UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

UPORABA RAČUNALNIŠKEGA RAZPOZNAVANJA GOVORA IN TRANSFORMACIJE NARAVNEGA JEZIKA V MDX SINTAKSO PRI KREIRANJU VEČDIMENZIJSKIH POIZVEDB IZ VELIKIH SKLADIŠČ PODATKOV

Ljubljana, september 2002 TOMAŽ ŠMID

IZJAVA

Študent TOMAŽ ŠMID izjavljam, da sem
avtor tega magistrskega dela, ki
sem ga napisal pod
mentorstvom prof. dr. MATJAŽA GAMSA in
skladno s 1. odstavkom 21. člena Zakona o avtorskih in
sorodnih pravicah dovolim objavo
magistrskega dela na fakultetnih
spletnih straneh.

V Ljubljani, dne 1.9.2002

Podpis:

KAZALO

1. OPREDELITEV PROBLEMATIKE

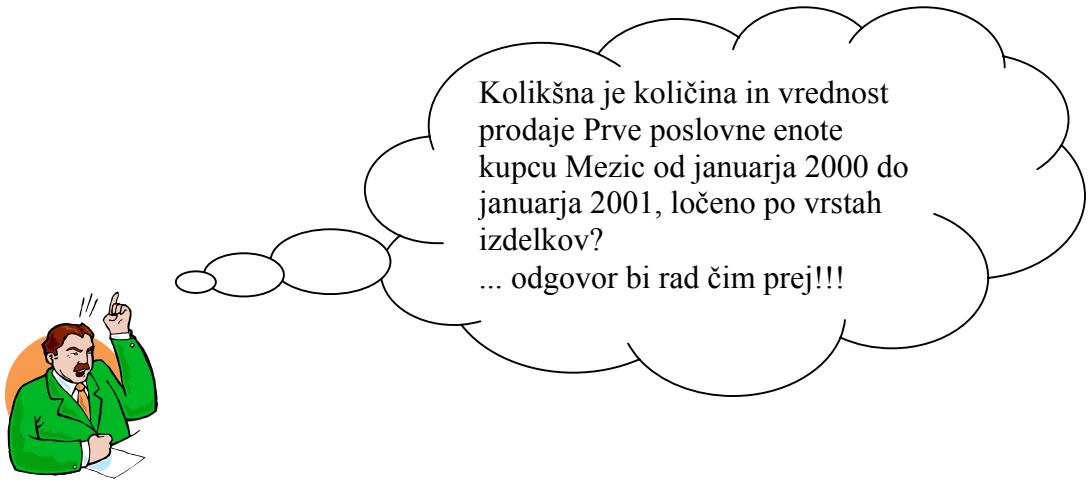
Namen naloge je izdelava računalniško podprtega sistema za podporo odločanja, ki bi uporabniku na podlagi govornega vprašanja v naravnem jeziku omogočil dostop do želenih informacij brez potrebe po znanju računalniškega programskega jezika.

Želje glede uporabe sodobne računalniške podpore pri odločanju so danes zaradi velike konkurence in dinamike na trgu še posebno prisotne v gospodarskih družbah. V tem smislu je bilo potrebno iskati rešitev, ki bi vodstveni strukturi v podjetjih omogočila enostavno, pregledno in hitro analizo prodaje.

Čeprav prikazani projekt zajema specifično področje analiz prodaje v proizvodnem podjetju, pa je namen magistrske naloge iskati koncept rešitve, ki bi bil dovolj splošen, da bi ga bilo možno uporabiti kot osnovo za gradnjo aplikacij za najrazličnejše namene uporabe.

1.1. Analiza problemskega vprašanja

Slika 1: Izhodišče raziskave - vprašanje



Vir: avtor

Slika 1 predstavlja pogosto situacijo v podjetjih, kjer se vodstvena struktura sooča z neprestanimi zahtevami po odločanju. Odgovori na kopico zapletenih vprašanj omogočajo nastanek strateških analiz, ki so ponavadi poleg neke neoprijemljive intuicije pogoj za sprejetje končne odločitve. Čeprav se je potrebno zavedati, da je pri poslovnih odločitvah še vedno človek tisti, ki se na koncu odloči med različnimi možnostmi, pa je pomoč

sodobnih računalniških orodij nujna, če naj odločitev temelji na podlagi analiz ogromnih količin izvornih podatkov o poslovnih dogodkih.

Izbor ustrezne računalniške rešitve je možno predvideti z analizo vprašanja, prikazanega na sliki 1:

a) *Vprašanje*: Potrebno je zagotoviti sistem za razpoznavanje govora in pretvorbo govora v pisano besedo.
b) Potrebno je pripraviti rešitev, ki ni pogojena z znanjem ustreznega računalniškega jezika s strani uporabnika. Najboljša bi bila rešitev, ki bi se čimbolj približala naravnemu komuniciranju med ljudmi. *Rešitev* je torej ustrezen odgovor računalnika (grafični, tabelarični, govorni) *na podlagi vprašanja uporabnika v naravnem jeziku*.
c) *Od januarja 2000 do januarja 2001*: Strateška vprašanja ponavadi zajemajo daljša obdobja, lahko tudi več let. Hranjenje vseh podrobnih transakcijskih podatkov za tako dolgo obdobje bi bilo nesmiselno. Ti podatki so praviloma zanimivi do takrat, ko se neka poslovna transakcija odvija, kasneje pa so pomembni le glavni, zbirni podatki o transakciji. V smislu zagotovitve ustrezne rešitve bo potrebno zgraditi podatkovno skladišče, kjer se bodo shranjevali vsi pomembni podatki, ki zadevajo prodajo podjetja za obdobje več let.
d) *Količina in vrednost prodaje poslovne enote, podjetja, izdelki*: V mnogih primerih odločitev zahteva predhodno pridobljene odgovore na vprašanja, ki so zelo zapletena. Vprašanja vključujejo veliko kriterijev, iščejo se morebitne soodvisnosti med dogodki, zahtevajo se vrednostni in količinski rezultati v najrazličnejših merskih enotah. Vse to je s strani informatikov v podjetjih zelo težko zagotavljati in obvladovati s klasičnimi metodami filtriranja zapisov v datotečnih ali relacijskih zbirkah podatkov. Izdelati inventurno stanje gotovih izdelkov v skladišču je sicer zