható mennyisége megköveteli a nagyméretű központi tárház megépítését. Az érkező adatok időintenzitása pedig a nagy hálózati keresztmetszetet.

Az alapvető jellemzők mellett fontos követelmény az adatok nagy biztonsággal történő továbbítása és megőrzése. A biztonságos adattovábbítás

[1] a napló az entitás egy példányának adatait tartalmazza

jelenti egyúttal, hogy nem lehet adatvesztés hálózati vagy hardveres meghibásodás miatt sem. Legalább is minimálisra kell azt szorítani. Jelenti továbbá a mozgatott adatok megfelelő titkosítását, hiszen közöttük olyan ipari titok jellegű adatok is mozognak, melyek nem kerülhetnek külső körbe. Így nagy hangsúly kerül a kriptográfiai eljárások gyors és hatékony alkalmazására.

A központi adattárház feladata a kifelé menő adatok szolgáltatása, a lekérdezések kiszolgálása. Mivel a kifelé menő adatok mind Weben, mind Wapon, mind belső kommunikációs csatornákon keresztül, vastag kliensekkel elérhetőek lesznek, az ilyen jellegű terhelés a felhasználók számával és a módszer terjedésével dinamikusan fog nőni.

A követelmények és a várható fejlődés szükségessé tették, hogy a kiépítésre kerülő adat-beáramlási útvonal, és felépítményt alaposan átgondolva túl legyen biztosítva. Ezért indokoltnak tűnt, hogy a központi szerverhez az adatok ne közvetlenül, "nyersen", hanem egy előfeldolgozáson átesve, pufferelve kerüljenek. Ennek a két feladatnak az ellátására állítottunk be úgynevezett átmeneti, vagy pufferszervereket[2] a rendszerbe.

Feladat volt a pufferszerverek helyének és pontos tevékenységének megkeresése, körülhatárolása és megvalósítása. A pufferszerverek helyének a kutatásban részt vevő cégek azon telephelyeit jelöltük meg, ahol a vizsgált termékeket előállítják, továbbá a főiskolai kutatólabor munkáját is egy ilyen szerver fogja össze.

A szerverek feladata, hogy a tartományukba tartozó gépek és mérőműszerek adatait elő feldolgozzák, és megfelelő rendszerbe csoportosítva küldjék azokat tovább a központi szerver felé.

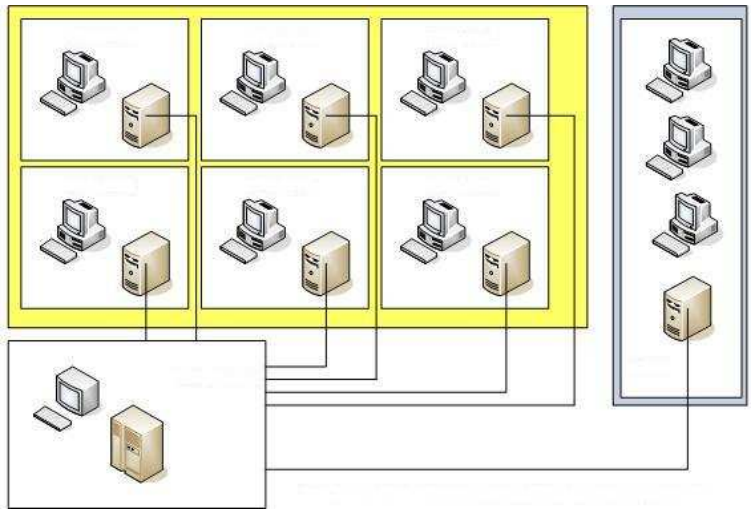
Fontos kérdés hogy az adatvesztést hogyan lehet a legnagyobb valószínűséggel elkerülni. Ennek két oldalát kell megvizsgálni.

- Az egyik a bekerült adatok hosszú távú védelme, melyet a jól átgondolt és kidolgozott archiválási rend hivatott biztosítani.

- A másik az adatforrásoknál előállított, de a központi szerverre még be nem került adatok rövid és hosszú távú védelme.

A rövid távú védelem alatt azt kell érteni, hogy az adat keletkezés pillanatához minél közelebb kell az első biztonsági mentés. Majd az előfeldolgozott, titkosított, a továbbításra felkészített adatokat újra menteni szükséges. Így a központi szerver adattárházába való bekerülés előtt az adatok már két, egymástól független helyen tárolásra kerülnek.

[2] A termelő cégeknél illetve a laborban elhelyezett adatbázis- és kommunikációsszerverek.



2.1. ábra. A külső cégek és a labor adatszolgáltatása

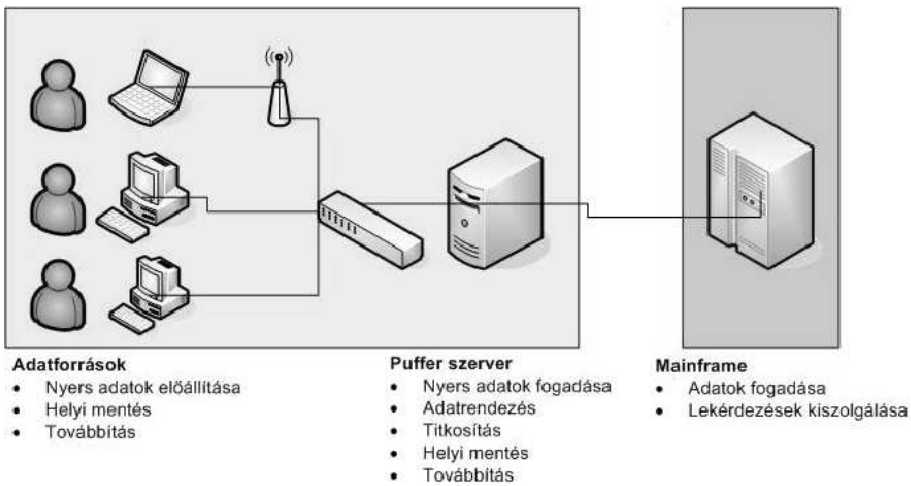
Ilyen mértékű adatredundancia igen erőforrás-igényesnek tűnik, de a követelmények teljesítésére ezek az intézkedések szükségesek. Az erőforrások optimálisabb felhasználását megcélozva ki kellett alakítani egy archívumtisztítási rendet mind az első mind a második tárolási és mentési szinten. Ez a gépek kapacitásától, tárolóképességétől és a tárolandó adatok keletkezésétől függő idő, ami után a helyi tárolt adatokat adatarchiváló eszközzel való mentés után törölhetjük a tárakból.

A tárolás első szintje: Az adatgyűjtő és közvetlenül a mérő számítógépek szintje. Az adattárolás formátuma a könnyen mobilizálható, kifejezetten gyorsan kezelhető szöveges fájl, ami a könnyebb adatbázis kapcsolat miatt XML formátumban kerül tárolásra.

A tárolás második szintje: A pufferszerverek szintje. Az előfeldolgozás, kiértékelés után adatbázis szinten tárolható adatok. Ezek könnyen exportálhatóak szükség esetén akár XML formátumba, akár más portábilis formátumba.

A tárolás harmadik szintje: a központi adattárház.

Kérdésként merült fel, hogy a pufferszervereken, illetve a központi tárházban milyen operációs rendszert, adatbázis szerver felépítést használjunk. Erre a választ a helyi sajátosságok adják meg. A meglévő rendszerek integrálása szükséges, így vegyes operációs rendszerek és adatbázis szerverek jelennek meg.



2.2. ábra. Az adatáramlás útja, és a csomópontok szerepe egy adatszolgáltatótól a központi adattárházig.

2.1.2. Következtetések

Mivel a teljes rendszer nagyon összetett, sokféle platform, operációs rendszer, adatbázis rendszer fordulhat elő benne, így a következő következtetésekre lehet jutni:

1. Az igényelt adatbiztonság az adatredundancia segítségével biztosítható.

2. Feladat a különböző operációs rendszerek közötti megfelelő kommunikáció biztosítása.

3. Szoftverfejlesztési feladat volt a különböző adatbázisrendszerek közötti átjárhatóság, az adatok konverziójának megvalósítása.

4. Matematikai és szoftverfejlesztési feladat a megfelelő kódolási eljárások megkeresése és kidolgozása. Valamint ezek titkosítása megfelelő kriptográfiai módszerekkel.

5. Kutatási feladat a központi adattárház finomhangolása és programozása, hogy mind az érkező nagymennyiségű adat befogadása és a várhatóan nagyszámú lekérdezés kiszolgálása is a lehető legoptimálisabb legyen. Annak ellenére, hogy ezek a folyamatok optimalizálása egymás ellen dolgozik. (pl. az indexek száma és felépítése).

2.1.3. Az adatmodell kialakítása

A modell kialakítását megelőzően a következő tevékenységek lettek elvégezve:

1. Felmérés készült helyszíni interjú segítségével a konzorciumban részt vevő, és adatot szolgáltató cégek telephelyein.

2. El lett döntve az adatbázis felépítésének stratégiáját.

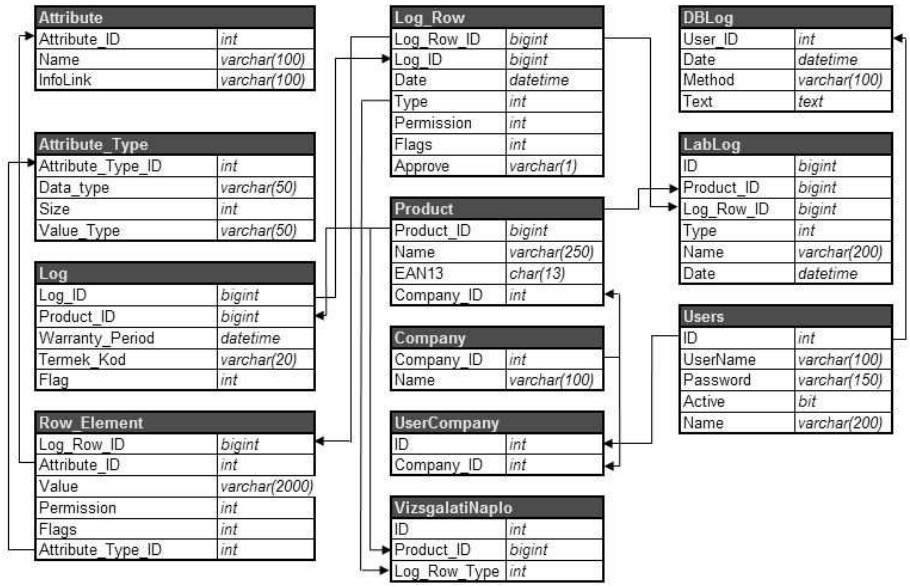
Azaz lehetséges-e egy olyan adatbázis modell kialakítása, mely lehetőleg cég és termék-független módon alkalmazható minden jelenlegi, vagy későbbi konzorciumi tag, minden termékére. Vagy minden tagnak az adatbázison belül külön terület jusson, gyakorlatilag összefüggés és átfedés nélkül. Azaz külön adatbázis partíciókat alakítsunk ki a részükre. Ekkor a statikus adatbázis előnyeit elérjük, viszont a kliens és feldolgozó szoftverek kifejlesztésének költsége, munkaigénye a többszörösére ugrik minden egyes új cég, új terméke esetén.

Az adatok áttekintése után az a dönté született, hogy egy általános adatmodell legyen kialakítva, melyben tetszőleges adat tárolható cégtől függetlenül. Ennek kezelése bonyolultabb, de a későbbi fejlesztéseket jelentősen meggyorsíthatja.

Amikor egy új termék jelenik meg az adatbázisban, amit egy már ismert cég, vagy akár egy új cég állít elő, akkor ennek a terméknek a teljes gyártási folyamatát le kell modellezni. Ehhez kiváló eszköz a munkafolyamat-gráf készítő szoftver, amiről egy későbbi fejezetben lesz szó. A termék előállítási folyamata viszont teljesen, vagy akár csak részben is, de különbözni fog az eddigi termékek gyártási folyamatától. Az élelmiszer termelésben a gyártási folyamatok annyira különböznek egymástól, hogy nem lehetséges egy hagyományos, közös attributumokkal rendelkező adatbázis létrehozása. Ezért merült fel a fent említett két lehetőség.

A második lehetőség azért került elvetésre, mert ha egy új adatbázis partíciót kell integrálnunk a rendszerbe, új adatbázis objektumokkal[3] akkor az nagyban változtatja az adatbázis sémáját is, és az ezt elérő kliensszoftvereket is. Hiszen követni kell a feldolgozó programokkal a változásokat. Mivel a rendszer várhatóan dinamikusan fog bővülni, ezért ezt a munkát a lehető legkevesebbre kell csökkenteni. A választás ezért esett az általános felépítésű, bonyolultabb, de később kevesebb idő alatt, kisebb erőforrások bevonásával bővíthető rendszerre.

[3] Adattáblák, nézetek, tárolt eljárások, triggerek, lekérdezések.



2.3. ábra. Az adatmodell sematikus felépítése.

2.1.4. Tekintsük végig a modell egyes részeit

Két tábla tárolja a konzorciumban részt vevő cégek adatait, és a cégek termékeinek adatait. (*Company*, *Product*) Ez lehetővé teszi, hogy később újabb cégek kapcsolódhassanak be a vizsgálatokba, illetve, hogy egy bent lévő cégnek tetszőleges számú termékére tudjunk adatot tárolni.

A *Product* tábla tárolja a rendszerben szereplő entitásokat. Az entitás rendszerbeli jelentésének meghatározása nehéz. Entitások például: egy termék, alapanyag, dolgozók, műszakok, riasztás, link. Tehát minden dolog, ami egy cég működése folyamán felmerül, vagy az adatbázisban használják, entitás. Vannak olyan entitások, amelyek egyfaja csoportosításnak is felfoghatók, hisz például a dolgozók entitás az összes dolgozót jelenti. De a csoport csak az egyik megközelítése a fogalomnak, mert egy késztermék vagy a link és a riasztás is entitás, ezekből pedig csak egy szerepel a *Product* táblában annak ellenére, hogy a rendszerben számtalan link vagy riasztás is létezhet.

Itt ezek még csak egyfajta általánosítás vagy elvonatkoztatás szintjén léteznek, mert a dolgok egy-egy példányai itt még nem jelennek meg. A *Product_ID* az entitás egyedi azonosítója, a *Name* a neve (legföljebb 250 karakter), az *EAN13* (max. 13 karakter) egyrészt lehet az adott entitás EAN

13 kódja (ha van neki) vagy pedig szerepelhet itt egy azonosító, ami bármilyen formában bővebb információt jelent a dologról.

A *Company_ ID* a *Company* táblával való kapcsolatot valósítja meg, egy céget rendel a dologhoz. Egy cégnek több (akár azonos nevű) entitása is van, de egy entitás csak egy céghez tartozhat (ez a táblák szerkezetéből egyértelmű).

Két tábla tartalmazza a tárolt adat attribútumait, azaz típusát és méretét (*Attribute*, *Attribute_ Type*). Ez szükséges a megfelelő irányú konverziók elvégzéséhez. Három tábla tartalmazza a naplókat, azok adatsorait, és a sorokban tárolt elemi adatokat. (*Log*, *Log_ Row*, *Row_ Element*)

Az adatbázis szerkezete úgy lett kialakítva, hogy minden felmerülő adat egységes szerkezetben tárolható legyen, ez azonban egy meglehetősen absztrakt adatszerkezetet eredményezett. Az adatok a következő táblákban tárolódnak:

Attribute	
Attribute_ID	*int*
Name	*varchar(100)*
InfoLink	*varchar(100)*

Attribute_Type	
Attribute_Type_ID	*int*
Data_type	*varchar(50)*
Size	*int*
Value_Type	*varchar(50)*

Log	
Log_ID	*bigint*
Product_ID	*bigint*
Warranty_Period	*datetime*
Termek_Kod	*varchar(20)*
Flag	*int*

VizsgalatiNaplo	
ID	*int*
Product_ID	*bigint*
Log_Row_Type	*int*

Log_Row	
Log_Row_ID	*bigint*
Log_ID	*bigint*
Date	*datetime*
Type	*int*
Permission	*int*
Flags	*int*
Approve	*varchar(1)*

Product	
Product_ID	*bigint*
Name	*varchar(250)*
EAN13	*char(13)*
Company_ID	*int*

Row_Element	
Log_Row_ID	*bigint*
Attribute_ID	*int*
Value	*varchar(2000)*
Permission	*int*
Flags	*int*
Attribute_Type_ID	*int*

2.4. ábra. A naplók tárolására szolgáló táblák.

Egy dolog egy példánya a *Log* táblában fog megjelenni.

A *Log* táblában kerülnek tárolásra egy cég egy termékének egy műszakhoz kötődő alapadatai. A cég azonosítója, a termék azonosítója, a műszak azonosítója. A műszak fogalom kisebb egységre bontható a termelés felosztá-

sától függően. Egy adott *Log* bejegyzéshez tartozhat tetszőleges sor, melyek egy adatfelvitelt jelentenek.

A *Log_ ID* egy példány azonosítója, a *Product_ ID* a példány entitását azonosítja (a *Product* tábla egyik azonosítója), a *Warranty_ Period* a szavatossági időt tartalmazza, ez késztermék esetén tartalmaz adatot. A *Termek_ Kod* a termékkódot tartalmazza, amely 20 karakter hosszú lehet. Ennek általában csak késztermék esetén van jelentősége, félkész termékek szinte soha nem kapnak termékkódot, ahogyan más entitások sem. A Flag jelzőkód szerepet lát el és a szinkronizálás során van rá szükség. Értékei a következők lehetnek:

- 1: új rekord a puffer szerver számára
- 2: módosított rekord a puffer szerver számára
- 4: törölt rekord
- 8: új rekord a központi szerver számára
- 16: módosított rekord a központi szerver számára
- 32: inaktív rekord

Ahhoz, hogy adatokat lehessen hozzá csatolni, naplóra van szükség. A rendszerben a napló kicsit bővebb értelmet nyer, mint amit általában megszokhattunk. Az egyes élelmiszeripari cégeknél a termelés folyamán az adott lépésre vonatkozó adatokat naplókban tárolják. Ilyen lehet például a raktárba beérkezett alapanyag mennyisége, hőmérséklete, stb. Azonban itt naplónak nevezzük például a dolgozók entitás egy példányának (dolgozó) adatait magába foglaló fogalmat is.

Általánosan tehát a **napló az entitás egy példányának adatait tartalmazza**.

Egy-egy naplót a *Log_ Row* tábla tartalmaz. Egy napló a *Log_ Row* táblában jön létre, de a napló adatait a *Row_ Element* tábla tárolja.
A *Log_ Row_ ID* a napló egyedi azonosítója. A *Log_ ID* mutatja meg, hogy melyik példány naplójáról van szó (kapcsolat a *Log* táblával). A *Date* a napló létrehozásának időpontját adja meg. A *Type* a napló típusát azonosítja, megmutatja, hogy a napló *Row_ Element* elemei milyen típusúak:

- 1: hivatkozás (link) típusú napló
- 2: riasztás (alert) típusú napló

A *Permission* a napló láthatósági szintjét mutatja, a *Flags* pedig, mint ahogyan a *Log* táblánál, jelzőkód szerepet lát el és a szinkronizálás során van rá szükség.

A *Row_Elemet* táblába tároljuk az elemi adatokat. A sorazonosítót és az attribútum azonosítót. Ez alapján akár az adatbeviteli formokhoz, akár az elemzésekhez biztonsággal kinyerhetőek az adatok. A *Log_Row_ID* a naplót azonosítja (kapcsolat a *Log_Row* táblával). Az *Attribute_ID* az adat nevét azonosítja (kapcsolat az *Attribute* táblával). A *Value* az értéket tartalmazza (legföljebb 2000 karakter), a *Permission* a napló láthatósági szintjét mutatja. Az *Attribute_Type_ID* a *Value* mezőben található adat típusát mutatja meg, a *Flags* pedig, mint ahogyan a *Log* táblánál, jelzőkód szerepet lát el és a szinkronizálás során van rá szükség.

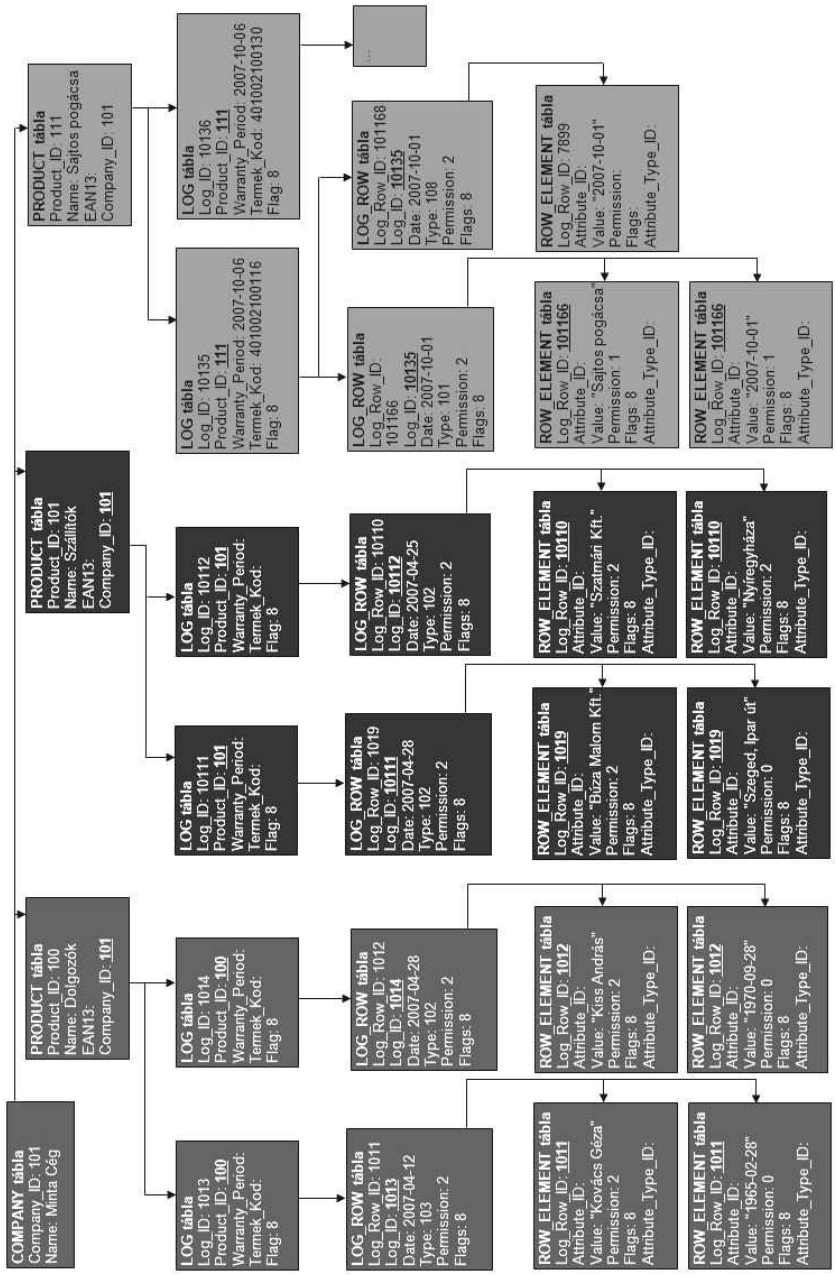
Az *Attribute* tábla a *Row_Element* tábla egy rekordjának nevét tartalmazza. Az *Attribute_ID* az attribútum egyedi azonosítója. A *Name* a nevet tartalmazza, az *InfoLink* pedig lehetőséget biztosít, hogy egy attribútumhoz további információt biztosító url-t helyezzenek el. Az url az internetes lekérdezés során fog fontos szerepet játszani.

Az *Attribute_Type* tábla a *Row_Element* tábla egy rekordjának típusát tartalmazza. Az *Attribute_Type_ID* a típus egyedi azonosítója. A *Data_type* az adattípus neve (legföljebb 50 karakter), a *Size* a maximális mérete, a *Value_Type* pedig a megjelenítés során használt formátumot tartalmazza a következő formában: {0} jelenti a paramétert, a körülötte lévő szöveg pedig egyszerűen átmásolódik. Mérete maximálisan 50 karakter lehet. A százalékok megjelenítésénél például {0} % a *Value_Type* értéke.

Tekintsünk egy egyszerű példát a 2.5 ábra segítségével:

A példában egy Minta Cég nevű cég szerepel, a *Company* tábla tartalmazza az azonosítóját és a nevét. Az ábrán az adatbázis tartalmának egy kis szelete szerepel. A *Product* tábla tartalmaz a cég entitásai közül hármat: Dolgozók, Szállítók, Sajtos pogácsa. Az "előző" táblára való hivatkozást megvalósító azonosítókat aláhúzott félkövér számok mutatják. A Flags illetve Flag mezők értéke mindenhol 8, ami azt jelenti, hogy ezek új rekordok, amelyeket a szinkronizálás során a központi szerverre el kell küldeni. Ebből az is kiderül, hogy a példa egy puffer szerver adatbázisának részletét mutatja.

Látható, hogy a dolgozók entitásból két példány jött létre (1013 és 1014 *Log_ID* azonosítja), mindkét példányhoz egy-egy napló tartozik (*ID*: 1011 és 1012). A *Row_Element* táblában mindkét példányhoz két-két adat is tárolódik. Az ábrán lévő példákban a *Value* mező tartalma alapján következtetni lehet, hogy mi is ez a két adat, de pontos információt majd az *Attribute*



2.5. ábra. Egy egyszerű adatszerkezet az adatbázis struktúrában

és *Attribute_Type* táblák adnak. Ugyanígy a Szállítók entitásból is két példány jött létre. A *Row_Element* tábla itt láthatóan más adatokat tárol, mint a dolgozók esetében, de ezek itt is ugyanúgy szöveges információként kerülnek be az adatbázisba, típusukat majd az *Attribute* és *Attribute_Type* táblák azonosítják.

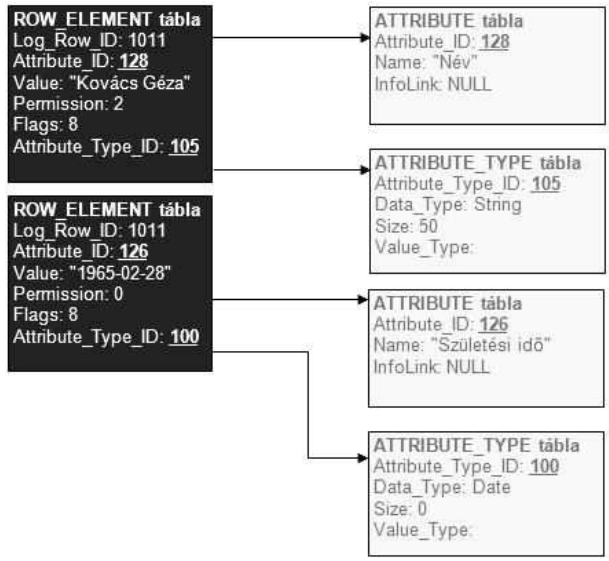
Az adatbázis termék entitásokat is tartalmaz, a példán egy Sajtos pogácsa nevű látható. A termékből két példány is létrejött már, mindkettő késztermék, ennek megfelelően van termékkódjuk is, szavatosság-megőrzési idejük is. Ezekhez ugyanúgy kapcsolódnak a megfelelő naplók. A *Log* táblában nem lehet két olyan rekord, amelynek azonos lenne a *Termek_Kod* mezője. Ezt a kliens program nem is engedi megtenni. Az ábrán a legfelső sorban található (bal oldalon) a cég táblája, aztán a második sorban az entitások, a harmadik sorban a példányok, a negyedik-ötödik-hatodik sorban pedig a példányokhoz tartozó naplók.

2.1.5. Az attribútum-táblák

Látható, hogy egy napló a *Log_Row* táblában jön létre, elemeit pedig a *Row_Element* tábla tartalmazza. Ez a tábla azonban minden értéket szöveges adatként tárol, így közvetlenül nem derül ki az érték típusa, vagyis jelentése. Ezek meghatározására szolgálnak az attribútum táblák.

Az attribútum táblák egyes bejegyzéseihez nem csak egy *Row_Element* táblabeli rekorból (bejegyzésből) történhet hivatkozás, hanem bármennyiből, hiszen ezek az adatok csak típusokat azonosítanak, a bejegyzések pedig néhány fajta típust használnak. A 2.6. ábrán láthatjuk a mezők és az attributumok kapcsolatát.

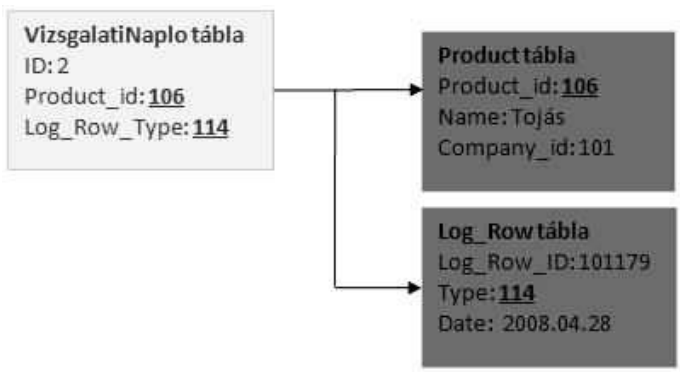
A "Kovács Géza"-t tartalmazó rekord a 128-as azonosítóval jelölt attribútumra hivatkozik, ez pedig megmutatja, hogy a rekord neve: "Név". A 105-ös attribútum típus azonosító pedig megmutatja, hogy a rekord sztring, vagyis karakter típusú adatot tárol, melynek maximális hossza 50 karakter lehet. A NULL az *InfoLink* mezőben azt mutatja, hogy ehhez a bejegyzéshez nem tartozik információs link. A *Value_Type* mezőbe pedig kerülhetne a "neve: 0" szöveg, ami megmutatná, hogy a rekord értékét a következő módon kell megjeleníteni: "neve: Kovács Géza". Megjegyzendő, hogy a Date típus dátumot jelöl, hossza pedig azért 0, mert a dátum típus fix hosszúságú, ezért szükségtelen a hosszt külön specifikálni.



2.6. ábra. A mezők és az attribútumok kapcsolata

2.1.6. A vizsgálati naplók

A vizsgálati naplónak csak az internetes lekérdezések során van fontos jelentősége. A vizsgálati naplók tárolásához egy új tábla szükséges.



2.7. ábra. A vizsgálati naplók tárolása

A *VizsgalatiNaplo* tábla egy összekapcsolást végez a *Product* és a *Log_Row* tábla adatai között. A tábla adatai új cég létrehozásakor automatikusan

létrejönnek, pontosan annak a folyamatnak a során, amikor a gráfot[4] a központi szerverre feltöltik. Minden naplóban minden gyökér elemnek van egy azonosítója. A feltöltés során ezekkel az értékekke jön létre a *VizsgalatiNaplo* tábla.

A webes lekérdezés során a rendszer megvizsgálja, hogy az aktuális entitáshoz milyen vizsgálati naplók tartoznak (az entitást a *Product_Id* azonosítja). A kapott naplók közül megvizsgálja, hogy melyek vannak ehhez a példányhoz is fölvéve. Ezeket megjeleníti. A többi típushoz, amelyekből még nem szerepel egy fölvett sem a példánynál, megkeresi az utolsó dátumút a már korábban felvett példányok közül. Természetesen, ha az adott típusú naplóból még nincs egy sem, akkor nem jelenít meg semmit.

2.1.7. A hivatkozás (link)

A link lehetőséget biztosít arra, hogy egy naplóból egy másik naplóra lehessen hivatkozni. A link is egy entitásból "származik", egyetlen példánya van, ami az adatbázis felépítésekor létrejött. Ehhez a példányhoz lehet naplóként felvenni az újabb linket, amikor szükség van rá. A linknek nincs hagyományos értelemben vett példányosító naplója; minden naplója egyforma. A linkeket nem lehet kézzel létrehozni, a rendszer ezeket automatikusan generálja egy napló létrehozásakor. A hivatkozás így az alapvető követési mechanizmust valósítja meg. A link entitás az adatbázisban 1-es $Product\_ID$-vel szerepel és az 1-es azonosítóval rendelkező SYSTEM céghez tartozik, viszont minden céghez létrejön egy példány belőle. Mivel a cég azonosítója lesz a Log_ID első három számjegye, ebből lehet tudni szükség esetén, hogy az adott link példány melyik céghez is tartozik, annak ellenére, hogy entitása a SYSTEM céghez.

Linket a napló bejegyzésekben (*Row_Element* tábla) az 1-es attribútum típus (*Attribute_Type_ID*) jelzi. Ekkor a tábla $Value$ mezője a Log_Row tábla egy rekordjának Log_Row_ID értékére hivatkozik. Ez a rekord egy link naplója, amely a következő bejegyzéssekkel rendelkezik (három rekord a *Row_Element* táblában):

- 4: példány hivatkozás: egyszerűen a $Value$ mező értéke a Log tábla egy rekordjának azonosítóját tartalmazza.
- 5: kezdő dátum
- 6: záró dátum

[4] munkafolyamat gráf

- 7: példány azonosító hivatkozás: megmutatja, hogy a link által hivatkozott példány példányosító naplójában mely attribútummal jelölt *Row_Element* rekord tartalmazza a példány azonosítóját

- 8: példány név hivatkozás: megmutatja, hogy a link által hivatkozott példány példányosító naplójában mely attribútummal jelölt *Row_Element* rekord tartalmazza a példány nevét.

A megadott értékek a rekord $Attribute\_Type\_ID$ mezőjében jelzik a típust. A bejegyzés értéke a *Log* tábla egy példányára mutat, vagyis egy rekord Log_ID mezőjének értékét tartalmazza.

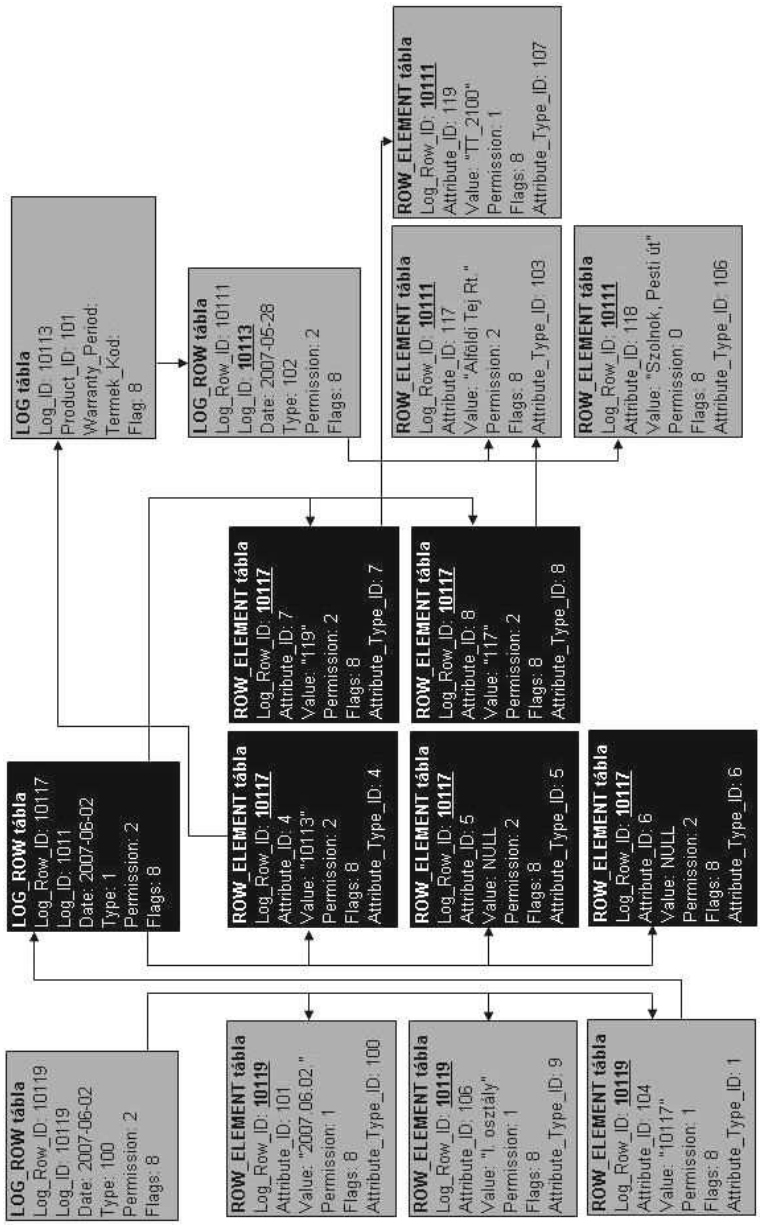
A könnyebb érthetőség kedvéért a 2.8 ábrán mindez megtekinthető. A példában a Minta Cég sajt entitás egy példányának napló bejegyzéseiből indulunk ki. Az egyszerűség kedvéért a sajt $Product \rightarrow Log$ hivatkozásai nem szerepelnek az 2.8 ábrán, de természetesen ezek is léteznek. Így a link $Product \rightarrow Log$ hivatkozása sem szerepel, de a *Log* tábla megfelelő (1011-es azonosítójú) adatából látható lenne, hogy a $Product\_ID$-je 1. A naplónak (Log_Row_ID: 10119) itt három bejegyzése lett kiemelve. A lényeges most a harmadik bejegyzés; az $Attribute\_Type\_ID$ 1 értéket tartalmaz, tehát ez egy link. Az értéke 10117, tehát ez mutatja meg, hogy a link példány 10117-el jelölt naplójára hivatkozik. Az 2.8 ábrából jól látható, hogy a link nem közvetlen, hanem közvetett (indirekt) hivatkozást jelent, vagyis az attribútum egy hivatkozás típusú naplóra mutat, ami megmutatja, hogy mi a hivatkozás eredeti célja. Erre azért van szükség, mert így a link is egy összetett struktúra lehet, amihez tetszőleges adatokat lehet kapcsolni.

A link szerkezete a következő:

- a dátuma fontos

- bejegyzései:

  - A 4-es attribútum id-vel jelölt bejegyzés értéke mutatja, hogy a 10113 azonosítójú példányra mutat a hivatkozás (a *Log* táblában). Ez egy példányosító napló lesz, de ez csak a 7-es és 8-as id-vel jelölt bejegyzések nem üres mivoltából derül ki.
  - Az 5-ös és 6-os id-vel jelölt bejegyzések NULL sztringje az ábrán azt jelöli, hogy azoknak nincs érték adva, vagyis nem használtak.



2.8. ábra. A link működése

– A 7-es és 8-as id-vel jelölt bejegyzések pedig megmondják, hogy a 4-es attribútum id-vel jelölt bejegyzés által muatott példány példányosító naplójában mely *Attribute_ID*-vel jelölt bejegyzések jelentik a példány nevét (itt Alföldi Tej Rt.) és azonosítóját (itt *TT_2100*).

Az 5-ös bejegyzés a kezdő, a 6-os pedig a záró dátumot jelölné. Ezeknek többféle kombinációja többféle esetet jelöl:

- Mindkét dátum null: a link egy listára, vagyis törzsadatra mutat.
- Csak a kezdő dátum van megadva: a link által hivatkozott példánynak csak azokra a naplóira van szükség, amelyek dátuma nem korábbi a link kezdő dátumánál (a bejegyzés Value mezője).
- Csak a záró dátum van megadva: a link által hivatkozott példánynak csak azokra a naplóira van szükség, amelyek dátuma nem későbbi a link záró dátumánál (a bejegyzés Value mezője).
- Mindkét dátum meg van adva: csak a két dátum közé eső naplók elérésére van szükség.

Mindkét dátum, illetve csak a kezdő dátum megadása esetén a link által hivatkozott példány példányosító naplójának elérésére is szükség van.

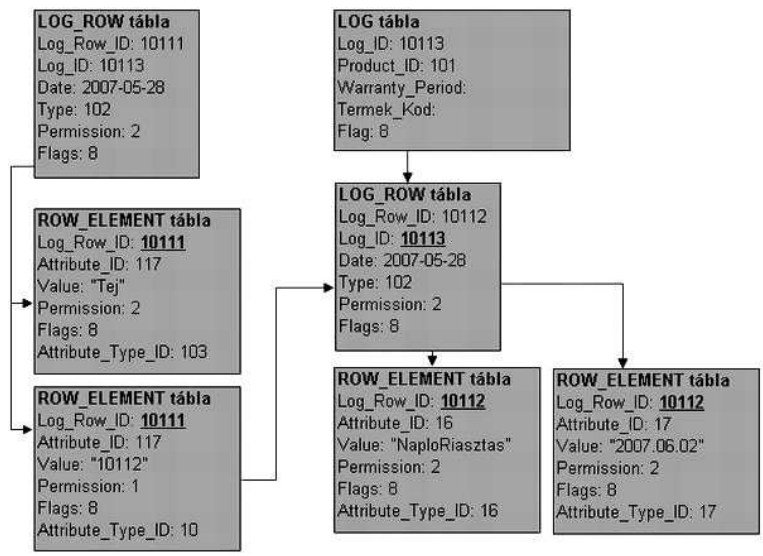
2.1.8. A riasztások (alert)

A riasztások egy bizonyos esemény bekövetkeztét vagy egy naplóbeli adat bizonyos rendellenes állapotát jelzi. A riasztás előírások létrehozása egyrészt automatikusan történik egy napló felvitelekor, de a riasztás feltételeit a napló gráf szerkesztőbeli szerkesztésekor természetesen be kell állítani. Riasztás előírást másrészt egy felhasználó is létrehozhat a kliens programban, ekkor ő adja meg a riasztás paramétereit.

A riasztás is egy entitásból "származik", egyetlen példánya van, ami az adatbázis felépítésekor létrejön. Ehhez a példányhoz lehet naplóként felvenni az újabb riasztást, amikor szükség van rá. A riasztásnak nincs hagyományos értelemben vett példányosító naplója; minden naplója egyforma. A riasztást nem lehet kézzel létrehozni, a rendszer ezeket automatikusan generálja egy napló létrehozásakor. A riasztás entitás az adatbázisban 2-es *Product_ID*-vel szerepel és az 1-es azonosítóval rendelkező SYSTEM céghez tartozik, viszont minden céghez létrejön egy példány belőle. Mivel a cég azonosítója lesz a *Log_ID* első három számjegye, ebből lehet tudni szükség esetén,

hogy az adott riasztás példány melyik céghez is tartozik, annak ellenére, hogy entitása a SYSTEM cégé.

Tehát cégenként egy darab riasztás példány van. Minden újabb riasztás ennek a példánynak a naplója. A riasztást a napló bejegyzésekben (*Row_Element* tábla) a 10-es attribútum típus (*Attribute_Type_ID*) jelzi. Ha egy attribútumhoz riasztás is tartozik, akkor a naplóhoz egy olyan *Row_Element* bejegyzés is létrejön, amiben az *Attribute_Type_ID* értéke 10 lesz, itt a *Value* értéke fog arra a *Log_Row* rekordra mutatni, amelyik a cég riasztás példányának egy naplóját tartalmazza, az *Attribute_ID* pedig annak a naplóbejegyzésnek az *Attribute_ID* értékét fogja tartalmazni, amire a riasztás vonatkozik. Így egy naplón belül bármely bejegyzésre és akár többre is létrehozható riasztás.



2.9. ábra. A riasztások szerkezete

A riasztás szerkezetét mutatja be a 2.9 ábra.

A riasztás napló bejegyzései szabványosak, konstans *Attribute_Type_ID* azonosítóval rendelkeznek, melyek a következő dolgokat határozzák meg:

- 11: a riasztás szövege; ez üzenet a jóváhagyónak. A szöveg a riasztás során automatikusan létrejön. Ha a kliensben a felhasználó kézzel hozza létre a riasztást, ő is megadhat szöveget.

- 12: a jóváhagyás szövege: a jóváhagyó felhasználó is megadhat szöveget

a riasztáshoz, ez kerül ide

- 13: a jóváhagyás ideje, ha már megtörtént
- 14: a jóváhagyó felhasználó azonosítója (*User_ID*)
- 15: a riasztás státusz, amely két értéket vehet fel (minden más érték hibás):
  - 0: nincs jóváhagyva
  - 1: jóvá van hagyva
- 16: a riasztás típusa:
  - 0 vagy NaploRiasztas: amikor egy napló rendellenes bejegyzésének hatására történik meg a riasztás
  - 1 vagy RendszerRiasztas: amikor a rendszer rendellenes működése következtében jön létre riasztás
  - 2 vagy LabornaploErkezett: amikor a labor egy naplót vett fel a céghez. Ez a riasztás azért szükséges, hogy a cég megfelelő alkalmazottja értesüljön erről az eseményről
- 17: a riasztás dátuma
- 18: a jóváhagyó felhasználó felhasználói neve

2.1.9. A jelzőkódok (flagek) az adatbázisban

Jelzőkódokat a *Log*, *Log_Row* és *Row_Element* táblákban használ a rendszer. Ezek egyszerű egész típusú mezőkben tárolt értékek. Használatuk bitenként történik, mert egy rekordhoz több kódot is lehet rendelni egyszerre. Ezért az értékek is kettő hatványai:

- 1: új rekord a puffer szerver számára (a kliens állítja elő a puffer szerver számára)
- 2: módosított rekord a puffer szerver számára (a kliens állítja elő a puffer szerver számára)
- 4: törölt rekord (a kliens állítja elő a puffer szerver számára): ezt a rendszer minden komponense változatlanul meghagyja, soha nem törlődik. Ez a bit csak akkor változhat 0-ról 1-re, amikor az adat törlődik. A rendszer komponensei soha nem adnak vissza olyan adatot, aminek ez a bitje aktív.

- 8: új rekord a központi szerver számára
- 16: módosított rekord a központi szerver számára
- 32: inaktív rekord

Az 1 és 2 biteket a puffer szerver értelmezi, de az adatbázisba csak a 4, 8, 16 bitek kerülnek be. Szinkronizáláskor a puffer szerver a 8 és 16 flagek alapján összegyűjti a szükséges adatokat, szinkronizáláskor a központi szerveren az adat módosításra kerül, ezután a puffer szerveren a 8 és 16 jelzőbitek törlődnek. Ha egy rekordot mentenek a puffer szerverre 8-as flaggel, aztán megkapja a 16-os flaget is, a 8-as nem törlődik, mert szinkronizáláskor az a központi szerveren új adatnak számít majd.

2.1.10. A lekérdezések naplózása

A központi szerver egyik fő feladata az internet felől érkező lekérdezések kiszolgálása. A lekérdezések számos forrásból érkezhetnek és nagyon sok mindent kérhetnek az adatbázisból. Ezen lekérdezések megfigyelésével később számos üzleti és technikai jellegű kérdésre válaszolni lehet a rendszer működése közben, ezért az ezekre jellemző információt is tárolja az adatbázis. Ez a lekérdezések naplózása. Erre a feladatra az adatbázisban külön táblák szolgálnak.

A *session_data* tábla tárolja a lekérdezést végző eszköz adatait. Ezt mutatja a 2.10 ábra. A *session_id* 50 karakteres azonosító, amely a lekérdezést mint szerveroldali folyamatot azonosít (a session azonosítója). A *date* egy dátum típusú érték, amely másodpercre pontosan megmutatja, hogy mikor történt a lekérdezés. Az *uap* egy legföljebb 200 karakter hosszú érték, amely wap-os lekérdezés esetén a lekérdezést végző telefonkészüléket azonosítja. Ez tulajdonképpen egy internetes url, amit a telefon a lekérdezés során a szervernek küld. A *phone_name* az uap információkból megállapítja a telefon gyártóját és a telefon típusát. Ez legfeljebb 50 karakter hosszú lehet. Az *user_agent* egy maximum 50 karakter hosszú érték, amely a böngésző nevét és verzióját tartalmazza.

A *query_data* tábla tárolja a lekérdezés adatait. A 2.11 ábrán látható a tábla szerkezete. A *session_id* 50 karakteres azonosító, amely a lekérdezést mint szerveroldali folyamatot azonosít. A *date* egy dátum típusú érték, amely másodpercre pontosan megmutatja, hogy mikor történt a lekérdezés. A *termekkod* a lekérdezett termékkódot jelenti, legföljebb 20 karakter hosszú lehet. A *hiba* mezőben a hiba szerepel szövegszerű formában, amennyiben a lekérdezés közben valami hiba történt. A *logid* és a *companyid* a lekérdezett

session_data	
session_id	*varchar(50)*
date	*datetime*
uap	*varchar(200)*
phone_name	*varchar(50)*
user_agent	*varchar(50)*

2.10. ábra. A session tábla szerkezete

napló esetleg a cég azonosítóit tartalmazza. A *get_info* mező azt mutatja, hogy a felhasználó kért-e információt egy dologról. A *sitemap* mező azt mutatja, hogy a felhasználó a felső navigációs sávra kattintva végezte-e el az adott lekérdezést (ebben az esetben tehát egy már megtekintett szintre lépett vissza).

query_data	
session_id	*varchar(50)*
date	*datetime*
termekkod	*varchar(20)*
hiba	*varchar(200)*
logid	*bigint*
companyid	*int*
get_info	*bit*
sitemap	*bit*

2.11. ábra. A query tábla szerkezete

A két tábla közötti kapcsolat a *session_id* és a date mezőn keresztül történik; azonos *session_id* és date azonos lekérdezést jelent.

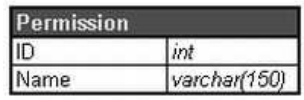
2.1.11. A puffer szerverek szintje

A puffer szerver leginkább egy konkrét cég adatait hivatott tárolni (például a felhasználók). Emellett tárolja a szerverre kerülő napló-adatokat és esemény-naplózási információkat is a felhasználók tevékenységéről.

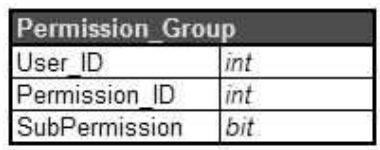
2.1.12. A felhasználók és jogosultságok tárolása

Mind a felhasználók, mind pedig a jogosultságok tárolása a pufferszerveren történik, mert mindig egy adott céghez köthetőek és annak a belső, privát adatait tartalmazzák.

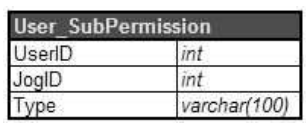
A *Permission* tábla tárolja a rendszerben létező összes jogosultságot azonosító és név párosként. Az *ID* mező elsődleges kulcsként szolgál a táblában értéke egész szám. Nincs kitüntetett érték, de egy érték mindig ugyanahhoz a jogosultsághoz kötődik (értelemszerű). A *Name* mező tartalmazza a jogot megnevezve. Értéke maximum 150 karakter lehet.



2.12. ábra. A *Permission* tábla szerkezete



2.13. ábra. A *Permission_Group* tábla szerkezete



2.14. ábra. A *User_SubPermission* tábla szerkezete

A *Permission_Group* tábla köti össze a felhasználókat a jogosultságokkal. Ezt mutatja a 2.13. ábra. Három mezője van. Az *User_ID* a *Users* táblából azonosít ID alapján egy felhasználót, így értékére is annak megszorításai érvényesek. A *Permission_ID* a *Permission* táblából azonosít egy megadott jogot, értékére annak megszorításai érvényesek. Ha egy felhasználóhoz több jog is rendelve van, akkor az több rekordként szerepel ebben a táblában. Amikor törölnek egy felhasználót, a *Permission_Group* táblából is törlődnek a felhasználóhoz tartozó jogok. A *User_SubPermission* tábla azonosítja az aljogokat. Az *UserID* egy felhasználót jelöl ki a *Users* táblából. A *JogID* egy rekordot jelöl ki a *Permission* táblából (hivatkozás a

Permission_ID-re). A *Type* egy szöveges mező, amiben különböző módokon tárolódnak az adott aljogosultságok.

Users	
ID	*int*
UserName	*varchar(100)*
Password	*varchar(150)*
Active	*bit*
Name	*varchar(200)*

2.15. ábra. A *User* tábla szerkezete

A *Users* tábla tárolja az adott cégnél a rendszerben létező összes felhasználót (még a törölteket is). Az *ID* mező elsődleges kulcsként szolgál, értéke tetszőleges egész szám. A *UserName* mezőben tárolódik a felhasználói név, ez maximum 100 karakter hosszú lehet. Ebben a mezőben is egyedi értékeknek kell szerepelni. A *Password* mezőben tárolódik a felhasználó kódja MD5 kódolással rejtjelezve legföljebb 150 karakter hosszan. A *Name* mező tartalmazza a felhasználó nevét, ez 200 karakter hosszú lehet. Az *Active* egy jelzőbit, azt mutatja, hogy az adott felhasználót törölték-e már a rendszerből (logikai törlés, fizikai törlés nincs).

2.1.13. Szinkronizálás a központi szerverrel

A puffer szerverre véletlenszerűen érkeznek a kliensektől az adatok. Ezek között vannak nyilvánosak, védettek és saját minősítésűek. Eközben a központi szerver adatai is módosulnak (a többi puffer szerverről vagy a laborból beérkezett adat, vagy éppen új cég létrehozása miatt). Ezért szükség van arra, hogy a puffer szerver és a központi szerver adatbázisát összehangoljuk. Ezt az összehangolást nevezik szinkronizációnak. A szinkronizáció során a puffer szerverről az új adatok átmásolódnak a központi szerverre, a központtól pedig a cégre vonatkozó új adatok átmásolódnak a puffer szerverre. A szinkronizálás időpontját a kliens programból lehet beállítani és a megfelelő időpontban a szinkronizálás automatikusan lefut.

2.1.14. Összefoglalva

Sikerült egy olyan adatbázis szerkezetet létrehozni, mely általánosan használható, általános felépítésű és sémamódosítás nélkül bővíthető rendszer, ezzel bizonyítást nyert **az 1. tézis.**

2.2. Hatékonysági vizsgálatok

Napjaink kutatási területei között fontos helyet foglal el az adatbázis rendszerek vizsgálata, ezek hatékonyságának mérése. [3] A nagy adatforgalmú rendszereknél az adatok felvitele, azok beszúrása jelentős és kifejezetten erőforrás-igényes művelet.

A benchmark mérésnél szem előtt kell tartani mind a szerver, mind a kliensoldali szoftverek lehetőségeit. [6] A mérési eredményeket befolyásoló tényezők, mint a Dataset mérete, az SQL parancsok összetettsége, és a hardver/szoftver környezet rögzítése az első feladat.

A kliens program a Microsoft Visual .Net rendszerében készült, C# nyelven. Az ADO.NET technológia használata hatékony eszköz az adatbázisok eléréséhez. Az optimális teljesítmény eléréséhez felhasználtuk a Microsoft ajánlásokat és kutatások eredményeit. [4, 5]

Először megvizsgáltam, hogy mely adatbázis szervert kell választani, hogy a kialakult feladatot a leghatékonyabban lehessen elvégezni. Ez jelenti egyrészt a műveletek sebességének növelését. Másrészt szem előtt kellett tartani a gazdasági szempontokat is a választásnál.

2.2.1. Választás az adatbázis szerverek között

Egy plusz vélemény

A nyílt forrású adatbázis-kezelő rendszerek használata akár több mint 50 százalékos megtakarítást is jelenthet a vállalatok számára ahhoz képest, mint ha valamelyik kereskedelmi adatkezelő versenytársukat használnák. Legalábbis erre a megállapításra jut a Forrester Research piackutató intézet, amely a két típusba tartozó megoldások előnyeit és hátrányait veti össze egy mással.

A Forrester tanulmánya bár elismeri, hogy a kereskedelmi adatbázis kezelők általában magasabb szintű funkciókat és több szolgáltatást kínálnak, mint nyílt forrású társaik, ezt a többletet azonban szerinte a legtöbb helyen valójában nem használják ki. Ugyanakkor a tanulmány szerzői szerint továbbra is vannak olyan területek, ahol a nyílt forráskódú megoldások nem érnek fel a kereskedelmi termékekkel, mert teljesítményük, biztonságuk és rendelkezésre állásuk elmarad azoké mögött. A nyílt forrású adatbázis kezelők tömegbázisát inkább a kevésbé kritikus, és általában viszonylag csekély adattal dolgozó megoldások képezik. A tanulmány végkövetkeztetése az, hogy a nyílt forrású rendszerek elvitathatatlan előnyét továbbra is áruk képezi, és a legtöbb helyen egyértelműen ez - nem pedig technológiai jellemzőik

- miatt döntenek használatuk mellett.

Az SQL szerverünk kiválasztását egy teszt előzte meg, melynek során különböző SQL szervereket használva, szimulációs program segítségével több millió rekorddal lett feltöltve az adatbázis. Vizsgáltam az adatfeltöltés hatékonyságát, és a lekérdezések futási idejét.

A vizsgált adatbázisok:

1. Oracle 10g
2. MSSSQL 2005
3. PostgreSQL 8.2

A vizsgált SQL szerverek Windows 2003 szerver operációs rendszerre lettek telepítve.

Elkészült 3 program, melyből az első az alapadatokat tölti fel, majd a második az MSSQL 2005 szerver, illetve az Oracle alatti adatbázisok *log*, *log_row*, *Row_element* tábláit tölti adattal. A harmadik a PostgreSQL alatti adatbázist kezeli.

A programok VS 2005 eszközzel, C# nyelven készültek, ahogy a projekt szoftverei készülnek majd.

A programok segítségével az adatbázisokba közel 3 millió rekord került. Ezek megoszlása a fenti rendszer szerinti. Már a feltöltés közben vizsgáltuk az időszükségletet.

Már a 19. oldalon a 2.3. számú ábrán bemutattam az adatbázis sémáját. Ez a felépítés lett használva a vizsgálatok elvégzéséhez, hiszen ennek a rendszernek kellett működnie a leghatásosabb módon.

A vizsgálat két területre terjedt ki.

Első eset az adatfelvitel hatékonysága a különböző SQL szerverek esetén. Várhatóan az információs rendszerbe időnként nagy mennyiségű adat kerül. Ennek sem szabad a rendszer túlságosan lassítania.

Másodsorban a lekérdezések lettek vizsgálva. Alapvető fontosságú kérdés, mert az adatfelhasználók kiszolgálása akár weben, akár wapon keresztül nagy sebességgel kell hogy történjen.

2.2.2. Az adatfeltöltés vizsgálata

Elkészült egy tesztszoftver, mely felelős az adatok feltöltéséért a három SQL szerver alá. A feltöltésben használt adatok lehetnek véletlenszerűek, de a megadott listákból kiválogatva, a minta alapján, és a cégek között az

adott súlyokat alkalmazva. Ezek a súlyok a cégekkel folytatott interjúkon kialakult várható rekordszám arányokat tükrözik. Hiszen nem minden cég, minden termékpálya azonos intenzitással termeli az adatokat.

Cégek	*Súlyok*	*Valószínűségek*
Halak	3	0,075
Húsverő	6	0,150
Tésztanyújtó	6	0,150
Borkimérő	1	0,025
Kekszsütögető	5	0,125
Gombaszedő	7	0,175
Kutató	12	0,300

2.1. táblázat. Cégek adatszolgáltatásának valószínűségei

A valószínűség annak a valószínűsége, hogy az érekző adat az adott cégtől jön.

Termékek:

Cégek	*Termékek*
Halak	Halkonzerv
Húsverő	Paprikás kolbász
Tésztanyújtó	Réteslap
Borkimérő	Fehér édes
Kekszsütögető	Kókuszos különleges
Gombaszedő	Csiperke
Kutató	Saját keksz

2.2. táblázat. Cégek termékei

Ezek alapján feltölthető az adatbázis *Company* és *Product* táblája. Az *Attribute* és *Atribute_Type* táblák szükségszerűen töltendőek a rögzítendő adatok alapján típusai alapján.

Nézzünk erre egy példát:

Mary Smith 2007.december 15-én a nap második műszakában a 10. termelő sátorban 3 kg gombát szedett. A **Gombaszedő** cég **Csiperke** termékének a nyomokövetéséhez 2007.12.15-én a 2. műszakban készített napló

tárolása:

Dátum	*Dolgozó*	*Műszak*	*Sátor*	*Termék*	*kg*
2007.12.15	Mary Smith	2	10	Csiperke	3

2.3. táblázat. Példa adatai

Hogyan jelenítjük meg ezeket az adatokat az adatbázisunkban?

Tábla	*Adatok*
Company	Gombaszedő (id: 3) ez megvan, csak hivatkozni kell.
Product	Csiperke (id 17), és a Gombaszedő cég id-je (3, idegen kulcs: FK)

2.4. táblázat. Példa adatai az adatbázisban

Log-tábla:
Log_Id: generálandó Product_Id: idegen kulcs (FK), keresni kell (17) Work_period: kapott adat (2)

2.5. táblázat. Példa adatai a Log-táblában

Log_row-tábla:
Log_Row_Id : generálandó (100103) Log_Id: FK keressük (200100) Datum: kapott adat (2007.12.15)

2.6. táblázat. Példa adatai a Log_Row-táblában

A terhelést a *Log*, *Log_Row* és *Row_Element* táblák kapják. Ezek közül is elsősorban a *Row_Elemet* tábla. Az *Attribute* és az *Attribute_Type* táblák pár tucat esetleg száz rekordnál telítődnek. Így a lekérdezéseknél ekkora nagyságrendű rekordból kell választani. A rekordok feltöltéséhez a megadott cégek közül választottunk az adott valószínűséggel, majd a hozzá tartozó 1 terméket használjuk.

Row_Element-tábla:
Row_Element_Id: generáljuk Log_Row_Id: FK, keresni kell Attribute_Id: FK, keresni kell Value: string típus a kapott cella értéke. pl.: Mary Smith

2.7. táblázat. Példa adatai a Row_Element-táblában

Row_Element_Id	*Log_Row_Id*	*Attribute_Id*	*Value*
12345	100103	15	Mary Smith
12346	100103	19	10
12347	100103	3	3

2.8. táblázat. Azonosítók és értékek

Attribute-tábla:
Attribute_Id: generálódik Attribute_Type_Id: FK, keresni kell Name: string, az attributum neve, leírása. Pl.: name vagy sátor

2.9. táblázat. Példa adatai a Attribute-táblában

Attribute_Type-tábla:
Attribute_Type_Id: generálandó Name string, az attributum adattípusa Size : adattípus mérete

2.10. táblázat. Példa adatai a Attribute_Type-táblában

2.2.3. A mérések eredményei

A mérések eredményeit a 2.11 táblázat tartalmazza. Az első oszlop a rekordok számát, a második az MSSQL , majd az Oracle végül a PostgreSQL szerveren mért adatokat mutatja. A táblázat minden egyes adata egy átlagérték, melyet 10-30 mérési eredményből számoltam. Az adatokat szemlélteti a 2.16 ábra.

Jól látható, hogy a rekordszám növekedésével lineáris növekedést mutat az inserthez szükséges idő.

A lineáris regresszióval kapott egyenletek:

MSSQL:

$$y = 0,6221x - 1,0057$$

Oracle:

$$y = 0,5959x - 0,9633$$

PostgreSQL:

$$y = 1,2644x - 2,044$$

Látható, hogy

$$\underline{m_{MSSQL} = 0,6221 \simeq m_{Oracle} = 0,5959}$$

Látható, hogy az eltérés 5% alatt marad:

$$\frac{m_{MSSQL} - m_{Oracle}}{m_{MSSQL}} = \frac{0,0262}{0,6221} = 0,042$$

Míg

$$m_{PostgreSQL} = 1,2644$$

Ezeket összevetve:

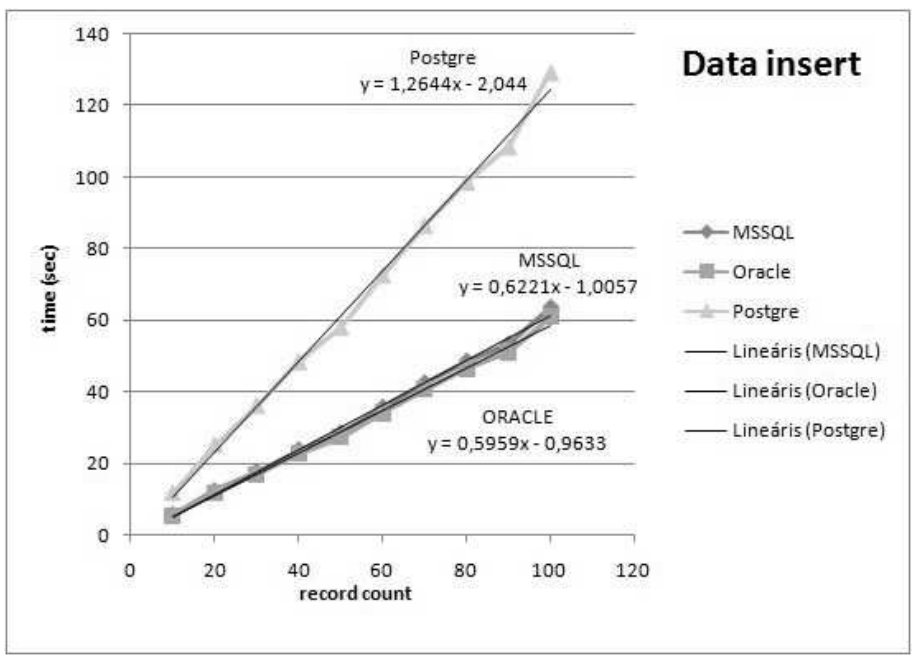
$$M = \frac{m_{PostgreSQL}}{m_{Oracle}} = \frac{1,2644}{0,5959} = 2,1218$$

Az MSSQL és az Oracle egyenese szinte illeszkedik egymásra. Míg a PostgreSQL egyenese sokkal meredekebb, azaz ebben az esetben az adatok beszúrása kevésbé hatékony.

Így ezek alapján az Oracle és az MSSQL szervert vizsgáltam tovább a lekérdezésekkel.

RecCount (ezer db)	MSSQL	Oracle	PostgreSQL
10	5,70626	5,46576	11,59809
20	12,38332	11,86142	25,16935
30	17,76040	17,01188	36,09837
40	23,79584	22,79295	48,36553
50	28,53130	27,32883	57,99045
60	35,70000	34,19540	72,56098
70	42,54800	40,75479	86,47967
80	48,58960	46,54176	98,75935
90	53,42400	51,17241	108,58540
100	63,66440	60,98123	129,39920

2.11. táblázat. Adatbevitel mérések eredményei



2.16. ábra. Adatbevitel mérése

2.2.4. A lekérdezések vizsgálata

A mérés második szakasza a lekérdezések hatékonyságát vizsgálta a feltöltött adatok segítségével. Az MSSQL szerver optimalizálója által javasolt indexeket használtam az egyes lekérdezésekhez.

Adott napló adott sorának visszaállítása.

```
SELECT  Company.Name, Product.Name AS productName,
        [Log].Work_Period, [Log].Log_ID, Log_Row.Datum,
        Log_Row.Log_Row_ID, Row_Element.Value,
        Attribute.Name AS AttName,
        Attribute_Type.Name AS AttTypeName
FROM Product INNER JOIN Company
    ON Product.Company_ID = Company.Company_ID
    INNER JOIN [Log]
    ON Product.Product_ID = [Log].Log_ID
    INNER JOIN Log_Row
    ON [Log].Log_ID = Log_Row.Log_ID
    INNER JOIN Row_Element
    ON Log_Row.Log_Row_ID = Row_Element.Log_Row_ID
    INNER JOIN Attribute
    ON Row_Element.Attribute_ID = Attribute.Attribute_ID
    INNER JOIN Attribute_Type
    ON Attribute.Attribute_Type_ID =
    Attribute_Type.Attribute_Type_ID

WHERE (Log_row.Log_Row_ID = 2)
```

A használt index:

```
use [TestDatabase]
go

CREATE NONCLUSTERED INDEX
[_dta_index_Row_Element_10_21575115__K2_K3_4]
ON [dbo].[Row_Element]
( [Log_Row_ID] ASC,[Attribute_ID] ASC)
INCLUDE ( [Value])
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF)
ON [PRIMARY]
```

```
go

CREATE NONCLUSTERED INDEX
[_dta_index_Log_Row_10_2137058649__K1_K2_3]
ON [dbo].[Log_Row]
( [Log_Row_ID] ASC,[Log_ID] ASC)
INCLUDE ( [Datum]) WITH (SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, IGNORE_DUP_KEY = OFF, ONLINE = OFF)
ON [PRIMARY]
go

CREATE STATISTICS [_dta_stat_2137058649_2_1]
ON [dbo].[Log_Row]([Log_ID], [Log_Row_ID])
go

CREATE NONCLUSTERED INDEX
[_dta_index_Log_10_5575058__K1_3]
ON [dbo].[Log]
( [Log_ID] ASC)
INCLUDE ( [Work_Period])
WITH (SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, IGNORE_DUP_KEY = OFF,
ONLINE = OFF) ON [PRIMARY]
Go
```

b. Adott cég, adott időintervallumban keletkezett naplói, és az ezekhez tartozó naplósorok.

```
SELECT  Company.Name, Product.Name AS Prod,
        [Log].Log_ID, [Log].Work_Period, Log_Row.Datum
FROM    Company INNER JOIN
        Product ON Company.Company_ID = Product.Company_ID
        INNER JOIN
        [Log] ON Product.Product_ID = [Log].Product_ID
        INNER JOIN
        Log_Row ON [Log].Log_ID = Log_Row.Log_ID
WHERE   (Company.Company_id = 3)
        and
        (Log_Row.Datum between '2007-01-01' AND '2007-01-31')
```

A használt index:

```
use [TestDatabase]
go

CREATE CLUSTERED INDEX
[_dta_index_Log_c_10_5575058__K1_K2]
ON [dbo].[Log]
( [Log_ID] ASC,[Product_ID] ASC)
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF) ON [PRIMARY]
go

CREATE STATISTICS
[_dta_stat_5575058_1_2]
ON [dbo].[Log]([Log_ID], [Product_ID])
go

CREATE NONCLUSTERED INDEX
[_dta_index_Log_Row_10_2137058649__K3_K2]
ON [dbo].[Log_Row]
( [Datum] ASC,
[Log_ID] ASC )ITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF) ON [PRIMARY]
go
```

c. Egy cég összes naplóazonosítója.

```
SELECT     Company.Name, Product.Name AS Prod,
[Log].Log_ID, Company.Company_ID
FROM  Company INNER JOIN
    Product ON Company.Company_ID = Product.Company_ID
    INNER JOIN [Log] ON Product.Product_ID = [Log].Product_ID
WHERE   (Company.Company_ID = 1)
```

A használt index:

```
use [TestDatabase]
go
```

```
CREATE NONCLUSTERED INDEX
[_dta_index_Log_10_5575058__K2_1] ON [dbo].[Log]
( [Product_ID] ASC )
INCLUDE ( [Log_ID])
WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF) ON [PRIMARY]
go
```

2.2.5. A mérések eredményei

A 2.12 táblázatban láthatjuk a mérések eredményeit.

Lekérdezés	$MSSQL$ (sec)	$Oracle$ (sec)	Δ	$Hiba(\%)$
1	0,5602	0,5476	0,0126	2,249
2	1,2132	1,1812	0,032	2,638
3	1,176	1,17018	0,00582	0,495
4	1,329	1,2912	0,0378	2,844
5	1,852	1,93122	0,07922	4,278

2.12. táblázat. Lekérdezések mérési eredményei

Nincs különbség az MSSQL 2005 szerver és az Oracle szerver lekérdezésekre adott válaszidejében. A mért eredmények 5%-os hibahatáron belül vannak. Így figyelembe véve, a gazdasági megfontolásokat és azt hogy az információs rendszer fejlesztése VS 2005 alatt C# nyelven folyik, a döntés az MSSQL2005 szerver mellett történt.

2.2.6. Összegzés

Sikerült hatékonyság mérések segítségével megkönnyíteni a választást az SQL szerverek között. Természesen a mi megoldanadó feladatunkhoz méretezve, és a végső döntésnél szem előtt tartva a gazdasági megfontolásokat is. Ezzel bizonyítottam **a 2. tézist**.

2.3. Az MSSQL szerver vizsgálata

A cél az, hogy megvizsgáljuk, hogy az MSSQL szervert hogyan tudjuk a lehető leghatékonyabban használni. Összehasonlítottam, hogy a .NET Framework-be épített ADO.NET eszközök és az MSSQL2005 tárolteljárásainak használata által biztosított sebességet.

A mérést egy teszt adatbázison végeztem el, mely egy telefonos társaság forgalma közben keletkező adatokat szimulálta. Azért egy ilyen rendszert választottam, mert ez egy kifejezetten nagy terhelést kapó információs rendszer. Azaz nagyszámú új rekord kerül be kis idő alatt az adatbázisba. Ezen adatok kivétel nélkül tranzakció vezérelten kerülnek az adatbázisba. Így nem használható ki az esetleges rekordhalmazok batch feldolgozása, és az indexek utólagos generálása. Azaz nem számíthatunk az ebből adódó hatékonyság növekedésre.

Az egri Eszterházy Károly Főiskola Számítástudományi tanszékén került felállításra egy, a méréshez elengedhetetlenül szükséges szerver számítógép. Ez a gép biztosítja a megfelelő szerveroldali teljesítményt, mely nem távolodik el a valós felhasználási környezetek biztosította lehetőségektől.

A mérés során használt hardver és szoftver rendszerek

Szerver (dragon.ektf.hu):

Processzorok típusa: 2 db Intel Pentium III Xeon

Memória: 1024 MB

HDD: 2 db SCSI vezérlésű, 30 Gb méretű, de nem Raidbe kapcsolt

Operációs rendszer: Microsoft Windows 2003 szerver

Adatbázis szerver: Microsoft SQL Server Enterprise Edition

Munkaállomások (csoportos terhelés esetén ?):

Processzor: Intel Pentium 4 (1600 MHz)

Memória: 256 MB

HDD: 1 db 40 Gb méretű IDE vezérlésű 7200 ford/perc

Operációs rendszer: Microsoft Windows XP professional SP1

Hálózat

Belső hálózat: 100 Mbps, DHCP, DNS szolgáltatásokkal

Külső hálózat: 512 Kbps ADSL, a szolgáltató által biztosított DHCP és DNS szolgáltatásokkal

A programfejlesztés a Microsoft Visual Studio .NET 2005 részét képező C# nyelven történt. Az adatbázis egy Microsoft SQL Server 2005 található.

Az adatbázis:

Segédtáblák: az előfizetők adatait tartalmazó tábla véletlenszerű feltöltéséhez használt egyszerű táblák melyek alapadatokat tartalmaznak: (sHelysegnev, sVezeteknev, sKeresztnev, sUtcanev). Ezek nem játszanak szerepet a mérésben, mindössze a környezet megteremtésében van szerepük. Mivel a rendszer egy éles rendszer egyszerűsített modellje, a kezdeti adatokat, az előfizetők adatait egy eljárás generálja a segédtáblákban tárolt adatok segítségével. Ezek a táblák nem tartoznak a klasszikus értelembe vett adatbázishoz, nem vesznek részt a mérésben, annak eredményeire nincsenek befolyással, így adatbázisba való kapcsolásuk kulcsok és referenciák segítségével felesleges, és káros. Az indokolja mégis használatukat, hogy a mérések során legenerált közel 10 millió rekordot később tudjuk használni a lekérdezések vizsgálatához, és olvasható, valósághű eredményeket, listákat kaphassunk. A mérést az alábbi táblák adatai befolyásolják, eredményeket ezek szolgáltatnak.

A tesztekhez közvetve, illetve közvetlenül szükséges adattáblák

Elofiz: a telefontársaság előfizetőinek adatait tárolja. Ezek az adatok lesznek előállítva a segédtáblák alkalmazásával.

Az ügyfelekhez tartozó telefonszámok:

A hívás módja tábla, 2.15 táblázat, amiben a kezdeményezett hívás típusát tároljuk. (vezetékes, mobil, belföld, ...)

A mérések alapjául szolgáló forgalom tábla, 2.16 táblázat, mely a hívások adatait tárolja. Ebbe a táblába került közel 10 millió rekord a mérések során. Később alapvető szerepe lesz a lekérdezések vizsgálatánál.

LogTab: A mérések eredményeit tároljuk benne, 2.17 táblázat, a rendszer automatikusan generál minden mérési feladathoz egy rekordot ebbe a táblába.

ID	Int (Identity)	Az előfizető egyedi azonosítója. A rendszer által biztosított sorszám.
VNev	Varchar(25)	Az előfizető vezetékneve (SVezeteknev táblából)
KNev	Varchar(20)	Az előfizető keresztneve (SKeresztnev táblából)
Lakhely	Varchar(25)	Város, ahol lakik (SHelysegnev táblából)
Utca	Varchar(25)	Utca (SUtcanev táblából)
SzulDatum	Datetime	A születési dátum
SzemIg	Char(8)	Személyi igazolvány száma (véletlenszerűen előállított karaktersorozat)

2.13. táblázat. Előfizető tábla felépítése

Tszam	Char(12)	Egyedi telefonszám
IDElofiz	Int	Az előfizető azonosítója, idegen kulcs, kapcsolatot tart az Elofiz táblával. A két tábla között egy-több kapcsolat van, hiszen egy előfizetőnek akár több telefonszáma is lehet, de viszont nem.

2.14. táblázat. Telefonszámok tábla felépítése

Tszam	Char(12)	Egyedi telefonszám
IDElofiz	Int	Az előfizető azonosítója, idegen kulcs, kapcsolatot tart az Elofiz táblával. A két tábla között egy-több kapcsolat van, hiszen egy előfizetőnek akár több telefonszáma is lehet, de viszont nem.

2.15. táblázat. Hívásmód tábla felépítése

A program

A kliensprogram technológiája a legújabb Microsoft fejlesztést, a Visual .Net rendszert használja. A szoftver C# nyelven íródott, mely rugalmas

ID	Bigint (Identity)	Elsődleges kulcs, a rendszer által biztosított sorszám.
DTszam	Char(12)	Az ügyfél telefonszáma
IDHMod	Int	Hívás módja, idegen kulcs a HMod táblához, a táblák közötti kapcsolatot tartja.
Hszam	Char(12)	A hívott telefonszám
Hkezd	Datetime	A hívás kezdő időpontja
Hbef	Datetime	A hívás végének időpontja
Hido	Int	hívás időtartama, értékét egy trigger számítja

2.16. táblázat. Forgalom tábla felépítése

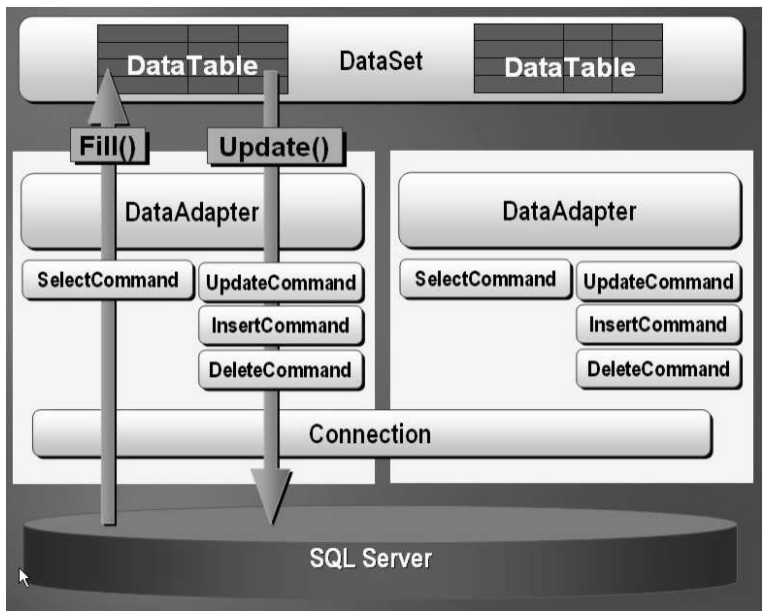
ID	int (Identity)	Elsődleges kulcs, a rendszer által biztosított sorszám.
Midopont	Datetime	Az ügyfél telefonszáma
Mkezd	Datetime	Hívás módja, idegen kulcs a HMod táblához, a táblák közötti kapcsolatot tartja.
Mbef	Datetime	A hívott telefonszám
Mido	Float	A hívás kezdő időpontja
MtipSQL	Char(10)	A hívás végének időpontja
Rekordszam	Bigint	hívás időtartama, értékét egy trigger számítja
TriggerAll	Bit	Jelző, hogy minden trigger aktiv volt-e, vagy sem. A szerver leterhelését befolyásoló tényező.
Mtip	Char(10)	Mérés típusa
Gepszam	Smallint	A mérésben részt vett gépek száma
Cel	Char(10)	Cél adattábla, esetünkben a Forg
Modszer	Char(10)	StoredProc/ADO az összehasonlítás

2.17. táblázat. A LogTab tábla felépítése

eszközt biztosít a megfelelő vizsgálatok elvégzéséhez.

Adatbázis-elérési módszerek

Mivel a vizsgálatunk a Microsoft MSSQL szerverének adat-insert műveletére terjedt ki, ezért a lehetséges DataReader élő kapcsolattal rendelkező, de csak olvasásra alkalmas és a DataSet kapcsolat nélküli lehetőségek közül a DataSet megoldást választottuk. A DataSet osztály kommunikációját az SQL szerverrel jól szemlélteti az 2.17 ábra.



2.17. ábra. A DataSet osztály a .Net rendszerben

A program két különböző adatkezelési módszert használt.

Egyik esetben az ADO.NET keret által biztosított SqlDataSet osztály DataTable osztályának a Rows collection-jét bővíti az új rekordokkal, majd a bővítés befejezésekor az adott SqlDataAdapter osztály Update metódusával aktualizálja az adatbázis tartalmát.

A másik esetben pedig tárolt eljárások használatával teszi ugyanezt. Az adatbázissal történő kapcsolattartás mindkét esetben ADO.NET alapokon nyugszik, csak utóbbi esetben az adatfelvitelért az adatbázis szerveren lévő tárolt eljárások felelősek, melyeket paraméteresen az SqlCommand osztály segítségével lehet meghívni. Amikor tárolt eljárást használunk az adatok feltöltésére, akkor mindössze a SQLCommand osztály megfelelő paraméterezésére van szükség, és a parancs futtatására. A két módszer vizsgálata volt

a célom, és ennek eredményeit jelenítem most meg.

A mérések, és eredményei

A méréseknél szem előtt tartottam, hogy a rendszer összetett felépítése miatt sok tényező befolyásolhatja az eredményeket. Ezért minden itt közölt mért eredmény nagyobbrészt néhány tíz mérésnek az átlagából adódik. A különböző rekordszámok, mérésénél így összesen mintegy 1800 mérést végeztünk. A mérési eredmények átlagolása előtt egy vizsgálaton mentek át az értékek, és az egy-két alkalommal előforduló szélsőségesen kiugró eltéréseket mutató értékek nem kerültek be az átlagosba sem. Ezen eltéréseknek mindig valamilyen, a méréstől független oka volt (hardverhiba, a szerver nem tervezett terhelése). A szerveren a mérés idejére le lettek állítva az egyébként erőforrás-igényes folyamatokat. Így nem futottak az egyéb SQL szerverek (Oracle, MySQL). Ezzel próbáltuk a legzavarmentesebb körülményeket biztosítani.

Helyi hálózaton belülről történt mérés

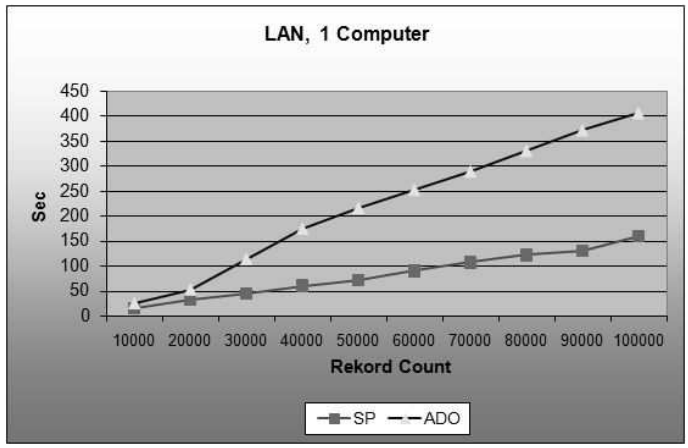
Jelölés: SP jelölje a tárolt eljárást (stored procedure), ADO jelölje az ADO.NET DataSet osztály használatát, és RCount jelölje a rekordok számát. 2.18 táblázat.

RCount	*SP*	*ADO*
10 000	14,26565	24,2969
20 000	30,9583	52,224
30 000	44,401	113,5
40 000	59,4896	174,3
50 000	71,32825	216,016
60 000	89,25	289,2373
70 000	106,37	330,556914
80 000	121,474	371,876529
90 000	129,271	406,723
100 000	159,161	289,2373

2.18. táblázat. Helyi hálózatból történt mérés eredményei.

Ezen adatok alapján készült a 2.18 grafikon, mely szemléletesen mutatja a két módszer közötti különbséget.

A kialakult görbe láthatóan jól leírható lineáris egyenlettel, ahol a



2.18. ábra. Helyi hálózatból történt mérés eredménye

$$y = mx + b$$

alakú egyenletből, az m paraméter értékét vizsgáljuk meg egymáshoz viszonyítva. Az egyenlet meghatározását a legkisebb négyzetek módszerével végeztem, így illesztettem az egyenest a mért értékpárokra. Ebből a számításból kapott eredmények:

$$m_{SP} = 0,00150933$$
$$m_{ADO} = 0,00411515$$

Ahogy a grafikon is mutatja, a tárolt eljárás használata egyenletesebb, és sokkal kedvezőbb hatékonysági mutatóval rendelkezik:

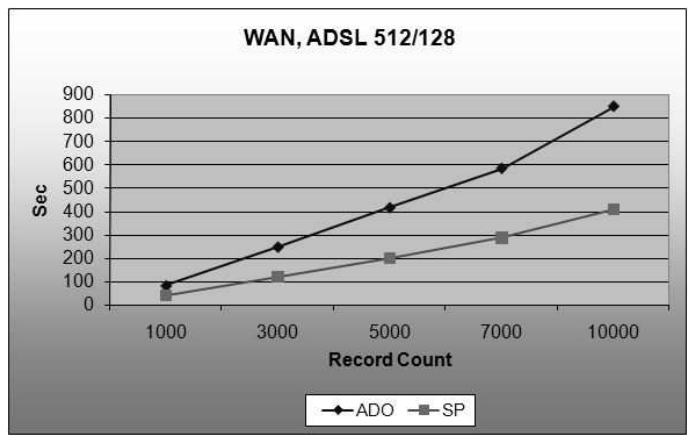
$$M = \frac{m_{ADO}}{m_{SP}} = 2,7265$$

Ez mutatja, hogy közel háromszoros sebességet biztosít a tárolt eljárás használata ebben az esetben.

Egy fontos megjegyzés: ha a rekordok felvitelénél az Update metódust nem a teljes rekordcsoport memóriabeli létrehozása után alkalmazzuk, hanem minden egyes rekord után, akkor ez a szám akár 100 szorosára is nőhet. Így ha több ezer rekordot viszünk fel, és nem feltétel a pillanatnyi aktualizálás, akkor a rekordok rögzítése után tegyük ezt meg, de mindenképpen nagyobb csoportonként.

RCount	*SP*	*ADO*
10 000	82,8625	39,9775
30 000	248,073	120,266
50 000	415,618	200,004
70 000	583,255	286,318
100 000	847,462	407,74

2.19. táblázat. WAN hálózatból történt mérés eredményei.



2.19. ábra. WAN hálózatból történt mérés eredménye

A második mérés sorozat WAN hálózatból, ADSL kapcsolat felhasználásával történt. A mérés adatai a 2.19 táblázatban láthatóak.

A kialakult görbe ebben az esetben is jól leírható lineáris egyenlettel. A számítási műveletek elvégzése után a következő együtthatók adódnak:

$$m_{SP} = 0,040665$$
$$m_{ADO} = 0,084036$$

Ahogy a grafikon is mutatja, a tárolt eljárás használata egyenletesebb, és sokkal kedvezőbb hatékonysági mutatóval rendelkezik:

$$M = \frac{m_{ADO}}{m_{SP}} = 2,06652$$

A hatékonysági mutató csökkenését befolyásolja a hálózat eltérő sebessége, és stabilitása is.

2.3.1. Következtetések

Az adatbázisok programozása, elérése felhasználói programokból napjainkban egy elterjedt, az élet minden területén megjelenő, sok helyen vezető szerepet betöltő problémakör. Az adatok kezelésének első lépése, azok tárolása, mely művelet minden rendszerben megjelenik, helyenként jelentős erőforrásokat felemésztve a rendelkezésre álló keretekből. A célom ezzel a vizsgálattal az volt, hogy a napjainkban széles körben használt rendszer esetén vizsgáljuk ezen terhelés csökkentésének lehetőségét.

A mérési eredmények egyértelműen alátámasztják, hogy a rendszer adatfelviteli hatékonysága nagy mértéken növelhető, ha kihasználjuk az SQL szerverek biztosította lehetőségeket, a tárolt eljárások használatát még látszólag más módszerrel is könnyedén megoldható feladatok esetén is.

2.3.2. Összegzés

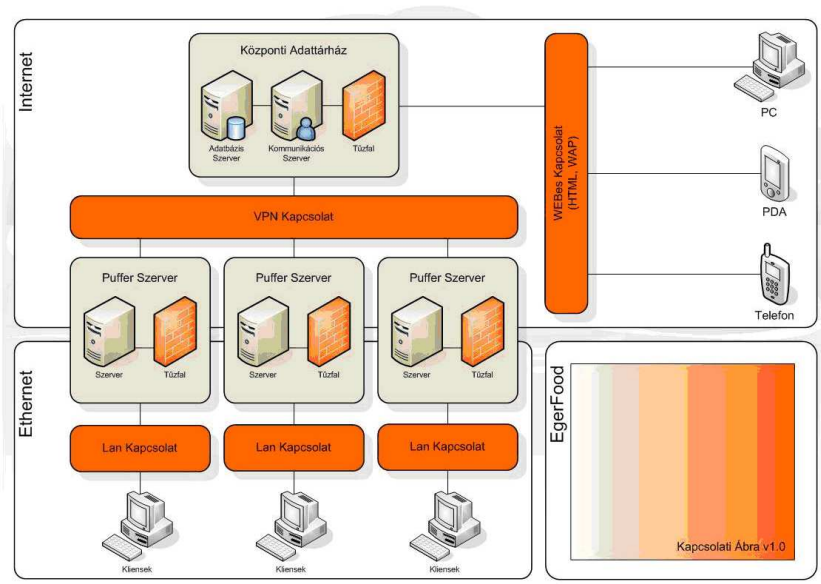
A mérési adatokból kitűnik, hogy a tárolt eljárások használata a .NET rendszer ADO.NET alrendszere által biztosított belső adatelérési komponensek használatával szemben mintegy 2-3-szoros sebességnövekedést eredményezett. Ezzel sikerült bizonyítani **a 3. tézist**.

3. fejezet

Az információs rendszer

3.1. Az információs rendszer felépítése

A rendszer jelenleg 6 ipari cég 1-1 termékének életpályáját követi és a tudásközpont laboratóriumnak eredményeit kapcsolja be a rendszerbe. Ezek az objektumok biztosítják a bemenő adatokat.[63] A fő célokat a 3.1 ábra szemlélteti:



3.1. ábra. Az információs rendszer felépítése

A kimeneti oldal többrétegű. A publikus információk, melyek akár weben keresztül, akár wapon keresztül elérhető, a rögzített adatok tájékoztató jellegű részét mutatja. Ennek célja, hogy egy azonosító számsor alapján tájékoztató adatokat adjon a rendszer a termék előállítási folyamatáról és származásáról. A védett jogosultsági szint a projekt résztvevőinek szolgáltat adatokat, melyek köre lényegesen szélesebb, mint a publikus szint adatai. Felhasználhatóak a kutatásban, a termék előállításának fejlesztésében. A vállalatok kizárólagos, belső használatú adatainak megtekintéséhez a belső jogosultsági szint szükséges. Így belső információkat használhat a felhasználó. A projekt információs rendszerének kialakításakor kiemelt szerepet kapott a megfelelő adatbiztonság biztosítása [51]. A szoftverrendszer alapja a kutatási oldal által fejlesztett adatbázis [56, 50], mely lehetővé teszi, hogy tetszőleges cég tetszőleges termékét könnyen integrálhassuk a rendszerbe. Ehhez készült a munkafolyamat gráf[1] készítő és elemző program, melynek kimenete egy olyan adathalmaz, ami alapján gyakorlatilag automatikusan előáll a szükséges kliensprogram[2] felhasználói felülete. Ez azt eredményezi, hogy a későbbi bővítések gyorsan és zökkenőmentesen megtehetőek, és a karbantartás egyetemes módszerekkel végezhető.

3.1.1. Az adatszolgáltató cégek oldala

Minden külső cégnél és a kutató laborban is úgynevezett pufferszerverek[3] kerülnek felállításra. Feladatai a következők:

- A tagok EgerFood projekthez tartozó összes adatának tárolása (például törzsadatok, mérési eredmények stb.)
- A beérkező adatok dekódolása

[1] a termékek előállításának modellje, az előállítás során keletkezett információk és mérési eredmények tárolásának leírására szolgáló eszköz. Minden cégnél egyedi. Az utolsó pontja mindig a késztermék, kezdete viszont bármilyen távolságra lehet a késztermék̈től. Általában az alapanyag raktárba való beérkezésénél kezdődik.

[2] vagy más néven adatgyűjtő számítógépek: A programban részt vevő cégek telephelyein vagy laborokban, gyárakban lévő általános célú számítógépek (PC). Csak a puffer szerverrel vannak kapcsolatban. Regisztrálják a mérések eredményeit, a naplók adatait és az összes, a munkafolyamatok során keletkező adatot. Ezeket XML formátumban tárolják és küldik a puffer szerver felé.

[3] Az adatgyűjtő számítógépek adatait átmenetileg tárolják adatbázisban, rendszerezik és küldik a központi adattárház felé. Kapcsolatot tartanak mind az adatgyűjtő számítógépekkel, mind pedig a központi adattárházzal de nem feltétlenül küldenek minden, az adatgyűjtő számítógépekről beérkező adatot a központi adattárház felé. Kezelik a riasztásokat is.

- A beérkező adatok feldolgozása és letárolása
- A kimenő adatok titkosítása és elküldése meghatározott időnként a központi adattárház[4] felé (VPN kapcsolaton keresztül)
- Biztonsági mentések megvalósítása

3.1.2. A kliens program

Az EgerFood kliens program segítségével lehet a rendszert a végfelhasználóktól elérni. A kliens program egy cégnél bárhol elhelyezkedhet, akár a gyártósor mellett is, de célszerű olyan helyre telepíteni, ahol egyébként is a naplók felvitele zajlik.

Az egyes kliens programok számítógépes hálózaton keresztül kommunikálnak egy központi számítógéppel, amit puffer szervernek hívunk. A rendszer használatához minden esetben előbb be kell jelentkezni. A bejelentkező képernyőt látjuk a 3.2 ábrán.

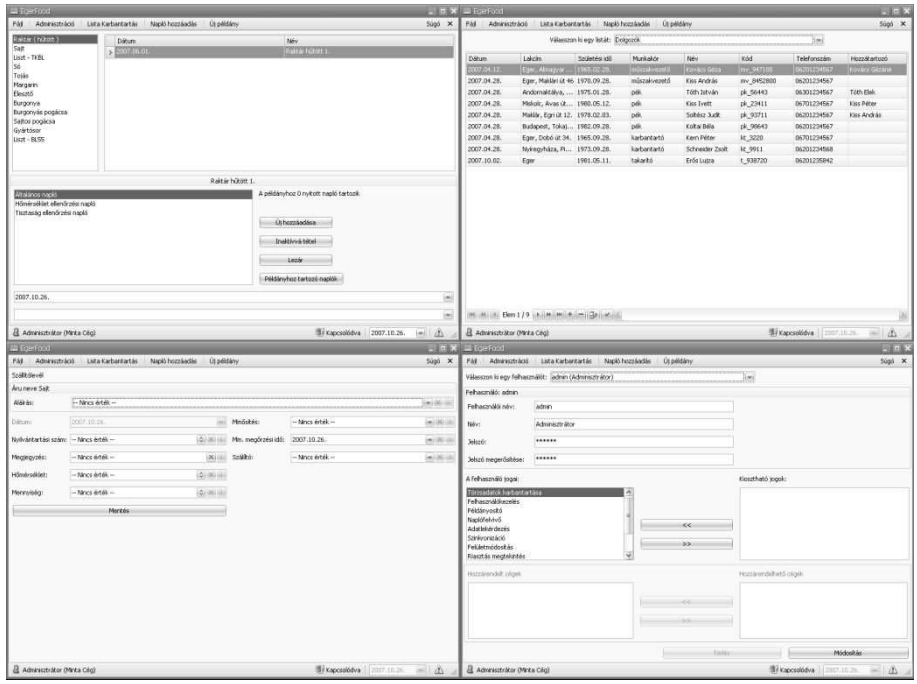
3.2. ábra. Bejelentkezés a kliensbe.

Bejelentkezéskor a cég választására csak a laborban van lehetőség. A labor felhasználói itt kiválaszthatják azt, hogy melyik céghez szeretnének bejelentkezni (azok közül, amelyekhez van jogosultságuk), de ezt a kliens futása közben is módosíthatják.

Bejelentkezés után a megszokott Windows alatt futó program felülettel találkozunk. Menüsor, munkaterület, állapotsor az ablak alján. Alapvetően igaz, hogy a megszokott konvenciók lettek követve a fejlesztés során.

A kliens program olyan mértékben testreszabható, hogy az egyes felhasználók ablakai szinte egyáltalán nem hasonlítanak egymásra. Néhány lehetséges állapotát mutatja a 3.3 ábra. Ami pedig még ennél is lényegesebb, az

[4] a programban részt vevő összes cég puffer szervereivel kapcsolatban van és az onnan érkező adatokat adatbázisban tárolja. Emellett kiszolgálja a különböző forrásból (internet, telefon, stb) érkező lekérdezéseket is.



3.3. ábra. A kliens néhány állapota.

hogy a program felületét a naplók és a jogosultságok szerkezete is jelentősen befolyásolja. Ezért nem lehet egy általános leírást adni a használatáról.

3.1.3. A munkafolyamat gráf

Az EgerFood rendszer célja a projektbe bevont élelmiszerek nyomon követésének lehetőségét biztosítani a vásárlók számára. Az EgerFood konzorciumban több élelmiszer gyártó cég van. A cél, hogy a munkafolyamataiknak megtaláljuk azt a közös részét, ami lehetővé teszi, hogy minél általánosabban lehessen megvalósítani az EgerFood rendszert.[60]

Alapvetések:

1. Minden termék nyersanyagokból és adalékanyagokból (ezeket lehet közösen kezelni, így a továbbiakban csak a nyersanyag és termék kifejezést használjuk) áll.

2. Minden termék a nyersanyagaiból valamilyen munkafolyamat során áll

elő. A munkafolyamat több lépésből is állhat, de akár, az általánosság érdekében, nulla lépésből is.

3. A nyersanyagok és a kész termékek gyakran nem azonnal kerülnek felhasználásra, tárolni kell őket.

A fenti alapvetésekből a következő következtetéseket lehet levonni:

1. **Munkafolyamat gráf:** A termék előállítása felfogható egy irányított gráfnak, amelynek csúcsai a termékek (vagy nyersanyagok), az élei pedig a munkafolyamatok, az élek a kiinduló termékből az elkészült termékbe vezetnek.
   Ebben a felfogásban ez idáig a gráf egy fa, de megengedjük azt is, hogy egy terméket részeire szedjünk szét, így visszafelé vezető nyilak is lehetségesek, illetve lehetséges olyan folyamat is (pl.: tárolás), ami nem változtatja meg a terméket, illetve egy termékhez több odavezető utat is tárolhat.
2. A 3. alapvetés kimondja, hogy az egyes termékeket (nyersanyagokat) tárolni is kell. Mivel a rendszernek nem része a raktárnyilvántartás, csak arra kell felkészülni, hogy egy esetleges raktárnyilvántartás vagy egyéb más informatikai rendszer azonosítóit lehessen tárolni [azonosító, megnevezés] formában. A megnevezés a felhasználói felületen jelenik meg a könnyebb adatfelvitel miatt, de mindig meg kell engedni az azonosító begépelését is, még akkor is, ha egyébként az azonosítóhoz rendelkezésre áll a szöveges megnevezés. Az [azonosító, megnevezés] pároknak egyszerűen bővíthetőknek kell lenni. Ezeket XML állományba kell tárolni, hogy külső rendszerből is származhassanak.
3. Habár a raktárnyilvántartás nem része a rendszernek, a tárolás alatt keletkező adatokat tárolni kell. Ilyen adat például a hűtési hőmérséklet. Adódik a kérdés, hogy a munkafolyamat gráfban hol tároljuk le ezeket az értéket, illetve milyen értékeket tároljunk?
   A. Csak olyan értéket tárolunk, amelyhez az adott élelmiszertermelő cég minőségbiztosítási kézikönyvében van "napló" leírása. Napló [5] leírás alatt olyan leírást értük, ami megmondja, milyen adatot kell, mikor és kinek rögzítenie. Ez általában papírlapon kézzel történik jelenleg. Az ilyen lapokat a felelősséget viselő alkalmazott aláírásával hitelesíti. Az

[5] olyan dokumentum, amelyben egy entitás egy példányának adatait tároljuk. Ez általában a munkafolyamat során egy-egy lépésnél az arra a lépésre vonatkozó, a munkafolyamat-gráf által meghatározott adatokat jelenti. Egy naplóba adatokat csak a cég dolgozói vihetnek fel a kliensprogramokon keresztül. A napló adatai adatbázisban tárolódnak. A nyomon követés is ezen naplók segítségével történik.

ilyen napló adatokat a [56] cikkben bemutatott *Log*, *LogRow*, *Attribute* táblákban kell tárolni.

B. Ezek az adatok mindig munkafolyamathoz kötöttek. Még azok az adatok is, amik látszólag a termék (nyersanyag) sajátjai, mint például a tárolás adatai. A tárolás is egy munkafolyamat, amit többféleképpen ábrázolhatunk a munkafolyamat gráfban:
   a. A termékre önmagába visszamenő éllel, azaz hurokkal. Előny: kevesebb csúcs; hátrány: elveszhet a munkafolyamatok sorrendje (illetve csak dátumkezeléssel nyerhető vissza).
   b. A termékből egy új termékbe, a tárolt termékbe, vezető él segítségével. Előny: A munkafolyamatok sorrendje világos; hátrány: több csúcs.
   c. Nem ábrázoljuk az ilyen munkafolyamot, csak a termékhez tartozó csúcshoz rendelünk a munkafolyamat gráfban még egy dokumentumot. Előny: Könnyebben megérthető, elegendő új csúcsot felvenni, ha ténylegesen új (félkész) termék keletkezik; hátrány: az egyes dokumentumok időbeli sorrendje nem adható meg. Mivel ennek megoldása vet fel kevesebb kérdést, ezt javasoljuk az implementációban, illetve a későbbiek során mindig használható az elv, hogy naplóbejegyzések a csúcsokhoz tartoznak és nem az élekhez.

C. A fenti pontban még egy esetet nem tárgyaltunk: a nyersanyag megérkezik egy szállítólevél kíséretében. Ekkor a fenti elvet használva a megoldás: a nyersanyagból indul egy él, a nyersanyag érkeztetése, és ez megy az érkeztetett (megérkezett) nyersanyag csúcsba. Megjegyzés: A cégekkel folytatott megbeszélés alapján ez a megoldás el lett vetetve. A szállítólevél a nyersanyag csúcshoz tartozik.

A fenti következtetések elfogadása után már csak ezeket a kérdéseket kell megválaszolni:

1. Hogyan állítjuk össze egy termék munkafolyamat gráfját?

2. Mennyire lehet attól eltérni? Lehet-e a raktáron lévő lisztet másik raktárba vinni, esetleg eladni, habár a munkafolyamat gráfban azt adtuk meg, hogy a liszt a raktárból a masszába megy?

3. Hogyan kell egy munkafolyamat gráfot példányosítani? Példányosítás alatt azt értjük, hogy megérkeznek a kézzelfogható nyersanyagok, a hűtőkamrában dermesztő hideg van, a nagynyomású főzőedényben 200 bar nyomás, azaz konkrét értékek kerülnek az adatmezőkbe.

4. Hogyan lehet ebből az EgerFood termékkód alapján adatot szolgáltatni?

Ezekre a kérdésekre a következő válaszokat adtuk:

3.1.4. Munkafolyamat gráf összeállítása

1. A termék munkafolyamat gráfját egy erre létrehozott felületen kell létrehozni az élelmiszer gyártó cég minőségbiztosításért felelős szakemberének. A gráfon meg kell adnia az egyes termékekhez (alapanyag, félkész termék, termék) tartozó naplókat. Amihez nincs ilyen napló, az az EgerFood rendszer szemszögéből érdektelen köztes félkész termék, felvétele nem javasolt, illetve az első verzióban nem lehetséges. Minden munkafolyamathoz meg kell adni annak maximális idejét órában vagy napban mérve, hogy például az ettől régebbi masszát a rendszer ne tegye ki feleslegesen a napló feltöltő felületre. A munkafolyamatok felvétele után megadható az alapértelmezett út a munkafolyamat gráfban. Ha csak egy út van, akkor a rendszer automatikusan azt választja alapértelmezettnek. A napló fejlécének felvitele munkaigényes feladat, de minden termék esetében csak egyszer kell elvégezni és csak akkor kell megváltoztatni, ha a minőségbiztosítási szabványok a cégen belül változnak. Ezt a tényt a minőségbiztosítási kézikönyvben is szerepeltetni kell! A napló fejlécének felvitele a következő lépésekből áll (kétféle naplót lehet felvinni, termékhez (alapanyag, félkész termék, termék) tartozót, vagy kivételes esemény naplót):
   a. A napló oszlop fejeinek megadása névvel, érték típussal, ami lehet szöveg is. Meg kell adni a lehetséges legkisebb és legnagyobb értéket, egy alapértelmezett értéket, illetve riasztási alsó és felső határt is. Lehetőség van színek definiálására is, hogy a későbbiekben olvasható legyen. Alapértelmezett érték megadása nem kötelező. Minden oszlophoz kell egy rövid szöveges leírása, ami a napló felvitelénél ad majd segítséget a kitöltőnek.
   b. Ha a napló nem csak oszlopokat, hanem sorokat is tartalmaz, akkor ezek adatait is fel kell vinni, azzal együtt, hogy mely sor/oszlop pozícióba nem kerülhet érték, illetve minden pozícióhoz rendelhető korlát, riasztás, alapértelmezett érték.
   c. A fent említett 1D-s (csak oszlopokat tartalmaz) és 2D-s (oszlopokat és sorokat tartalmaz) naplókon kívül a rendszer magasabb dimenziójú naplót nem támogat.
   d. Ha egy érték egy külső rendszerből jövő azonosító, akkor ezt fel kell tüntetni, hogy a megfelelő [azonosító, megnevezés] tartalmú XML ál-

lomány létrejöjjön. Az XML állomány neve és elérése itt megadható, amiről a rendszer másolatot készít, amit csak akkor használ, ha az eredeti nem található.

e. Minden naplóban kell lennie megjegyzésnek, még akkor is, ha erről a minőségbiztosítási kézikönyv nem rendelkezik. A megjegyzés mezőbe szöveges, ha máskép nem definiálják 128 karakteres, de sosem több 2048 karakternél. Ha ez mégsem elég, akkor egy külső txt vagy html állomány linkje írható be ide, amit a rendszer nem jelenít meg a vásárló felé, csak linket ad rá.

f. Ki és hogyan hitelesíti az adatokat. A hitelesítő felelősséget vállal a rendszerben rögzített adatok valódiságáról. Ezt a tényt rögzíteni kell a minőségbiztosítási kézikönyvben is!

3.1.5. Munkafolyamat gráftól való eltérés, a kivételes esemény napló

2. A munkafolyamat gráftól el lehet térni a következők szerint. Az EgerFood rendszer nem ellenőrzi, hogy felhasználtuk-e az összes nyersanyagot, illetve többet használtunk-e fel, mint ami a raktárban (a rendszer tudomása szerint) van, nem zavarja, ha hamarabb visszük fel egy későbbi munkafolyamathoz tartozó naplót, mint az azt megelőzőt (habár ilyenkor figyelmeztető üzenetet ad). El lehet térni abban az esetben is, ha előre nem látható, váratlan esemény miatt a megszokottól eltérő munkafolyamatokat kell végezni. Ebben az esetben kivételes esemény naplókat vehetünk fel. Kivételes esemény napló felvételekor a munkafolyamat gráf adott példányába fel kell venni egy kivételes esemény hatására keletkezett félkész terméket. Ehhez kell kötni a kivételes esemény naplót. Az ilyen naplót nem kell megtervezni, mint egy normál naplót, csak a releváns értékeket kell megadni a megfelelő [érték, attribútum] pár megadásával. Ha az *Attribute* táblában, lásd a [56] cikkben leírt adatbázis tervet, nem található a megfelelő attribútum, akkor annak bővítésére lehetőséget kell adni. Azt itt leírtaktól csak akkor lehet eltérni, ha létezik egy vagy több kivételes esemény napló terv. Ilyenkor ezeket kell felajánlani kitöltésre, de nem kötelező ezek közül valamelyiket kitölteni, azaz továbbra is lehetséges [érték, attribútum] pár megadása.

3.1.6. Munkafolyamat gráf példányosítása, a termékfa példány

3. Maga a munkafolyamat gráf ténylegesen egy gráf. Akár azt is megengedi, hogy a késztermékhez a kiinduló nyersanyagokból több alternatív úton jussak el, így könnyen adaptálható minden cég valós munkafolyamataihoz. A munkafolyamat gráf példánya már konkrét utakat tárol, mégpedig úgy hogy minden él iránya megfordul. Ezzel az apró trükkel kapjuk a termékfa példányt. A termékfa akkor keletkezik, amikor beérkezik egy nyersanyag a szállítólevelével együtt. Ekkor a fa gyökere lesz az érkeztetett nyersanyag, amihez hozzárendeljük a szállítólevél. Ha a nyersanyag saját termelésű (pl. forrás víz), akkor az adatainak a bemérésével keletkezik a termékfa példány. A termékfa példányok a gyökerüknél bővülnek a következő módon. Ahogy haladok előre a munkafolyamat gráfban és például a masszát keverek a lisztből, sóból, vízből, élesztőből, úgy jönnek létre újabb és újabb termékfa példányok. A fenti példa esetében létrejön egy termékfa példány, aminek a gyökere a massza, négy éle a négy nyersanyagot leíró fapéldányra mutat. A masszához kell hozzárendelni a keverés adatait tartalmazó naplót. Fontos, hogy ez csak referencia hivatkozás, így egy nyersanyag több késztermékbe is mehet. A munkafolyamat legvégén előáll a késztermék leíró fa példány, amihez egy EgerFood termékazonosítót rendelünk. Ez a hozzárendelés egyben egyedi sarzs számot [6] jelent, ha a termékkódnak van sarzs azonosító része. További fontos pont, hogy a váratlan események adatait is tárolni kell, illetve a váratlan esemény után készült terméket is szerepeltetni kell a listában, amikor egy új munkafolyamat után kiválasztom, mely termékfa írja le ennek a kiinduló termékét. Továbbá, hogy el lehessen térni a munkafolyamat gráftól, kell olyan gomb is (eltérő munkamenet), aminek hatására felsorolom az összes termékfát, ami csak van.

3.1.7. Adatszolgáltatás az EgerFood termékkód alapján

4. Minden késztermékhez tartozó termékfa kap egy EgerFood termékazonosítót, ami a sarzs részében egyedi. Minden végfelhasználói felületben ezt a termékazonosítót kérjük be és visszaadunk minden ehhez tartozó ter-

[6] a sarzs számokat a vegyipar, élelmiszeripar, esetleg a kohászat használja. Ebben a rendszerben az egyszerre, egy alkalommal felhasznált alapanyag-mennyiségnek adnak egyedi sarzs-számot azért, hogy az ebből a mennyiségből előállított késztermék-sorozatot meg lehessen különböztetni későbbi sorozatoktól. Ez a termék-sorozat követéséhez ad jó információs alapot.

mékfa minden napló információját. Ha esetleg véletlenül több termékfa tartozik a termékazonosítóhoz, akkor lehetőséget kell adnunk a szűrésre a csomagolás időpontja szerint. Ezen túl a következő szűrő feltételek adhatók meg. Csak néhány nyersanyag (adalék, összetevő, stb. ...) adatait mutassa. Ezen belül is lehessen munkafolyamatra szűrni, ahol megjelenik a váratlan esemény is.

A modell ereje

A modell lényege, hogy a munkafolyamat gráfba csak olyan termékeket (alapanyag, félkész termék, termék) veszünk fel, amihez tartozik napló. Ezeket a termékeket összekötjük a felhasználásuk sorrendjében. Ebből a gráfból lesz a termékfa, úgy hogy a termelés során, ahogy haladunk előre a munkafolyamattal, úgy minden résztermékhez létrehozunk egy újabb és újabb csomópontot, ami visszafelé össze van kötve a kiinduló alapanyagokkal, késztermékekkel. Azaz az élek megfordulnak a munkafolyamat gráfhoz képest, hiszen abban a nyersanyagtól mutattak a késztermék felé, itt a készterméktől a nyersanyagok felé. A gyártás során keletkező konkrét naplókat a termékfa egyes csúcsaihoz tartoznak. Egy csúcs több termékfában is előfordulhat, hiszen a visszamutató nyilak csak referenciák. Így megoldott, hogy egy szállítmány liszt több réteslapba is bekerülhet. Továbbá a modell biztosítja, hogy a termékfa egyes csomópontjaihoz több napló is tartozhat. Így megoldott, hogy a késztermékhez még további információkat is lehet csatolni. Például, amit a labor mér vagy a hűtés információkat, ha hűteni kell a késztermékét a szállítás előtt. [60]

A funkciók részletezését az 1. függelékben olvashatjuk.

3.1.8. A megvalósítás tapasztalatai

A megvalósítás során kiderült, hogy a naplók élhez rendelése nehezebben megvalósítható, mint a csomóponthoz rendelése. Ezért minden napló termékhez, félkész termékhez lett rendelve.Azaz a 3. fejezet 3.B.c. pontjában megfogalmazott megoldás lett előnyben részesítve.

A gráf csomópontjai lehetnek csúcs és lista típusúak. A listákat csúcsukra állított négyzetek, a csúcsokat körök ábrázolják. A gráf élei az aktuális csomópontból kiinduló gyártási folyamatokat modellezik.

Minden csomópont egy-egy entitást jelképez az adatbázisban. Látható, hogy a szállítók csomópontból származnak az alapanyagok, amelyekből a késztermékek készülnek. A dolgozók csomópont az alapanyagokra és a gyár-

tósorra van hatással, a gyártósor szintén részt vesz a késztermék elkészülésében. A négyzetek a csomóponthoz tartozó naplókat reprezentálják. A példányosító naplók ebben a nézetben nem jelöltek speciális módon az ábrán. A szállítók és dolgozók csomópont lista típusú, ami azt jelenti, hogy törzsadatokat tárolnak, vagyis beszállítók-, dolgozók-, műszakok-adatait, illetve ezekhez hasonló adatokat. A naplókat egy szerkesztő modul segítségével lehet összeállítani.

Egy napló legegyszerűbben név-érték párosként fogható fel. A naplószerkesztő modul segítségével lehet megadni a neveket és a hozzájuk rendelhető értékek típusát. A szerkesztő csoportok definiálására is lehetőséget ad. A típusok általában egyszerű típusok (szám, szöveg, dátum, idő), amelyekhez megadható riasztás is. A listákhoz csak ilyen egyszerű típusú mezők adhatók.

A csúcsokhoz viszont jóval bonyolultabb típusok rendelkezésre állnak a következő okok miatt: a legegyszerűbb esetben a burgonyás pogácsa elkészítéséhez sok összetevőre van szükség. Minden összetevőből több példány található a rendszerben.

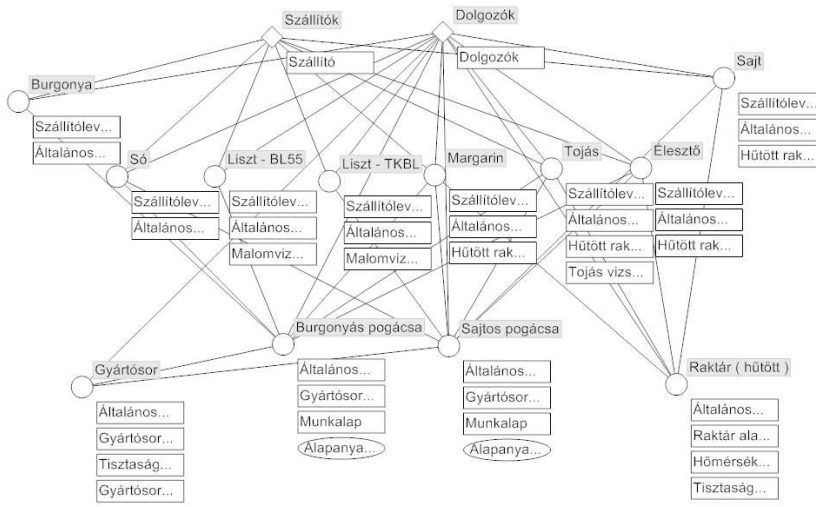
Tekintsük csak a lisztet, amiből több szállítmány is lehet a raktárakban. Egy-egy szállítmány új példányként jelenik meg, mert amikor megérkezik, példányosító naplót kell létrehozni hozzá. Amikor egy pogácsát sütnek, szükséges tudni, hogy melyik szállítmányból készült. Erre szolgál a csúcs típus. Ennél a típusnál meg lehet adni, hogy melyik csúcsról van szó. A kliens majd a napló kitöltésénél megjeleníti a megadott csúcs példányait, vagyis példánkban a liszt szállítmányait. A naplót kitöltő felhasználó kiválasztja a megfelelő példányt. Ez a módszer más esetekben is nagyon hasznos lehet.

Egy entitáshoz ún. dinamikus listát is létre lehet hozni. A gráfon ezt ovális alakzat jelképezi. A dinamikus listához tetszőleges csomópontokat lehet rendelni. Arra szolgál, hogy az ott lévő naplók közös részét kiemeljem, így a redundancia csökkenthető.

A szoftverrendszer alapja a kutatási oldal által fejlesztett adatbázis, mely lehetővé teszi, hogy tetszőleges cég tetszőleges terméke könnyen integrálható legyen a rendszerbe. Ehhez készült a munkafolyamat gráf készítő és elemző program, melynek kimenete egy olyan adathalmaz, ami alapján gyakorlatilag automatikusan előáll a szükséges kliensprogram felhasználói felülete. Ez azt eredményezi, hogy a későbbi bővítések gyorsan és zökkenőmentesen megtehetőek, és a karbantartás egyetemes módszerekkel végezhető.

3.1.9. Összegzés

Ezzel sikerült bemutatni a munkafolyamat gráf készítő programot, melynek a segítségével automatizálhatjuk a kliensprogram kezelői felületének lét-



3.4. ábra. Egy programmal elkészített gráf.

rehozását. Ezzel **a 4. tézis** bizonyítást nyert.

3.1.10. A központ

A központi adattárházon belül elkülönül az adatbázis szerver és a kommunikációs szerver [7], melyek különböző feladatokat látnak el.

Tekintsük át az adatbázis szerver egy lehetséges konfigurációját a 2. függelékben.

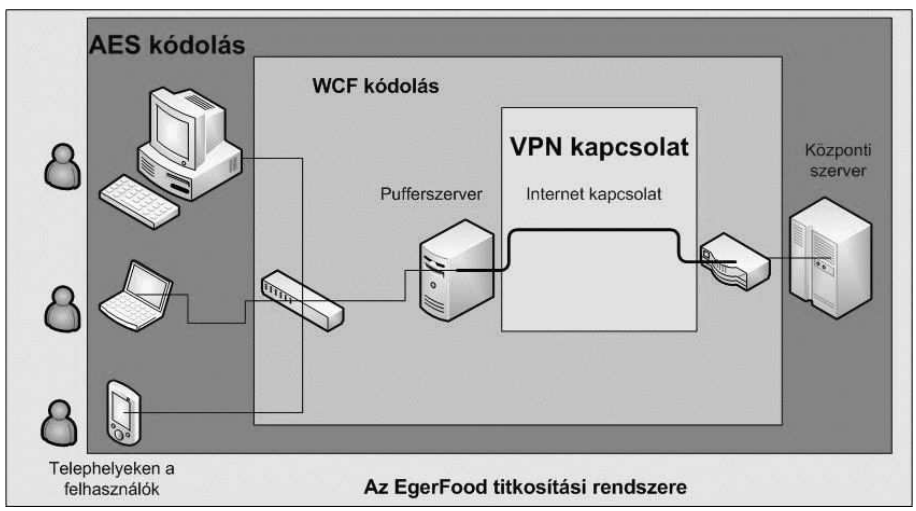
A kommunikációs szerver konfigurációt hasonló lehet, annyiban kiegészítve, hogy a szoftverek között szerepelnie kell egy IIS [8]-nek is.

[7] a központi adattárház azon része, amely az internet vagy a mobiltelefonos hálózat felől érkező lekérdezéseket szolgálja ki.

[8] Microsoft Internet Information Services

3.2. Az adatok titkosítása, a biztonság

A projekt információs rendszerének kialakításakor kiemelt szerepet kapott a megfelelő adatbiztonság biztosítása. Ennek érdekében egy háromszintű titkosítási rendszer került kialakításra, ahogy ezt a 3.5 ábrán láthatjuk. Így az adatok keletkezésének pillanatától kezdve minden adat AES-128 algoritmus szerint kerül kódolásra, az adattovábbításkor a szoftvertechnológiában legmodernebb "windows comunication foundation" eszköztárat használjuk, mely önmagában is titkosítottan végzi a kommunikációt. A hálózati adatforgalom VPN hálózaton keresztül történik, így a VPN routerek által biztosított hardveres titkosítást is kihasználható. [64, 51]



3.5. ábra. Az Egerfood kriptográfiai sémája

3.2.1. Az AES-128

A Rijndael titkosítási eljárást, mint Advanced Encryption Standardot az USA Szabványügyi Intézete (NIST) 2001-ben fogadta el, lecserélve ezzel az addigi, már elavult titkosítási eljárást, a DES-t. Az AES kiválasztását széleskörben meghirdetett verseny előzte meg. A NIST olyan szimmetrikus kulcsú blokk kódolót keresett, amely 128 bites adatblokkok kódolására képes, és ehhez háromféle kulcsméret használatát teszti lehetővé: 128, 192 és 256 biteset. A kiválasztás szempontjai voltak a kicsi méret, nehéz törhetőség, a gyorsaság és a kis eszközökben való alkalmazhatóság. 1999 augusztusában a kiválasztá-

si verseny második fordulójában mindössze öt algoritmus maradt: a MARS, az RC6TM, a Rijndael, a Serpent és a Twofish. A győztes végül a Rijndael lett, amely eredeti nevét kitalálóiról (Vincent Rijmen és Joan Daemen) kapta - továbbiakban ezt nevezzük AES-nek. Az AES kódolóban a kódolást és dekódolást különböző eljárások végzik. A kódolás négy különböző transzformáció többszöri megismétlése, míg a dekódolás az egyes transzformációk inverzének megfelelő sorrendben történő végrehajtása. [35, 29]

3.2.2. Miért az AES-t választottuk?

Tekintsük át, hogy a fejlesztő környezet milyen lehetőségeket biztosít számunkra a titkosítási algoritmusok használatára.

A Microsoft által fejlesztett dotNet 3.0 lehetőségei:

1. Titkos kulcsú titkosítás (szimmetrikus titkosítás): Ez a fajta titkosítás egy megosztott, tikosított kulcsot használ az adatok titkosítására és visszafejtésére.

    - DESCryptoServiceProvider, ami a 64 bites kulcsú DES titkosítást valósítja meg.
    - RC2CryptoServiceProvider, 40-128 bites kulcsú titkosítás. A kulcs hossza 8 bitenként nőhet.
    - RijndaelManaged, az AES megvalósítása 128, 192, 256 bites kulcsokkal
    - TripleDESCryptoServiceProvider, 3DES megvalósítása. A kulcs 128 vagy 192 bites lehet.

2. A nyilvános kulcsú titkosítás (aszimmetrikus titkosítás): Ez a fajta titkosítás a köz-magán kulcspárt használ az adatok titkosítására és visszafejtésére.

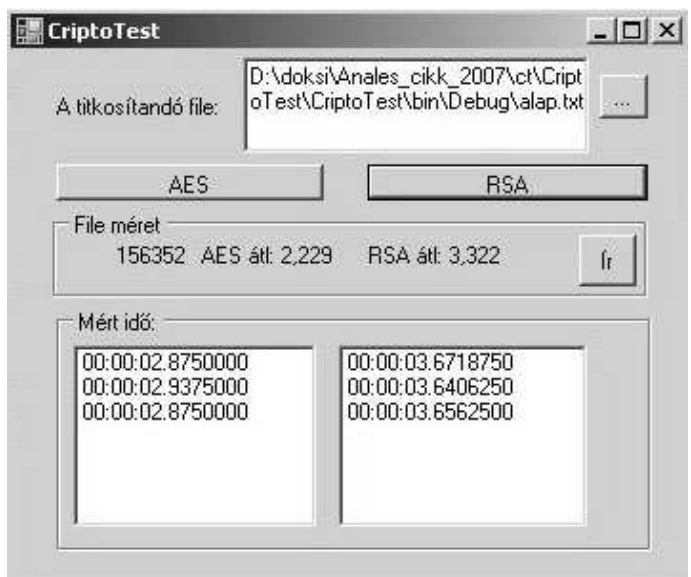
    - DSACryptoServiceProvider. Az osztály a digitális aláírás létrehozására szolgál. Az algoritmus 512-1024 bites kulcsot tud használni.
    - RSACryptoServiceProvider. Az ismert RSA algoritmus megvalósítása. 384-512 bites tartományban változhat a kulcshossz, 8 bites lépésekkel. Lehetőség van ennek kiterjesztésére a 384-16384 tartományra, amennyiben a rendszerünkre telepítjük a Microsoft Enhanced Cryptographic Provider-t.

A figyelmünket két algoritmus felé fordítottuk. Az egyik az Advanced Encryption Standard (AES), a másik pedig az RSA, melynek leírása 1976-ban jelent meg Ron Rivest, Adi Shamir és Leonard Adleman MIT.

3.2.3. Az algoritmus kiválasztása mérés segítségével

Az információs rendszerben az AES algoritmus lett kiválasztva, mert egy hatékonysági mérés azt mutatta, hogy az AES algoritmus gyorsabb a használt rendszerben (.NET 3.0).

A programmal a következőt vizsgáltam: különböző méretű szöveges fileokat mennyi idő alatt lehet kódolni egyrészt az AES, másrészt az RSA algoritmussal. Mindkét algoritmus implementálva van a framwork 2.0 és 3.0 rendszerekben. A tesztprogram felülete egyszerűen kezelhető, mint a 3.6 ábra mutatja.



3.6. ábra. Tesztprogram az AES és az RSA összehasonlításához.

A képen láthatjuk, hogy egy file kiválasztása után mind az AES mind az RSA kódolást háromszor futtatja le a program, és méri a közben eltelt időt. Majd ennek számolja a számtani közepet. Egy kattintással tudjuk állományba menteni a mért eredményeket. A file méretet és a hozzá tartozó, kiátlagolt két időértéket.

A mérések eredményét az 3.1 táblázat mutatja. Az ebben látható adatokat ábrázoltuk grafikonon. 3.7, 3.8 ábra. Szépen látható, hogy az RSA-hoz tartozó görbe a mérés teljes ideje alatt az AES-hez tartozó görbe felett marad.

$File(byte)$	$AES(sec)$	$RSA(sec)$
116362	2,010	2,307
136356	2,125	2,244
156352	2,229	3,322
176346	3,208	4,302
196342	4,000	5,265
216336	5,223	6,520
236332	6,255	7,229
256326	7,104	9,140
276320	8,317	10,244
296316	9,234	12,000
316328	11,328	14,307
326444	12,260	15,276
326444	12,620	15,156
345474	13,710	17,584
365470	15,320	18,834
405460	18,312	23,297
445450	22,148	28,100

3.1. táblázat. A tesztprogram mérésének eredményei

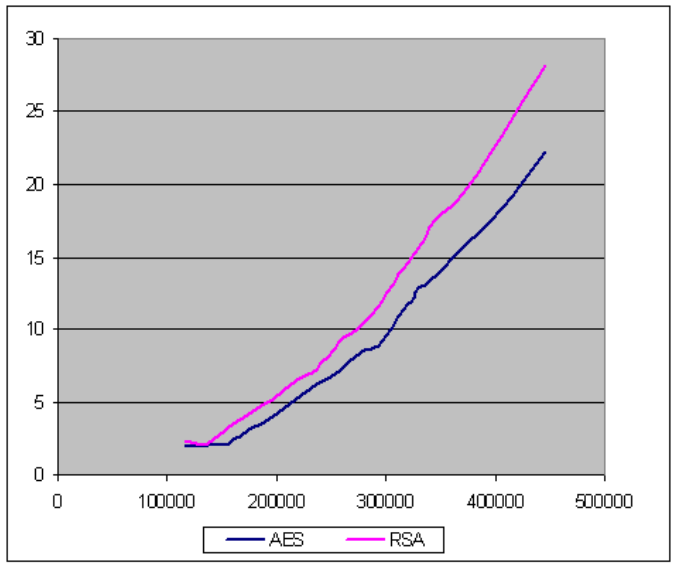
Így egyértelműen látható, hogy az RSA algoritmus ugyanazon méretű file esetén több időt igényel. A kapott eredményeket hatványfüggvénnyel közelítve a következő eredményeket kaptuk:

$f(x)$: RSA algoritmus esetén a kódoláshoz szükséges idő a fileméret függvényben

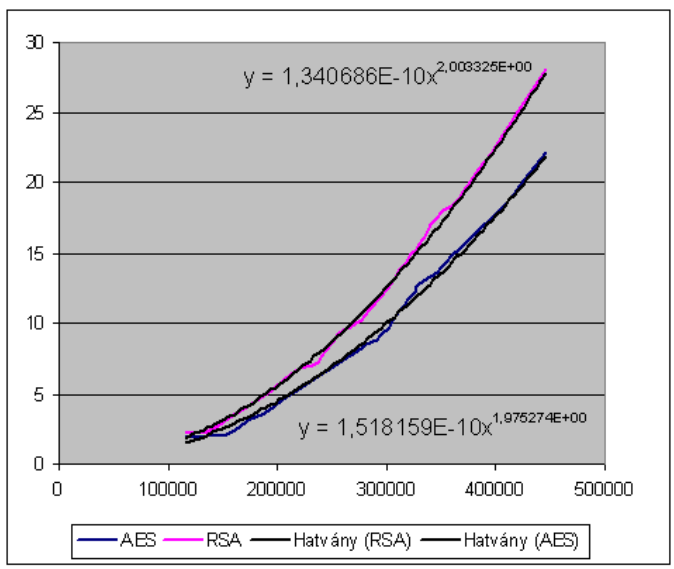
$g(x)$: AES algoritmus esetén a kódoláshoz szükséges idő a fileméret függvényben

$$f(x) = 1,340686 * 10^{-10} * x^{2,003325}$$
$$g(x) = 1,518159 * 10^{-10} * x^{1,975274}$$

A kapott eredmények alátámasztják az eleve előre elvárt eredményt, azaz, hogy a szimmetrikus kulcsú, AES algoritmus hatékonyabb, és a kódolt file méretének növekedésével ez egyre szembetűnőbb.



3.7. ábra. Mért adatok grafikonon



3.8. ábra. Interpolációs görbékkel

3.2.4. A Windows Communication Foundation

A .NET Framework 3.0 részeként megjelent Windows Communication Foundation (WCF, előző nevén Indigo)[31] a Windows platform újgenerációs technológiája elosztott alkalmazások fejlesztéséhez. A legnagyobb előnye, hogy egységes programozási modellt nyújt, legyen szó egyszerű vagy biztonságos webszolgáltatásról, rendkívül hatékony bináris formátumú üzenetsorról vagy akár peer-to-peer alapú kommunikációról. Ennek következtében a fejlesztők a jövőben egyetlen kommunikációs technológia ismeretével és jelentősen kevesebb kód megírásával, vagyis a korábbinál egyszerűbben és hatékonyabban, készíthetnek elosztott alkalmazásokat.

A WCF számos szolgáltatást nyújt többek között a biztonság, a tranzakciókezelés és megbízható üzenettovábbítás területén. Olyan Microsoft elosztott rendszer-technológiákat egyesít és terjeszt ki, mint az Enterprise Services, System.Messaging, Microsoft .NET Remoting, ASP.NET Web Services és Web Services Enhancements. Ezen túlmenően olyan teljes körű diagnosztikai funkciókkal rendelkezik, mint az üzenetnaplózás, nyomkövetés, teljesítményszámlálók, WMI [9]). A WCF igazi keretrendszer, működésének szinte valamennyi aspektusa testreszabható, kiterjeszthető. Míg egy új fejlesztői környezetet nyújt Microsoft technológiákon alapuló elosztott alkalmazások létrehozására, hatékonyan tud együttműködni a nem WCF világgal azaz más gyártók platformjaival is.

A WCF egyik fontos tulajdonsága, hogy szolgáltatások közötti kommunikációt valósít meg. Ez a technológia túlmutat a webszolgáltatások nyújtotta lehetőségek kihasználásán, hiszen célja egy, a webszolgáltatások képességeit felülmúló funkcionalitásokat összegző szolgáltatás orientált API [10] megvalósítása. A szolgáltatás és webszolgáltatás korábban még egy és ugyanazon fogalom megnevezésére volt használatos, ma már azonban nem csak szótani különbségek vannak a két megnevezés között:

1. A webszolgáltatásokat csak HTTP protokollon keresztül lehet meghívni. A szolgáltatások esetén viszont ilyen szempontból nincsenek korlátok, tetszőleges transzport protokoll használatával valósítják meg az adattovábbítást.

2. A webszolgáltatások ma csupán kérés-válasz jellegű kommunikációt képesek megvalósítani. Ezzel ellentétben a szolgáltatások számos egyéb

[9] Futás közbeni információt szolgáltat a szolgáltatásokról. Pl.: kilistázhatjuk, hogy épp milyen szolgáltatás fut és ezeknek mik a tulajdonságai, konfigurációi.

[10] Application Programming Interface = szabványos és jól dokumentált függvények és eljárások halmaza

üzenetküldési minta használatát is lehetővé teszik.

3. A webszolgáltatásokkal ellentétben a szolgáltatások rugalmasabbak, agilisek és jobban közelítik a szolgáltatás orientált paradigma szemléletmódját.

Napjaink információs rendszereiben a biztonság és a biztonságosság kérdése kiemelkedő szerepet kap. A problémák nagy részét bárki átláthatja: szeretnénk, ha adataink biztonságosan közlekednének egyik feldolgozó egységtől a másikig, s szeretnénk, ha ezeket megszerezve mások nem élnének vissza vele. Egy információs rendszer biztonságának garantálására sok technológia adott már megoldást, manapság, viszont ezek egy rendszerbe való integrálása, vagy felhasználása általában nagy hozzáértést, s gyakorlatot igényel. Az utóbbi időben egyre inkább előtérbe helyeződnek a webszolgáltatásokra épülő rendszerek kapcsán felmerülő kérdések is.

A webszolgáltatások egységesítése folyamán egy csokor olyan ajánlás született meg, amely a fentebb említett biztonság egy-egy szeletét kívánja biztosítani. A különböző platformokra ezek a megoldások részben, vagy egészben érhetők el. A példa kedvéért, .NET alapokon a Web Service Security (WS-S) ajánlások megvalósításait a Web Services Enhancements tartalmazza. A WSE főleg a kapcsolat védelmére és a felhasználók kezelésére koncentrál akár elkülönült, akár nagyméretű rendszerekben. A jogok kérdésével, s a csoport- vagy szerep alapú felhatalmazás (authorization) kérdésével külön nem foglalkozik, mindez a WCF-fel elérhetővé válik. A WCF megoldást kínál a felhasználók hozzáférés-ellenőrzésére is.

Manapság olyan átfogó és alkalmazás-független megoldás nem igazán létezik, amely a kommunikáció titkosításától kezdve a felhasználók azonosságának ellenőrzésén keresztül, a felhasználók jogosultságának ellenőrzésével bezárólag minden kérdésben egyaránt stabil alapot adva segítséget nyújtana a fejlesztőknek. A rendszer elemi jogokkal dolgozik, a könnyebb kezelés érdekében ezeket akár szerepkörökhöz is rendelhetjük. Mind szerepkör, mind elemi jog adható minden rendszerben szereplő felhasználónak. Szükség esetén az egyes elemi jogokhoz rendelhetünk előfeltételek is, azaz milyen más megszorításra van szükség ahhoz, hogy egy felhasználó az adott joggal rendelkezhessen. Lehetőség van az egyes metódusok kapcsán deklaratívan megadni, hogy milyen jogok szükségesek a meghívásához, s ezt a rendszer futásidőben ellenőrzi, minden egyéb kódsor nélkül. Könnyen látható, hogy egy metódus hívásakor gyakran nem csak az adott funkció lehet befolyással arra, hogy mely felhasználók használhatják, hanem az is fontos lehet, hogy milyen adatokon szeretnének dolgozni. A keretrendszer erre a problémára is ad egy

megoldást, ugyanis lehetőség van az egyes objektumokhoz is hozzárendelni, hogy melyik felhasználónak milyen elemi jogai vannak rá, így elérhető, hogy pl. csak a tulajdonos, vagy az adminisztrátor legyen képes módosítást végrehajtani a rendszer szempontjából kényes adatokon. A WCF biztonságkezelésének három legfontosabb aspektusa a biztonságos adattovábbítás, hozzáférés és vizsgálat vagy naplózás.

3.2.5. A VPN lehetőségei

Az adatgyűjtő szervert a legbiztonságosabban egy router mögé helyezhetjük el, amelyik fogadja a VPN kapcsolatokat, akár komplett hálózatoktól, akár egyedi munkaállomásoktól. A router feladata lehet igény szerint az adatgyűjtő szerver internet-kapcsolatának biztosítása egy NAT[11] -olt hálózaton keresztül. A router feladat ellátására egy Cisco 1812-as routert alkalmaztunk két darab LAN Port-tal. Az egyik port kapcsolódna az internethez, és NAT-olná a másik portja felé, amelyhez kapcsolódik az adatgyűjtő szerver. A router belső portjának a 192.168.0.254/24 IP-cím van megadva, mely tetszőlegesen és célszerűen változtatható a privát IP-címek tartományából. A router külső IP címének fix publikus IP-címnek kell lennie.

Biztonsági okokból, és az átláthatóság miatt célszerű a központban elhelyezkedő adatkezelő és feldolgozó munkaállomásoknak is VPN-en keresztül kapcsolódni az adatgyűjtő szerverhez. Egy másik, szintén biztonságos megoldás, ha az adatgyűjtő szerver egy másik hálózati kártyán keresztül kapcsolódik a központi adatfeldolgozó munkaállomásokhoz, amennyiben ez a hálózat eleget tesz a biztonsági igényeknek.

A távoli adatgyűjtő munkaállomások kétféle módon kapcsolódhatnak az adatgyűjtő szerverhez:

1. VPN routeren keresztül a helyi hálózat összes számítógépe elérheti a szervert

2. A helyi hálózat internet routerén keresztül VPN Client program segítségével kapcsolódhatnak a kijelölt munkaállomások a szerverhez.

Az első megoldás csak abban az esetben alkalmazható, ha nem jelent biztonsági kockázatot a helyi hálózat összes munkaállomásának a kapcsolódási lehetősége az adatgyűjtő szerverhez. A második megoldás viszont minden más esetben alkalmazható, viszont ilyenkor a munkaállomás processzorát terheli meg a titkosítási procedúra.

[11] Network Address Translation, magyarul hálózati címfordítás.

Mindkét megoldásnál ügyelni kell arra, hogy az adatgyűjtő szerver, és az adatgyűjtő, valamint adatfeldolgozó munkaállomások ne kerüljenek azonos IP tartományba. A központi router megfelelő konfigurációjával el tudja szeparálni egymástól az egyes VPN hálózatok és VPN kliensek hálózati forgalmát úgy, hogy közben a szervert mindenki lássa.

Az IPsec egy teljes biztonsági architektúrát definiál az Internet Protokoll szintjén. A szabványt az IPv4 és IPv6 protokollokhoz is kidolgozták; IPv4 esetén opcionális, míg IPv6 esetén kötelezo megvalósítása. Fontos kihangsúlyozni, hogy az IPsec nem egy VPN technológia, hanem egy általános biztonsági megoldás az IP alapú hálózatok számára; a VPN hálózatok létrehozása csupán egy az IPsec lehetséges alkalmazásai közül. Az IPsec a következo biztonsági szolgáltatásokat nyújtja:

- bizalmasság (titkosítás)
- küldő fél azonosítása (hitelesítés)
- adatcsomag integritás (sérülés, módosítás detektálása)
- korlátozott visszajátszás védelem
- korlátozott adatfolyam bizalmasság (harmadik fél nem mindig tudja megállapítani, ki kivel kommunikál)

Az IPsec biztonsági architektúra a következo biztonsági protokollokból áll össze:

- ESP, Encapsulating Security Payload:az ESP fejléc két különböző biztonsági szolgáltatás csoportot is nyújthat (vagy az egyiket, vagy a másikat, vagy egyszerre mindkettőt): egyrészt titkosítást, és az adatfolyam korlátozott bizalmasságát, másrészt megvalósítja az AH fejlécnél bemutatott hitelesítési funkciókat is.
- AH, Authentication Header: az AH fejléc adatcsomag integritás ellenőrzést, a küldő fél hitelesítését és opcionálisan korlátozott visszajátszás védelmet biztosít.
- IKE, Internet Key Exchange: az IKE protokoll egy általános hitelesített kulcscsere és paraméteregyeztetés szolgáltatást biztosít az IPsecen belül. Az automatikus kulcskiosztás további előnye, hogy periodikusan új kulcsokat lehet kiosztani, anélkül, hogy a tikosított kapcsolat megszakadna, nagyban növelve ezáltal a rendszer biztonságát.
- Kriptográfiai algoritmusok

3.2.6. Összegzés

Belátható, hogy a fent részletezett kriptográfiai eszközök alkalmazása megfelelően erős titkosítást és így biztonságot ad az ipari titkokat is tartalmazó adatoknak. Ezzel bizonyítottá vált **az 5. tézis**.

3.3. Az adatszolgáltatás

Az EgerFood rendszer legfontosabb funkciója a termékek nyomon követhetősége. A rendszerben szereplő cégek egyes termékei az alapanyagok beérkezésétől a boltok polcaira kerüléséig számos munkafolyamaton esnek át. A rendszerben az egyes részfolyamatok minden lényeges adata tárolódik. A folyamat végén a termék kap egy egyedi azonosító kódot, amely a csomagoláson található. A kód és az EgerFood adatbázis segítségével a termékkel kapcsolatos minden lényeges adat (minőség-megőrzési idő, alapanyagok és azok származási helye, gyártók, tárolási adatok) visszakereshető. A keresésre bármely vásárlónak lehetősége van, amennyiben rendelkezik

- internetes hozzáféréssel
- WAP-képes mobiltelefonnal

Két fontos megjegyzés: A keresés során csak azon adatok jelennek meg, amelyeket a gyártók nyilvánosnak minősítettek. A keresés mobiltelefonos formája kevesebb adatot szolgáltat, mint az internetes.

3.3.1. A WEB elérés

Internetes elérés esetén legalább InternetExplorer 5.5, Firefox 1.5, vagy ezzel egyenérték böngésző szükséges, amiben engedélyezni kell a JavaScript és a cookie-k használatát.

A böngészőbe gépeljük be a következő címet: http://egerfood.eu A megjelenő oldal felső mezőjébe írjuk be a kérdéses termék csomagolásán található termékkódot (egy élő kód például: 110007112313). A beviteli mező alatt egy biztonsági ellenőrző kód látható, amely a rosszindulatú internetes támadások ellen véd. Ha létező kódot adtunk meg és az ellenőrző mezőt is helyesen adtuk meg, akkor egy hasonló képernyőt láthatunk, mint a 3.9 ábrán.

Egy lekérdezés során a termék életútját tetszés szerinti mélységben nyomon követheti. A böngésző megjeleníti a riasztásokat és egyéb megjegyzéseket is. Lehetőség van minden egyes termékhez saját honlap rendelésére is.

3.9. ábra. WEB lekérdezés

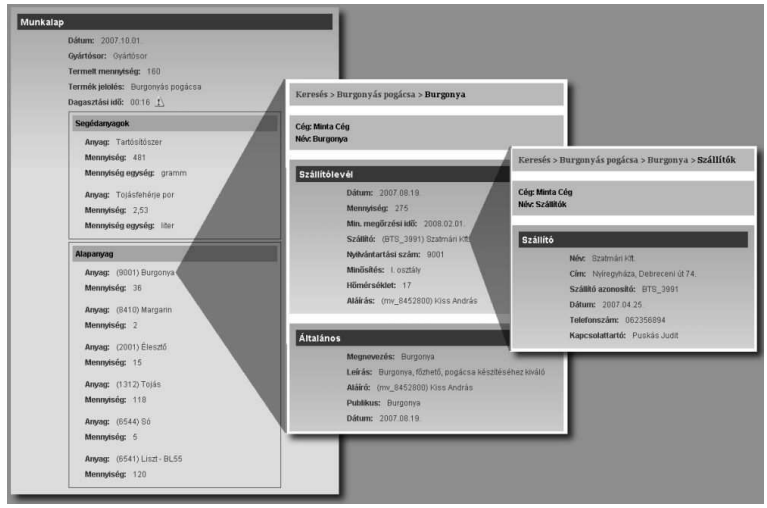
A napló tartalmazza a termék azon adatait, amelyek a gyártás közben keletkeztek. Ez a tulajdonképpeni nyomon követési információ. Megjegyzendő, hogy itt nem látható minden adat, csak azok, amelyeket a gyártó cég nyilvánosnak minősített. A napló neve piros címsorral jelenik meg, alatta találhatóak a naplóban szereplő adatok.

Az alapanyagok csoportban szereplő anyagok hivatkozásként is szolgálnak. Ha közülük valamelyikre rákattint, az adott alapanyag naplóit tekintheti meg. A Munkalap napló gyártósor adata is hivatkozás, az adott gyártósor naplóira mutat. A hivatkozott naplókból aztán tovább tud haladni, egyre mélyebbre a termék életútján. Ezt szemlélteti a 3.10 ábra.

A munkalap napló alapanyag csoportjának burgonya elemére kattintva megtekintheti annak naplóit (szállítólevél és általános), majd a szállító nevére kattintva annak egyetlen naplója, vagyis a szállító adatai jelennek meg.

A megjelenítés során még két további, eddig nem tárgyalt elem jelenhet meg az eredmények között:

- riasztások: az egérrel az ikon fölé állva egy szövegbuborékban megjelenik az elemhez tartozó riasztás. A riasztás a rendszer olyan mechanizmusa, hogy a felhasználók a termékekkel kapcsolatos rendellenes, nem szabályszerű eseményeiről figyelmeztetést kapjanak. Riasztásként jelenik meg például, ha egy raktárban a hőmérsékleti mérési eredmények



3.10. ábra. WEB lekérdezés több mélységben

nem felelnek meg az előírásoknak; ha nem a megfelelő ideig sütötték a terméket vagy bármi. A riasztás azonban nemcsak rendellenes eseményeket jelenthet, hanem valamilyen információt is, amit a cég erre jogosult dolgozója fontosnak tartott felvinni az adott dologhoz. Ez mind ebben a szövegbuborékban jelenik meg.

- információ: az ikonra kattintva az adott elemről bővebb információt

kaphat. Ez általában egy hivatkozás másik weboldalra. Nem biztos, hogy mindig működik ez a hivatkozás, ezt a cég akkor tölti fel, ha a megfelelő honlap rendelkezésre áll.

3.3.2. A WAP elérés

A kereséshez legalább 1.2-es WAP böngészővel ellátott mobiltelefonra van szükség. Ez ellenőrizhető a *http://www.mobilport.hu/* oldalon. A keresés mobiltelefonos formája kevesebb adatot szolgáltat, mint az internetes.

A kezelése hasonló. A csomagoláson lévő kód megadásával lehet indítani a keresést. A keresés a kiszolgáló számítógépek leterheltségétől és a telefonos kapcsolat sebességétől függően eltarthat pár másodpercig.

Az eredményt a kiszolgáló a telefon memóriájának megfelelő méretűre tördeli, ezért előfordulhat, hogy egy eredményhalmaz megjelenítése közben lapozni kell az oldalak között.

3.11. ábra. WAP-os lekérdezés eredménye

Egy lehetséges megjelenést láthatunk a 3.11 ábrán a 81. oldalon. A felső sorban egy ún. navigációs sáv található, amelyről láthatja, hogy milyen mélységben tekintette már meg az adott termék adatait. Felül Alapadatok címen a cég és a termékkód által azonosított dolog neve látható. Azért dolognak nevezzük ezt, mert nem minden esetben késztermeket azonosít a termékkód. Az Alapadatok után jelennek meg a termék naplói. A napló tartalmazza a termék azon adatait, amelyek a gyártás közben keletkeztek. Ez a tulajdonképpeni nyomon követési információ. Az, hogy egy naplóban milyen adatok jelennek meg, az adott cégtől és a napló típusától függ. A képen egy mun-

kalap és egy gyártósor napló látható a Minta Cégnek megfelelő felépítésben.

A mobiltelefonok memóriamérete és képességei nem egyformák. A kiszolgálónak erre fel kell készülni. Előfordulhat, hogy egy termék naplói nem férnek ki egy oldalra még a lapozás használatával sem, mert a telefonnak nincs elegendő memóriája. Ebben az esetben a megjelenítendő oldalt kell több darabra tördelni. A darabok között az oldal legalján látható »> és «< hivatkozásokkal navigálhat. Ez független a fentebb leírt lapozási funkciótól, ami a naplókat osztja több oldalra.

Az alkalmazott technikák

A wap-on keresztül, mobiltelefonról érkező lekérdezés ugyanarra a címre érkezik, amely az internet felől érkező lekérdezéseket is kezeli. A főoldal megvizsgálja, hogy a lekérdezés milyen forrásról érkezett és annak alapján szolgálja ki a kérést. A modern mobiltelefonok műszaki tulajdonságai nagyon sokfélék. Ezért nem lehet általános, minden készülékre alkalmazható módszert találni; a lekérdezés eredményeit az adott készülék tulajdonságainak figyelembe vételével kell visszaadni. Ehhez viszont tudni kell, hogy a készülék milyen tulajdonságokkal rendelkezik.

A legtöbb mai mobiltelefon a wap-os böngészés során a wap-cím mellé egy ún. UAP (User Agent Profile) információt is elküld a kiszolgáló felé. Ez tulajdonképpen egy url, ami egy xml formátumú, általában a gyártó szerverén lévő dokumentumra mutat. A kommunikációs szerver kiszolgáló szoftvere letölti ezt az xml-t, amiben a telefon pontos műszaki adatai szerepelnek szabványos formában. A wap-kiszolgáló ebből kinyeri a számára szükséges információkat és ezeket felhasználva adja vissza a keresés adatait. Mivel egyszerre nagyon nagy mennyiségű lekérdezés érkezhet a kommunikációs szerverre, nem lenne szerencsés, ha minden egyes alkalommal megpróbálná letölteni az adatokat tartalmazó xml állományt. Ezért egy cache-elési technikát tartalmaz. Ennek lényege, hogy amikor ismeretlen uap url érkezik, akkor a cache modul megpróbálja letölteni az xml-t. Ha sikerült, akkor visszaadja a kiszolgáló modulnak a kért adatokat, de az xml állományt elmenti a szerverre is. A legközelebbi lekérdezéskor így már nem szükséges azt újból letölteni. Sikertelen letöltés esetén (az url által megadott címen nincs megfelelő állomány) alapértelmezett adatokat ad vissza, de így a mobiltelefonra küldött információk nem biztos, hogy megfelelnek a telefon képességeinek. A cache modul biztonsági okokból egy sikertelen lekérdezést egy adott url-ről naponta csak egyszer próbál meg. Az előzetes tesztek szerint a mobiltelefon készülékek kevesebb, mint 10%-ánál merülhet fel ilyen probléma és inkább csak az ismeretlenebb gyártók vagy régebbi modellek esetén.

4. fejezet

Előrelépési lehetőségek, az azonosítás kiterjesztése

Az utóbbi évek rohamos technológiai fejlődésének köszönhetően egyre nagyobb teret hódítanak a hitelkártya méretű, műanyag kártyába ültethető, univerzálisan alkalmazható az adatokat biztonságosan tároló ún. multifunkciós intelligens kártyák. Nemzetközi gyakorlatban a legkülönbözőbb alkalmazási területeken sikeresen bevezették és alkalmazzák az intelligens kártyát, amely elsősorban közlekedési, továbbá kereskedelmi, pénzügyi, közigazgatási, egészségügyi, élelmiszer nyomonkövetési és szociális szolgáltatások elektronikus igénybevételét támogatja.

4.1. Kártyák jellemzői

A chipkártya belső felépítése, elektronikája képes az adatok gyűjtésére, tárolására és módosítására, és egyéb eszközök segítségével ezek az adatok elektronikusan továbbíthatók számítógépre. Kártyaolvasó terminálok használatával a kártya használója viszonylag egyszerű módon megtekintheti saját adatait (beleértve az azonosítási, hitelesítési és a digitális aláírással kapcsolatos adatokat), és ezzel az eszközzel "kulcsot" kaphat a nyilvántartásokban szereplő személyes adatainak "kinyitásához" ill. eléréséhez is. A közszférában mivel egyrészt szenzitív adatokról van szó (személyes, egészségügyi stb. információk), másrészt hivatalos ügyek intézéséhez kapcsolódik a megfelelő biztonsági szint biztosítása elkerülhetetlen.

Minden egyes funkcionalitáshoz külön kártyát nem célszerű kiadni, ezért a racionális és költséghatékonysági szempontok miatt, a többfunkciós ún. multiapplikációs eszköz kibocsátására kell törekedni. Ez azt jelenti, hogy

az alkalmazástól függően kontaktusos és kontaktusnélküli (rádiófrekvenciás) üzemmódban is kell tudnia a kártya és a kártyakezelő eszköznek egymással kommunikálni. Kártya tekintetében pedig hibrid vagy duálinterfészes megvalósítás kialakítása képzelhető el.

Ugyanakkor az azonosítási funkciók és szolgáltatások igénybevételének jogosultság ellenőrzése bármely EU tagállamban felmerülhet, ezért fontos az interoperabiliás követelményének, valamint az európai mértékadó szabványoknak való megfelelés is.

4.1.1. Adatvédelem

Az intelligens kártya eszköz lehet a szolgáltatások igénybe vételéhez, az egyének és a tevékenységeikre vonatkozó adatok összesített nyilvántartásaiba történő továbbításához. Számos esetben a nyert adatok gyűjtése, illetve megfigyelése a kártyatulajdonos előnyét is szolgálja, de fontos egyértelműen előre tisztázni, hogy kik, milyen adatokat gyűjthetnek, a gyűjtött adatokat milyen célból hol és hogyan tárolhatják. Ez abból a szempontból is különösen fontos, mivel az érintkezésmentes kártyákat egy bizonyos távolságon belül anélkül tudják olvasni és aktualizálni, hogy szükség lenne hozzá a kártyatulajdonos közvetlen hozzájárulásához, ill. egyéb tevékenység elvégzéséhez. A fentiek alapján mindenképpen kívánatos az adatvédelmi törvény követelmények szigorú betartása. A törvényi kereteken belül a korszerű technológia ugyanakkor garantálja, hogy a megfelelő hordozóra telepített egyéb alkalmazások és funkciók az elektronikus ügyintézéshez nélkülözhetetlen személyes azonosítókat kezelő alkalmazástól biztonságosan elkülönítve legyenek használhatók.

4.1.2. Technológia

Az intelligens kártya (további elnevezései: chipkártya, mikroprocesszoros kártya, smart card) egy olyan szabványos méretű (ISO 7810 szabvány szerinti) plasztik lap, amely tulajdonképpen egy külső táplálású számítógép. Rendelkezik saját processzorral, RAM- és ROM-típusú memóriával, de háttértárolási lehet5sége nincs, vagy nagyon korlátozott. A külső táplálás pedig azt jelenti, hogy általában a kártyán nincs áramforrás, így az csak az író-olvasó berendezésbe helyezve vagy azt bizonyos távolságra megközelítve képes a működésre.

Alap architektúra szempontjából a chipkártyák három nagy csoportra oszlanak:

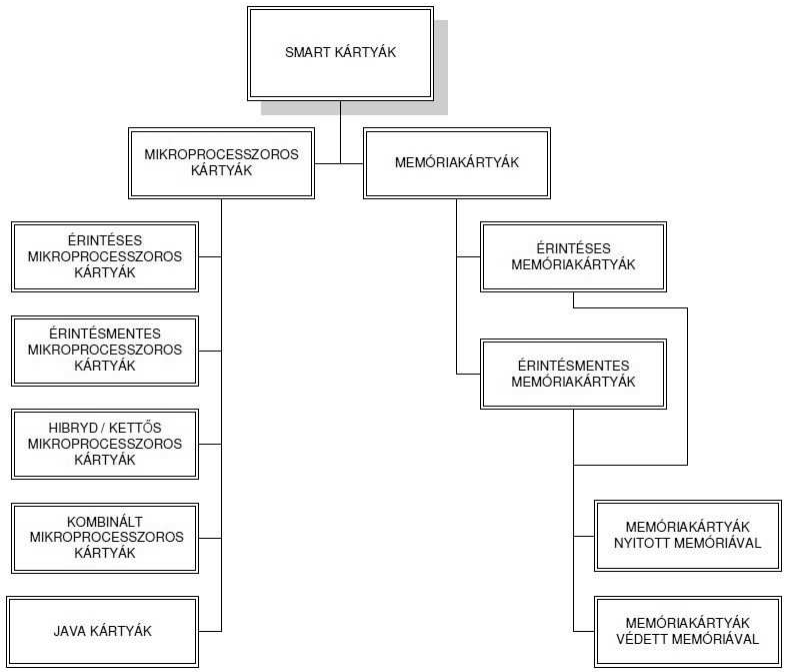
- **Memóriakártyák** (ID kártyák): A kártya nem tartalmaz processzort, így kizárólag adattárolásra alkalmas, egy hordozható memória-darabnak tekinthető.
- **Mikroprocesszoros kártyák:** A kártya komplex vezérlőegységet tartalmaz, amely lehetővé teszi magán a kártyán számítási műveletek végzését, illetve a kártyán tárolt adatok helyben történő manipulálását és védelmét.
- **Java kártyák:** Ezek is mikroprocesszoros kártyák, de jelentőségüket tekintve mégis külön csoportba sorolandók.

A chipkártyák osztályozhatóak az alapján is, hogy a kártya és a kártyát használó informatikai rendszer között milyen módon zajlik a kommunikáció:

- **Kontaktusos kártyák:** A kártya és a kártyát használó eszköz között fémes kontaktus van, így közvetlen áramköri kapcsolatokon keresztül zajlik a kommunikáció. Ebben az esetben a kártya felületére fémes érintkezők formájában kontaktusok vannak kivezetve.
- **Kontaktus nélküli kártyák:** A kártya és a kártyát használó eszköz között rádiófrekvenciás kommunikáció zajlik, így nem kell egymással közvetlen kapcsolatba kerülniük. A kártya ekkor egy antennát tartalmaz, amely egyrészt a rádiókommunikációt, másrészt indukciós elven a tápfeszültséget biztosítja.
- **Hibrid kártyák:** Olyan kártyák, amelyek mind fémes kontaktuson, mind rádiófrekvencián keresztül képesek kommunikálni.

Végül a chipkártyák csoportosíthatóak aszerint, hogy milyen speciális célra és milyen méretben készülte:

- **SIM kártyák:** GSM telefonok azonosító kártyájaként használható,kis méretű, speciális funkciókkal ellátott mikroprocesszoros chipkártya.
- **RFID:** Különleges, matricányi méretben készülő rádiófrekvenciás kártyák (rádiófrekvenciás azonosítók), amelyek így bárhol könnyen elhelyezhetőek és leolvashatóak.
- **Utazási kártya:** Speciálisan erre a területre fejlesztették ki, szegmentált memóriája van és duális módon képes kommunikálni.
- **Egyéb**, pl. pontgyűjtés



4.1. ábra. Smart kártyák csoportosítása

Az intelligens kártyák egy másik csoportosítását láthatjuk 4.1. ábrán.

Az intelligens kártya abban is különbözik az egyszerű mágnescsíkos bankkártyától, illetve a hagyományos telefonkártyától, hogy chipje a beépített mikroprocesszora segítségével programozható, a kibocsátás után is rendelkezik olyan területekkel, ahová további adatokat lehet rá írni, majd azokat feldolgozni és kiolvasni és azokkal műveleteket végezni.

Vannak olyan chipkártyák is, amelyeket nem szükséges behelyezni az író/olvasó készülékbe, megfelelő távolságról képesek kisméretű antennájuk segítségével kommunikálni. Ezeket érintkezésmentes chipkártyának nevezzük.

A biztonságos adattárolás érdekében a kártyán tárolt adatokat a legtöbb esetben kriptográfiai módszerekkel titkosítják is, ezzel magasabb biztonsági követelményeket is ki tudnak elégíteni, szemben mágneskártyákkal.

A legfontosabb kártyaszabványok az alábbiak:

- ISO 7810: a kártyák méretének és egyéb fizikai tulajdonságainak meghatározása,

- ISO 7811, 7812, 7813: mágnescsíkos kártya leírása,
- ISO 7816: chipkártya leírása,
- ISO 10356, 14443, 15593: kontaktus nélküli kártya leírása,
- EMV, SmartEMV: Euro pay, MasterCard és Visa bankkártya interoperabilitási szabvány.

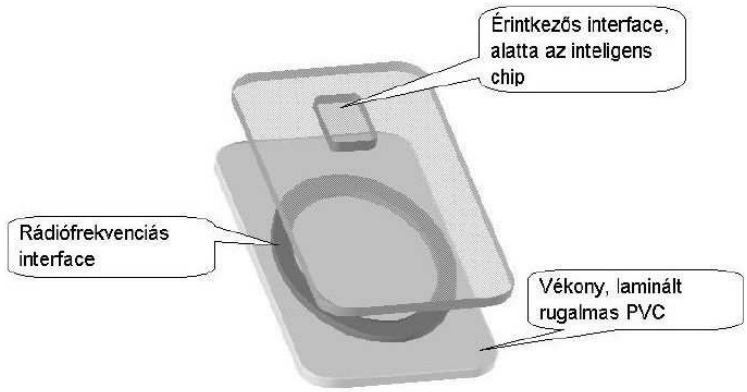
A legfejlettebb megoldások egy intelligens kártyán képesek integrálni az alábbi öt lehetőség bármilyen kombinációját:

- mágnescsík (klasszikus mágneskártya - nem intelligens),
- kontaktusos chip (klasszikus intelligens chipkártya),
- érintkezésmentes chip antennával (13.56 MHz-es technológia),
- érintkezésmentes proximity chip antennával (125 kHz-es technológia),
- Wiegand-csík (egyértelmű azonosítást segítő, biztonságot növelő technológia).

Az összetett, nagyméretű alkalmazások gyakran multifunkciós intelligens kártya használatával működnek. Egy ilyen rendszerű intelligens kártya újdonsága a kártyára integrált komplett számítógép. A kártyán elhelyezett mikroprocesszorok mindenféle számítási és egyéb műveleteket végezhetnek a memóriájukban tárolt adatokkal és programokkal, vagyis alkalmazásokat futtathatunk rajtuk. Ebben a rendszerben több alkalmazást is fel lehet tölteni a kártyára, és azok egymástól függetlenül futhatnak. A biztonsági funkciók a feltöltött alkalmazásoktól teljesen elválasztva, azoktól függetlenül működnek.

A Java Card alapja egy szilíciumlapkára felvitt, ROM-mal és EEPROM-mal felszerelt mikroprocesszor. A kártya a felhasználó által is programozható egy számítógéphez csatlakoztatott kártya író-olvasó segítségével. [14] A felhasználó programozási ismereteit használva csökkentett képességű és kisebb memóriaigényű programokat (applet) írhat, amelyekkel a kártya lehetőségeit és a kártyán tárolt adatokat bővítheti, illetve kezelheti.

A MULTOS-t eredetileg elektronikus pénztárcához szánták, így nem véletlen, hogy a bankszektorban került először alkalmazásra. A platformok közül ez tekinthető a kiforrottabb megoldásnak. A MULTOS-nak, mint szabványnak kitűnő az átjárhatósága. Tartalmaz operációs rendszert, virtuális



4.2. ábra. A dual interface-es kártya felépítése

gépet, valamint kártyamenedzsert. A fejlesztés Java, C vagy a gépi kódhoz hasonló MEL nyelveken történhet, amelyet a kifejlesztett fordítók bájtkódra fordítanak és az elektronikusan, hitelesítő központ által hitelesített alkalmazások installálhatók a kártyára. Kétségtelen, hogy ez a legdrágább kártya a piacon, de a legbiztonságosabb is.

Az intelligens kártyák használatához föltétlenül szükséges a megfelelő infrastruktúra kialakítása. Ennek a legfontosabb alkotóelemei a következők:

- intelligenskártya-kompetencia központ (a kártya, a kibocsátó, az alkalmazás és a szolgáltató hitelesítése, illetve regisztrálása),
- tranzakció hitelesítő központ.

4.1.3. Trendek

A chipkártyás rendszerek elterjedésének oka, hogy lehetővé válik olyan kártyarendszerek kiépítése, ahol egyazon kártyával többféle szolgáltatás is igénybe vehető. A kártyák használati módját meghatározó kommunikációs módozatok közül, egyre inkább terjedőben vannak a kontaktus nélküli módon használható kártyarendszerek. Ezek előnye, hogy a kártya, a kezelő berendezés közelébe helyezve anélkül működésbe képes lépni, hogy az eszközzel fizikai kapcsolatba lépne. [11] Ezáltal nagyságrendekkel csökkenthető a kártya használatának időtartalma.

4.2. Kártyatartósság és adatbiztonság

Az intelligens kártyát használó alkalmazások biztonságát a fizikai, ügyviteli és algoritmusos védelem hármas egysége határozza meg [8, 9]. Az informatikai alkalmazások és eszközök biztonságos fejlesztésére és biztonsági értékelésére létrehozott Common Criteria alapján kidolgoztak egy intelligenskártya védelmi profilt [10], amelynek hatóköre a kártyába ültetett chipre és a kártya operációs rendszerére terjed ki.

A védelmi profil részletesen tárgyalja a veszélyforrásokat, az elérendő védelmi célokat és az azokat kielégítő követelményeket. Kitüntetett szerepet kap az intelligens kártyák biztonságos működése terén a SAM (Security Access Modul), amely egy alkalmazás és a kártya közötti kommunikáció biztonságáért felelős. A felhasználó által működtetett alkalmazás valójában a SAM-ot utasítja, azzal kommunikál, majd pedig a SAM fordítja le az utasítást a kártya számára, miközben a kártyaprocesszor ellenőrzi a művelethez való jogosultságot is.

A kártyatartósság arra vonatkozik, hogy a kártya mennyire képes ellenállni különböző környezeti hatásoknak. Ezek lehetnek kopásállóság, (a kártyát az olvasón áthúzzák), képhalványulás elleni védelem (napfény okozta), illetve víz-és savállóság (folyadékba esés esetén).

A kártyabiztonság azt jelenti, hogy a kártya hitelessége ellenőrizhető. Többféle anyag használható a műanyag kártyák védelmére. A leggyakoribb, fizikai biztonsági elemek: a kártyatulajdonos fotója és aláírása, továbbá a hologram, a mikro-nyomtatás, a dombornyomás, a lézergravírozás és a kevéssé ismert Wiegand-csík.

Az ügyviteli védelem az informatikai rendszert üzemeltető szervezet ügymenetébe épített védelmi intézkedések, biztonsági szabályok, tevékenységi formák együttese, amelyet jogszabályok, szabványok, ajánlások és az üzemeltető szervezet Informatikai Biztonsági Szabályzata ír le.

Az algoritmusos védelem azokból az eljárásokból, protokollokból áll, amelyek az intelligens kártyát alkalmazó rendszer szolgáltatásaival egyidejűleg, velük szorosan együttműködve látják el a védelmi feladatokat. Általában egy nyilvános kulcsos infrastruktúrát (PKI) szokás alkalmazni, az elektronikus aláírásról szóló törvény is ilyet ír elő. A leggyakrabban használt RSA-algoritmus már klasszikusnak számít a PKI területén.

4.2.1. Támadások az érintkezésmentes kártyák ellen

A támadások elég széles köre ellen kell védekezni, és felkészíteni a rendszerünket. Ez egyrészt jelenti a megfelelő intelligens kártya kiválasztását,

másrészt a kártyát befogadó integrált rendszer biztonságos felépítését.

Lehallgatás

A RF hálózaton keresztül történő lehallgatás viszonylag egyszerű, a protokoll üzenetek lehallgatása, módosítása, törlése, visszajátszása viszonylag könnyen megvalósítható. [15] A lehallgatás célja:

- Adatlopás ill. az adatok célzott módosítása
- Szolgáltatás lopás
- Jogosultság lopás
- A kommunikáció bénítása, manipulálása

Titkosítással illetve hitelesített üzenetküldéssel (Message Authentication Code) lehet védekezni a lehallgatás illetve a visszajátszásos támadások ellen.

Működés akadályozása

A kártya véletlenszerű kiejtése a rendszerből, illetve a jóhiszemű felhasználók vagy az alkalmazottak megtévesztése a cél. Csak oktatással, folyamatos tájékoztatással előzhetők meg a problémák. Megbízható mentési és visszakeresési mechanizmust kell biztosítani.

Szolgáltatás igénybevételének megtagadása

Minden jogosultság eltávolítása a kártyáról az ügyfél tudta és beleegyezése nélkül. A kártya távvezérléssel "semmisíthető" meg, "üríthető" ill. az ügyfelet egy új funkcionalitásában jól működő kártya beszerzésére kényszerítheti a támadó.

Kártya ellopás és klónozás

A kártya, mint minden más értéket képviselő eszköz lopás tárgyává válhat, így ugyanolyan biztonsági lépéseket kell a birtokosnak tennie, mint bankkártya esetén. A Kártya klónozásával hamis kártyát készíthetünk, mely ellen a kártya vizuális kivitelének bonyolultságával, valamint az ilyen kártyák használatát feltételező, jól felkészített csalás analízissel kiegészített funkcionalitású alkalmazással védekezhetünk.

Alattomos kártyaműködések kezdeményezése

Tisztességtelen kereskedő véghezvihet illetéktelen tranzakciókat:

- Túl nagy összegek levonása
- Az első nagy összegű levonást követő, kisebb összegű levonás maszkolása
- A fizetendő összeg téves számlára történő átirányítása

Erős, bizalmatlansági alapokon nyugvó ellenőrzés és a megszemélyesítés megkövetelése, továbbá az ügyfél ún. jóváhagyó hozzájárulásának megkövetelése (pl.: PIN kód, nyomógomb megnyomása elfogadás esetén) a fentiekben említett "tranzakciók" kiküszöbölhetők.

Fizikai támadások

A chip áramkörének módosítása (fókuszált ionsugár segítségével), a belső EPROM tartalmának törlése UV sugárral.

A kártyán futó program végrehajtásának befolyásolása

A kártya órajelének manipulálása valamint az áramellátásának változtatása.

- időzítési fázisok módosítása
- magasabb feszültségszintek előállításának megakadályozása

A kártyán tárolt titkos adatok (kriptográfiai kulcsok) kinyerése

- Áramfelvétel analízis
- Elektromágneses analízis
- Rádiófrekvenciás analízis
- Utasítások végrehajtásához szükséges idő mérése
- A kártya orákulumként történő használata (pl.: GSM SIM klónozás)

Rosszindulatú, terminál felől indított támadások

- Hitelesítő adatok (PIN kód) ellopása
- Felhasználó megtévesztése, nem szándékolt tranzakció végrehajtása

4.2.2. Alkalmazható biztonsági szolgáltatások

Hozzáférés védelem / engedélyezés [12] :

- a rendszer erőforrásaihoz, folyamataihoz és szolgáltatásaihoz való hozzáférés korlátozása [13] [15]
- annak biztosítása, hogy ezekhez csak az arra jogosult entitások (személyek ill. programok) férjenek hozzá

Hitelesítés

- társentitás-hitelesítés: egy folyamatban résztvevő entitás azonosságának megbízható megállapítása
- üzenet-hitelesítés: egy folyamatban használt üzenet eredetének megbízható megállapítása
- tranzakció hitelesítés

Üzenetek, tranzakciók titkosítása és integritás védelme: annak biztosítása, hogy csak az arra jogosult entitások ismerhessék meg ill. módosíthassák a rendszerben használt üzenetek, tranzakciók tartalmát.

Letagadhatatlanság: egy folyamatban való részvétel letagadhatatlansága.

4.3. Jelentősebb nemzetközi kártyafelhasználások és alkalmazások

A kártyán alapuló közlekedési és egyéb alkalmazásokat is tartalmazó kártyarendszerek világszerte elterjedtnek számítanak, ami garanciát jelent arra, hogy bevált megoldásokra lehessen egy elektronikus díjbeszedési rendszert valamint kártya és tranzakció management központot kiépíteni.

Jelentős fejlesztésel történtek a kártyák közlekedésben történő felhasználása céljából. Ezek közül csak felsorolás szinjén egy pár:

4.3.1. Octopus-kártya - Hongkong

A Sony FeliCa névre hallgató kártyarendszert alkalmazták Hongkongban a metrók, buszok, és a térségi komprendszerek számára. A rendszer 2002. július óta működik és kb. 16 millió kártyát adtak már el, annak ellenére, hogy a populáció csak 6,85 millió. A kártyákat használni lehet telefonálásra vagy akár a kijelölt helyeken vásárlásra is, akár egy bankkártyát csak mindezt az érintésmentes felületen keresztül. A projekt specialitásai közé tartozik, hogy már nem csak kártyában gondolkodnak, hanem vannak egyéb szabadelvű megvalósítások is, mint pl. karórába épített chip, mely ugyanolyan funkciókkal bír, mint a normál chipkártya. A kártya alkalmazását kiterjesztették a beléptetésre és az iskolák területének megközelítésére. A 16-65 évesek körében lélegzetelállítóan magas, 95%-os az alkalmazása.

4.3.2. EzLink - Szingapúr

Az EzLink programot a Singapore Land Transport Authority (SLTA) valósította meg a hongkongihoz hasonló programelemmel rendelkezik - ezt a rendszert kibővítették szociális kérdések kezelésére. Az e-pénztárcát fizetések teljesítésén kívül egyéb programokra is felhasználták. Például az iskolákban diákigazolványként, és többek között irodai beléptető rendszerekben vagy mozijegyek vásárlására. Az iskolai program kibővítése lehetővé teszi, hogy a diákigazolványokkal nyomon kövessék a diákok igazolatlan hiányzását, de felhasználhatják a diákok arra is, hogy ezzel fizessenek az iskolai étkezésekért. Ez lehetővé teszi, hogy a diákok anélkül járhassanak iskolába, hogy készpénzt kellene maguknál tartani. A rendszer azt is lehetővé teszi a szülők számára, hogy teljes körűen ellenőrizzék, hogy gyermekeik mit ettek!

4.3.3. Navigo - Párizs

A párizsi rendszer napi 5 millió utast szolgál ki, 4 millió kártyával. E kártya a nyílt Calypso szabványra épül, amely rugalmasságot biztosít annak érdekében, hogy a kártyát más célokra is fel lehessen használni, például sportesemények látogatására vagy üzleti konferencián való részvételre, valamint a kártya hozzákapcsolható vállalati biztonsági beléptető kártyákhoz is. E program révén a munkaadók az alkalmazottak belépéséhez szükséges kitűzővel együtt a tömegközlekedési rendszerben használható bérletet is adhatnak munkavállalóiknak. A rendszer kibővíthető ugyanannak a kártyának a felhasználásával, hogy az a munkahelyi étkezdében pénzként funkcionáljon.

4.3.4. Közös vonások

Jóllehet, a fenti rendszereket különféle kezdeményezések alapján kezdték meg kiépíteni, majd a különféle igények megoldása érdekében kezdték meg a bővítést. Mindegyikben ugyanaz a közös cél lelhető fel, lépcsőzetesen kezelhető szakaszokban a kockázat csökkentése érdekében először egy alapmegoldást dolgoztak ki. Ezután következtek az egymásra épülő kezdeményezések, amelyeket mindig összehangoltak az együttműködő szolgáltatók üzleti céljaival.

A példaként említett megvalósítások közül a legfejlettebb és a legmagasabb biztonsági szintet képviselő megvalósítások a hongkongi projektek. A multifunkcionalitást tekintve szintén az ázsiai alkalmazások a legsokrétűbbek. Az új európai kialakítások azonban számos területet lefedő multifunkcionális jellemzőket hordoznak.

Az ázsiai alkalmazások nagy számban bocsátottak ki csak érintésmentes interfészes kártyákat, ellenben az európai példákkal, ahol a dual-interfészes kártyákból van jelentősen több. Az európai rendszerek teljes mértékben alkalmazták a két legfontosabb kártyaszabványt, az ISO 7816/1,2,3,4,5,6 és ISO 14443/A,B, ellenben az ázsiai megoldásokkal, ahol az előbb említett szabványokat nem 100 százalékban alkalmazva úgynevezett gyártó specifikus jellemzők jöttek létre, melyekből előbb vagy utóbb akár szabvány-kiegészítés is lehet. Az eltérések javára írható, hogy ezek a biztonságosabb alkalmazhatóság és a műveletek sebességének gyorsítása érdekében jöttek létre.

Az ázsiai térségben szinte kizárólag a SONY által kifejlesztett FELICA rendszert alkalmazzák, ellenben az európai megoldásokkal, ahol sokféle gyártó eszközeiből épülnek fel a rendszerek. Az európai térségben a norvég elektronikus pénztárca megoldás, mint kártyafunkció említendő meg, mely biztosítja az átjárhatóságot a szolgáltatók között országos szinten. A kommuniká-

ció és a hitelesítés biztonsága a kártyával kapcsolatban azonos szabványokra és technikai megoldásokra épült a két térséget tekintve.

A jelentkező előnyök:

- Javuló működési hatékonyság a következők révén:
  - a bevételek biztonságos, hatékony, rugalmas és megbízható behajtása
  - rövidebb tranzakciós idő
  - csökkentett megelőző és javító karbantartási tevékenységek
  - kisebb mértékű készpénzkezelés
  - a karbantartási költségek akár 40%-os csökkentése
- nagyobb bevétel a nagyobb mértékű felhasználó-barátság révén
- nagyobb biztonságosság a csalás minimalizálásával, a kártyák letiltási lehetőségével
- üzemgazdaságosság elérése a közös infrastruktúra együttes használatával valamennyi résztvevő szolgáltató részéről
- cafeteria rendszerbe egyszerűen beépíthető szolgáltatás, mellyel a vállalkozások szívesen élnek

4.3.5. A kártya lehetséges helye a rendszerünkben

Az Egerfood rendszer egyik továbbfejlesztési iránya lehet, hogy intelligens kártyák bevezetésével segítjük a termékek áramlásának dokumentálását és követését a projektben résztvevő termelő cégek telephelyei valamint a külső cégek között. A segítségével megvalósítható a:

- gépjármű mozgás követés
- a szállítólevelek gyorsabb, elektronikus megjelenése, és integrálása a rendszerbe
- a visszajelzések és hitelesítések rögzítése

Természetesen ezeknek az eszközöknek a felhasználása, rendszerintegrációja, és bevezetése komoly erőforrást igénnyel rendelkezik. Ez vonatkozik mind a kutatás-fejlesztés kiterjesztésére ebbe az irányba, mind anyagi eszközökre. A lehetőség és a kidolgozott technológia adott, a résztvevő konzorciumi cégek, és intézmények kívánsága és lehetőségei szerint felhasználható.

5. fejezet

Összefoglalás

Jelen értekezés az Egerfood Információs Rendszert írja le. Különös tekintettel az adatbázis séma tervezésének lépéseire, az SQL szerver kiválasztásnák kritériumaira, a kriptográfiai alrendszer megtervezésére, és a szoftverrendszer kialakítására.

Az értekezés első fejezetében áttekintem a fejlesztés kezdetekor volt helyzetet. Megfogalmazom a téziseket, melyeket később bizonyítok. Áttekintem a különböző résztémákhoz tartozó irodalmi előzményeket, azokat a cikkeket, melyek segítséget nyújtottak a rendszer fejlesztésében, melyek ötletet adtak a továbblépéshez.

A második fejezetben az adatbázis-modell kialakítását elemzem. Kiindulva a megvalósítást befolyásoló tényezők, a kezdeti feltételek felvázolásától. Ezek a feltételek nagyrészt a megrendelő, a RET kutatócsoportját alkotó kollégák feltételei, melyeket helyenként át kellett informatika nyelvre fogalmazni.

Olvashatunk az adatmodell kialakításáról részletesen, és a használatáról. A kialakított adatmodell legyen általános szerkezetű relációs adatbázis, mely segítségével a cégek és a termékeik előállításának folyamatai későbbi bővítések, vagy partícionált adatbázis használata nélkül tárolhatóak. Itt bizonyítom az első tézisemet.

A második fejezet második részében bemutatom, hogy hogyan sikerült hatékonysági vizsgálatok segítségével kiválasztani a kitűzött feladatra legalkalmasabb SQL szervert. A hatékonysági vizsgálat során mind adatfeltöltés mind lekérdezés szempontjából vizsgáltuk a szervereket. Ez a két alapvető adatbáziskezelő művelet hatékonysága különböző eszközökkel növelhető[1] , de

[1] indexek használata, optimális lekérdezés-záradék összeállítás, stb.

ezek az eszközök egymás ellen dolgoznak. Így az optimális középutat kellett megkeresni. Így sikerült a 2. tézist bizonyítani.

A második fejezet harmadik részében az MSSQL szervert vizsgáltam adatfeltöltés szempontjából. A rendszerben az adatfeltöltés egy fontos feladat lesz, hiszen a kezdetben, már a pilot rendszerben is a főiskolai labor adatain kívül 6 termelő cég adatai kerülnek a központi adattárház adatbázisába. A mérésekkel sikerült igazolni a 3. tézist, miszerint a tárolt eljárások használata lényegesen hatékonyabb feltöltést biztosít, mint az ADO.NET rendszer metódusai.

A harmadik fejezetben az információs rendszer bemutatása következik. Két fontos részre oszlik. Az elsőben a kliensprogram felhasználói felületének automatikus kialakítását segítő program leírása található. A munkafolyamat gráf készítő program egy olyan XML file-t generál, mely bemenetét képezi a kliensprogramnak, és melynek segítségével létrejöhet a felhasználói felület. Ez azért fontos, mert az Egerfood rendszerben külöböző profilú termelő cégek különböző termékeinek a nyomokövetése a cél. A gyártás során a különböző termékek teljesen eltérő munkafolyamatok során és receptúrák alapján készülnek el. Az ezekhez igazodó kliensfelületek teljesen eltérnek egymástól. Így a rendszer bővítésekor két lehetőségünk van. Vagy az új termék integrálásakor "kézzel" tervezzük meg és hozzuk létre a felületet, vagy automatizáljuk, és ezzel az integrációs idő töredékére csökkentjük. Sikerült a második utat végigjárni, így ezzel bizonyítani az 5. tézist.

A harmadik fejezet második nagy részében az Egerfood rendszer kriptográfiai alrendszerét fejtem ki. Nagy hangsúly került erre a rendszerre is, mert a termelő cégek ipari titkoknak minősülő receptúrákat is tárolnak az adatbázisban. Fontos követelményük volt hogy a megfelelő szintű biztonságot tudjuk fenntartani. A háromrétegű biztonsági rendszer, az AES-128 kódolás, a WCF használata a kommunikációban és a kommunikációs csatorna VPN-be helyezése megfelelő erősségű védelmet biztosít. Ezzel a 4. tézist is sikerült bizonyítani. A vizsgálat közben méréssel igazoltam, hogy az AES-128 kódolás gyorsabb, hatékonyabb a C# nyelv eszközeit felhasználva, mint az RSA. Így a mérés alátámasztotta a választásunkat.

Az értekezés negyedik fejezete egy kitekintés a tézisek köréből, egy továbblépési lehetőséget jár körbe, mely a ma oly elterjedt és közkedvelt intelligens kártyák használhatóságát vizsgálja a jövőben a projket keretein belül.

6. fejezet

Summary

The present PhD thesis describes the EGERFOOD Information System. In particular, the database schema design steps, criteria of SQL server selection, cryptographic subsystem design and software development.

In the first chapter I take in the situation it was the beginning of a development. I formulate the theses, which evidence is later. I review literary history ancillary the various topic, articles, which helped develop the system, an idea which has been granted to proceed.

In the second chapter, I analyze the development of the database model. I start factors affecting the implementation, initial conditions presentation. These conditions are largely the customer, in terms of RET resourcer team creative colleagues, which at times had to be drawn into IT.

We can read the data model in detail the development and use. The data model is designed to be a general relational database structure, which enables companies and their products in the production processes of future enlargements, or stored without using a partitioned database. In this chapter I prove the first thesis.

The second chapter in Part Two, I will show how the efficacy trials could help to choose the most appropriate task set for SQL Server. We investigated the servers study of efficiency in terms of both data entry and inquiry. These two basic database operations to increase efficiency by various means [1]but these tools are working against each other. So, the optimum was to find a middle ground. This way, 2nd thesis show.

In the third part of the second chapter I studied the MSSQL server in terms of data insert. In the system will be an important task the data entry,

[1] use indexes, optimal select-clause, etc.

because from the beginning in the initial pilot scheme is already outside of the college lab data of the 6 firm get into the central data warehouse database. The measurements could be proved in the 3rd thesis, that is significantly more efficient use of stored procedures for uploading, such as the ADO.NET method of the system.

The third chapter follows the presentation of the information system. There are two important parts. The description of a program helping the automatic forming of the application surface of the client program in the first can be found. The WorkFlow Graph maker program generates generates like XML file, which forms the input of client programs, and by the help of which to create a user interface. This is important because in the system EGERFOOD is the goal the tracing of different products of various companies. During production of different products and recipes in a completely different workflows are used to produce it. Client interfaces act on these are unlike each other. So the enlargement of the system we have got two options. Or a new product integrating we design and create with "hand" the interface, or automated, and with that a fraction of the integration time is reduced. I managed to go through a second trip, so this show 5. thesis.

The second part of the third chapter I present cryptographic subsystem of the system EGERFOOD. Great emphasis has been on this system, because the companies stores industrial secrets in this database. Important requirement was that the appropriate level of security can be maintained. The three-tier security system, such as the AES-128 encryption, the use of WCF in the communication and the communication channel put into VPN the appropriate strength of protection. So, the 4. thesis is proved. I proved during the test measurements that the AES-128 coding faster, more efficient using the C# tools such as RSA.

The fourth chapter is outlook from the thesis. An opportunity for further acts around, which is now so widespread and popular smart cards usefullness examine in the projekt.

7. fejezet

Függelék

7.1. A modell megvalósításához szükséges szoftver komponensek és ezek funkcionalitása

1. munkafolyamat gráf szerkesztő:

   - Új munkafolyamat gráf készítése. Létrehoz egy üres munkafolyamat gráfot.
   - Új termék (alapanyag, adalék, félkész termék, késztermék) csúcs hozzáadása. Ekkor meg kell adni a hozzá tartózó naplótípust is, vagy megszerkeszteni azt, vagy lehet később is megszerkeszteni, de akkor addig nem lehet elmenteni, amíg nincs minden csúcshoz naplótípus.
   - Csúcshoz napló típus hozzáadása. Ekkor a kész naplótípusok közül választhatok egyet, vagy meghívhatom a naplótípus szerkesztőt.
   - Csúcsok összekötése éllel. Az él a munkafolyamatot adja meg. Az élnek lehet nevet adni és egy maximális időt, ameddig a folyamat tart.
   - Késztermék csúcs megadása.
   - Gráf mentése. Csak olyan munkafolyamatot lehet menteni, amiben minden csúcshoz van legalább egy naplótípus, van késztermék csúcs, és minden csúcsból vezet út a késztermékbe.
   - Gráf betöltése.
   - Gráf mentése más névvel.

- Csúcs törlése.
- Él törlése.
- Csúcs áthelyezése.
- Valahogy meg kell tudni adni, melyik az alapértelmezett része a gráfnak, azaz tipikusan mely folyamatok zajlanak a termelés folyamán.

2. naplótípus szerkesztő:

   - Új naplótípus. Létrehoz egy üres 1D-s vagy 2D-s naplót.
   - Új sor (csak 2D-snél) / oszlop felvétele. Ekkor meg kell adni az oszlop / sor megjelenő fejléc szövegét. Továbbá meg kell adni egy magyarázó szöveget, ami gyorssegítségként jelenik meg.
   - Új mező felvétele. 1D-s esetén minden fejléc alatt van egy mező, amit létre kell hozni, 2D-s esetén minden sor / oszlop metszetben van egy mező. A mező felvételénél meg kell adni annak lehetséges értékek típusát. Ha előre tudott, hogy milyen értékek kerülhetnek ide, akkor azokat fel kell sorolni figyelembe véve az előző mezők értékeit. Azaz először azokat a mezőket kell felvinni, aminek az értéke nem függ valamelyik másiktól. Aztán azokat, amikhez már megvan minden mező, amitől függnek. A körkörös függés megadása egyelőre nem támogatott. Például van alapanyag és minőség mező. Ha az alapanyag értéke tojás, akkor a minőség lehet 1. osztályú, vagy 2. osztályú, de ha az alapanyag szirup, akkor lehet aranylóan fénylő, vagy zöldeskék árnyalatú. A mező típusa csak olyan lehet, ami az *Attribute_ Type* táblában megtalálható. A következő mező típusokra megadható értékek:
     - Szám: min-, max érték (ettől kisebb, nagyobb érték nem vehető fel), alapértelmezett érték, riasztási min-, max érték, kiírási szín, betű méret.
     - Szöveg: min-, max hossz, alapértelmezett érték, riasztási min-, max érték, kiírási szín, betű méret.
     - Dátum: min-, max érték (ettől kisebb, nagyobb érték nem vehető fel), alapértelmezett érték, riasztási min-, max érték, kiírási szín, betű méret.
   - Ki nem töltendő mező megjelölése. Ezeket át kell húzni.
   - Sor / oszlop kijelölése megjegyzés oszlopnak. A megjegyzés oszlop mindig szöveges, max 2048 karakter hosszú.

- Sor / oszlop kijelölés aláírási oszlopnak. Ebbe az adatokat rögzítő személy neve / azonosítója kerül.
- Sor / oszlop törlése, másolása.
- Mező szerkesztése.
- Mentés, csak akkor lehet menteni, ha minden mező meg lett adva vagy meg lett jelölve ki nem töltendőnek, van megjegyzés sor/oszlop, és aláírási sor / oszlop.
- Betöltés.
- Mentés más néven.

3. termékfa összeállító:

   A termékfa összeállító felületén a munkafolyamat gráf látható. A felhasználó valamely csúcsra kattintva hozhat létre azzal a gyökérrel termékfát. A csúcs kiválasztása után megjelennek a lehetséges gyermekek. Meg kell tenni az összekapcsolást. Természetesen a gyökérhez tartozó naplókat is fel lehet vinni a napló felvétel komponens meghívásával. Ha a termékfa gyökere késztermék, akkor EgerFood azonosítót is meg kell adni. Nagyon fontos megérteni, hogy a csúcs gyermekei is termékfák. Mivel tipikusan alulról kell felfelé építkezni, ezért a rendszer különböző színekkel mutatja majd, hova érdemes kattintani. Ha egy félkész termék minden alapanyagából van friss, akkor az zöld lesz, mert valószínűleg épp azt kell felvinni. Ha egy késztermékhez még közel se járunk, mert a közvetlen alkotó elemeiből nincs friss termékfa, akkor az halvány kék lesz, jelezve, hogy attól még messze vagyunk. A termékfát összeállító felhasználó felelőssége, hogy a rendelkezésre álló lehetséges gyermekekből a megfelelőt válassza. Ehhez gyors információként egy-egy csomópont felé odaállva láthatja a naplóinak egy-egy jellemző részletét, például a keletkezés dátumát, de akár részletesen is megnézheti a naplóit. Pontosan meg kell érteni, hogy egy termékfa a csomópontjaihoz rendelt naplók összessége. Általában egy-egy termékfát beazonosít a gyökerének típusa és a gyökerének naplói. Tehát nincs két olyan termékfa, ahol a gyökér típusa és a hozzá tartozó naplók teljesen megegyeznek. Fontos továbbá, hogy egy-egy (például alapanyag) termékfája több más termékfában lehet részfa. Ez úgy lehetséges, hogy a gyökérből csak referencia mutat a gyermekeire. Mindig elégséges megadni a gyökér közvetlen gyermekeit, hiszen azok a megfelelő termék részfák gyökerei.

4. napló felvétel:

- Naplótípus kiválasztása
- Naplómező felvitele
- Mező módosítása
- Napló mentése. Ennek hatására a puffer szerverre kerülnek az adatok. A puffer szerverről a központi adattárházba.

7.2. A központi adattárház felépítése

- javasolt legalább 2 GHz-es Intel Xeon vagy AMD Opteron processzor használata
- legalább 2 GB memória (4 GB vagy több javasolt)
- 140 GB merevlemezterület a felhasznált lemezterület a tárolt adatok mennyiségétől és az adatredundanciától függ, a redundáns adattárolás - RAID - használata erősen ajánlott)
- 2 Gigabites Ethernet hálózati kapcsolat
- Microsoft Windows Server 2003 vagy újabb
- Microsoft SQL Server 2005 vagy újabb

Irodalomjegyzék

[1] James Sallie: An economic analysis of food safety issues following the SPS Agreement: Lessons from the Hormones dispute, CIES Policy Discussion Paper 0005, February 2000.

[2] Jacques Trienekens, Adrie Beulens: The implications of EU food safety legislation and consumer demands on supply chain information systems, IAMA Symposium, June 2001.

[3] A. Ailamaki, M. Shao: DBMbench: Microbenchmarking Database Systems in a Small, Yet Real World, in Confidential, Submitted to ICDE 2004

[4] Microsoft: Improving .NET Application Performance and Scalability, (2004), 639-682

[5] Jim Gray: http://research.microsoft.com/ gray

[6] Mike Ruthruff (Microsoft Co.): Microsoft SQL server 2000 Index Defragmentation Best Practices, 2003

[7] J. Gray., Morgan Kaufman: The Benchmark Handbook for Database and Transaction Processing Systems, Publishers, Inc. 2nd edition, 1993

[8] Ködmön J.: Kriptográfia, ComputerBooks,2000

[9] Buttyán L., Vajda I.: Kriptográfia és alkalmazásai, TYPOTEX

[10] Smart Card Protection Profile (Smart Card Security User Group)

[11] Karl Koscher, Tadayoshi Kohno, Vjekoslav Brajkovic (University of Washington), Ari Juels (RSA Labs): EPC RFID Tags in Security Applications: Passport Cards, Enhanced Drivers Licenses, and Beyond

[12] MEZGÁR, I., KINCSES, Z.: "Intelligent System Techniques in Secure Identification and Communication in Distributed Manufacturing System", In Theme Volumes on "Intelligent Systems Techniques and Applications", Editor: Professor C. T. Leondes, Vol. V., Chapter 2, CRC Press International, Boca Raton, FL, 2002, pp 39-59.

[13] KINCSES, Z.: "Security controls for database technology and applications", In "Encyclopedia of Database Technologies and Applications". Editor: Laura C. Rivero, Jorge Horacio Doorn, Viviana E. Ferraggine (Eds.) Idea Group 2005, pp 581-586.

[14] E. Poll, J. van den Berg, and B. Jacobs. Specification of the JavaCard API in JML. Fourth Smart Card Research and Advanced Application Conference (CARDIS 2000), p. 135-154 Kluwer Acad. Publ. , 2000.

[15] Hess, E., Janssen, N., Meyer, B., and Schutze, T. 2000. Information leakage attacks against smart card implementations of cryptographic algorithms and countermeasures. In Proceedings of the EUROSMART Security Conference. 55–64.

[16] Kretschmer, T.: Competing technologies in the database management systems market, CEP Discussion Paper vol. 737, pp: 5-17, 2006

[17] Zolotová, I. and FlochovÁ, J. and Ocelíková, E.: Database technology and real time industrial transaction techniques in control, Journal of Cybernetics and Informatics, vol.:5, pp: 18-23, 2005

[18] Hamid, NRA: Database imperatives in managing supply chain: an empirical study, WSEAS Transactions on Computers, vol.:2, num.: 2, pp: 379-385, 2003

[19] Milicevic, M. and Batos, V. and Mornar, V.: Real Life Data Mart-Models Comparison, WSEAS transactions on computers, vol.: 2, num.: 4, pp: 967-972, 2004

[20] Holliday, J.A. and Agrawal, D. and Abbadi, A.E.: The performance of database replication with group multicast, In Proceedings of IEEE International Symposium on Fault Tolerant Computing (FTCS29), 1999

[21] Ceri, S. and Pernici, B. and Wiederhold, G.: Distributed database design methodologies, Proceedings of the IEEE, vol.:75, num.: 5, pp: 533-546, 1987

[22] Pleshachkov, P.: Transaction Management for XML Stored in Relational Database Systems, Proc. PhD Forum BNCOD, 2006

[23] Florescu, D. and Kossmann, D.: A performance evaluation of alternative mapping schemes for storing XML data in a relational database, RAPPORT DE RECHERCHE-INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE, 1999

[24] Baraani-Dastjerdi, A. and Pieprzyk, J. and Safavi-Naini, R. and Getta, J.R.: Using cryptographic hash functions for discretionary access control in object-oriented databases, journal:JUCS, vol.: 3, pp: 730-753

[25] He, J., Wang, M.: Cryptography and Relational Database Management Systems. Proceedings of the 5th International Database Engineering and Applications Symposium. pp: 273-284, 2001

[26] Hacigumus, H., Iyer, B., Li, C., Mehrotra, S.: Executing SQL over Encrypted Data in the Database Service Provider Model. ACM SIGMOD Conference on Management of Data, 2002

[27] Iyer, B. and Mehrotra, S. and Mykletun, E. and Tsudik, G. and Wu, Y.: A framework for efficient storage security in rdbms, Lecture Notes in Computer Science, pp:147-164, 2004

[28] D. Trcek and G. Kandus: Security Policy - Human Factor Modeling and Simulation, WSEAS Transactions on Computers, vol. 2, pp.: 339-342, 2003.

[29] A. J. Menezes, P.C. van Oorschot, S. A. Vanstone, Handbook of Applied Cryptography, CRC Press 1996.

[30] Announcing the Advanced encryption standard (AES), Federal Information Processing Standards Publication 197, 2001.

[31] Sharp, J.: Microsoft Windows Communication Foundation Step by Step, Microsoft Press, 2007

[32] Diaa Salama Abdul. Elminaam, Hatem M. Abdul Kader and Mohie M. Hadhoud: Performance Evaluation of Symmetric Encryption Algorithms on Power Consumption for Wireless Devices, International Journal of Computer Theory and Engineering, Vol. 1, No. 4, 2009

[33] Hardjono, T. and Dondeti, L.R.: Security in Wireless LANS and MANS (Artech House Computer Security), Artech House, Inc. Norwood, MA, USA, 2005

[34] D. Coppersmith, The Data Encryption Standard (DES) and Its Strength against Attacks. IBM Journal of Research and Development, vol. 38, num. 3, pp. 243 -250, 19943

[35] Daemen, J., and Rijmen, V. Rijndael: The Advanced Encryption Standard. D r. Dobb's Journal, PP. 137-139. March 2001

[36] Bruce Schneier. The Blowfish Encryption Algorithm Retrieved October 25, 2008,http://www.schneier.com/blowfish.html

[37] N. El Fishawy: Quality of Encryption Measurement of Bitmap Images with RC6, MRC6, and Rijndael Block Cipher Algorithms, International Journal of Network Security, Nov. 2007, PP.241-251.

[38] Zeng, J. and Ansari, N.: Toward IP virtual private network quality of service: A service provider perspective, IEEE Communications Magazine, vol. 41, num. 4, pp: 113–119, 2003

[39] Ferguson, N. and Schneier, B.: A cryptographic evaluation of IPsec, Counterpane Internet Security, Inc., January 2000

[40] Ferguson, P. and Huston, G.: What is a VPN?, Citeseer, 1998

[41] Mackie, R.I.: Object oriented implementation of distributed finite element analysis in. NET, Advances in Engineering Software, vol.: 38, num.: 11-12, pp.: 726-737

[42] Shakhgeldyan, C. and Kryukov, V.: Integration of University Information Resources into the Unified Information Environment, Proceedings of the 10-th International Conference of European University Information Systems (ENUS 2004). Slovenia, pp.: 321-327, 2004

[43] Olson, L.: NET P2P Writing Peer-to-Peer Networked Apps with the Microsoft.NET Framework, MSDN Magazine, vol.:16, num.:2, pp.: 131-140, pub.: Citeseer, 2001

[44] Lapalme, J. and Aboulhamid, EM and Nicolescu, G. and Charest, L. and Boyer, FR and David, JP and Bois, G.: .NET Framework–A Solution for the Next Generation Tools for System-Level Modeling and Simulation, Design and Test Automation in Europe, 2003

[45] Colafigli, C. and Inverardi, P. and Matricciani, R.: InfoParco: an experience in designing an information system accessible through WEB and WAP interfaces, PROCEEDINGS OF THE ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 2001

[46] Tull, C.: WAP 2.0 development, Que Publishing, 2002

[47] Forta, B. and Fonte, P. and Bromby, D. and Lauver, K.D. and Juncker, R.M. and O'Leary, A. and Mandel, R.: WAP Development with WML and WMLScript, Macmillan Publishing Co., Inc. Indianapolis, IN, USA, 2002

[48] Teorey, T.J. and Yang, D. and Fry, J.P.: A logical design methodology for relational databases using the extended entity-relationship model, ACM Computing Surveys (CSUR) vol.:18, num.: 2,pp.: 197-222, 1986

Publikációs lista

Referált cikkek:

[49] S. Szegedi, T. Radvanyi: Determination of boron in glass by alpha-spectrometry, J. Radioanal. Nucl. Chem., Letters 187 (6) 409-417 p. (1994)

[50] Tibor Radvanyi: Examination of the MSSQL server from the user's point view considering data insertion, , Acta Academiae Pedagogicae Agriensis, 2004, 69-77 p.

[51] Kalman Liptai, Gabor Kusper, Tibor Radvanyi: Cryptographycal Protocols in the Egerfood Information System, Annales Mathematicae et Informaticae, 2007, 61-70 p.

[52] Ildiko Tothne Molnar, Tibor Radvanyi, Emod Kovacs: The usage of adapted ICT in the education of children with special educational need in different countries of Europe, Annales Mathematicae et Informaticae, 2008, 189-204

[53] Tibor Radvanyi: Use of knowledgebases in education of database management, Acta Didactica Napocensia Kolozsvár, 2008.11.06, 7-11p

[54] Tibor Radvanyi, Emod Kovacs: Infiltration of RFID technological knowledge in teaching of informatics teacher MA, Romanian Journal of Education Kolozsvár, ISSN: 2067-8347, vol.:1, num.:1, pp.: 49-54

Elfogadva, megjelenés alatt:

[55] Tibor Radvanyi, Emod Kovacs, Janos Kormos: Information Systemś experiences of EGERFOOD projekt experiences making use of him in the education of the database management, Teaching Mathematics and Computer Science, Debrecen

Konferencia kiadványban megjelent cikk:

[56] Tibor Radvanyi, Gabor Kusper: Requirement Analyzes and a Database Model for the Project EGERFOOD Food Safety Knowledge Center, 7th International Conference on Applied Informatics Eger, Hungary, January 28 - 31, 2007, plenáris előadás page 15-25

[57] Radványi Tibor: Az MSSQL szervez hatékonyságának vizsgálata adatinsert szempontjából, Agria Média Eger, 2004 október 14-16. 2004 II kötet, 451-461 oldal.

[58] Radványi Tibor: Tudásbázisok használata az adatbázis-kezelés oktatásában, Agria Média Konferencia 2006 Eger, 2006 október 16-17, Agria Média 2006, 334-339 oldal.

[59] Radványi Tibor, Kusper Gábor, Kovács Emőd: Adatforgalom és mobilkommunikáció az Egerfood rendszerben, Agria Média Konferencia 2008 Eger, 2006 október 27-28, 7 oldal

[60] Tibor Radványi, Gábor Kusper: Az EGERFOOD élelmiszerbiztonsági nyomkövető rendszer - Hogyan modellezzük a cégek munkafolyamatait , NetworkShop 2008 Dunaújváros, 2008. március 17-19, 8 oldal , https://nws.niif.hu/ncd2008/index.htm

Konferencia előadások:

[61] Radványi Tibor: Adatbázis kezelők hatékonyság vizsgálatának mérési módszere, Informatika a felsőoktatásban 2005 Debrecen, 2005. augusztus 24-26

[62] Radványi Tibor, Kiss Attila, Naár Zoltán: Élelmiszerbiztonsági nyomkövető rendszer adatbázis háttere, Alkalmazott Informatikai Konferencia 2006 Kaposvár, 2006 május 26

[63] Tibor Radványi, Gábor Kusper: Az EGERFOOD élelmiszerbiztonsági tudásközpont projekt információs rendszerének kialakítása , NetworkShop 2007 Eger, 2007. április 11-13

[64] Radványi Tibor, Kusper Gábor, Kovács Emőd: Kommunikáció az Egerfood élelmiszerbiztonsági projekt információs rendszerében, Informatika a felsőoktatásban 2008 Debrecen, 2008. augusztus 27-29

[65] Radványi Tibor, Kovács Emőd: Az RFID technológiai ismeretek beépülése az informatika tanár MA képzésbe, Multimédia az oktatásban, Debrecen, 2009 június 24-25

[66] Gabor Kusper, Emod Kovacs, Tibor Radvanyi, Peter Incze, Peter Magyar, Robert Szabo: RFID technological knowledge in our teaching, , Eger, ICAI 2010, 2010 január 27-30

Magyar nyelvű tankönyv, jegyzet:

[67] Dr. Kovács Emőd, Hernyák Zoltán, Radványi Tibor, Király Roland: A C# programozási nyelv a felsőoktatásban, Programozás tankönyv, Elektronikus tankönyv, 2005, http://csharptk.ektf.hu

[68] E. Kovács, Z. Hernyák, T. Radványi, R. Király, A C# programozási nyelv a felsőoktatásban Programozási tankönyv, "Algorithms of Informatics" kiadványban, ELTE Faculty of Informatics, 2005.

[69] Sulinet Digitális Tudásbázis, Elektronikus jegyzet, 2006 Objektum Orientált Programozás Delphi nyelven, Adatbázis kezelés, Táblázat kezelés fejezetek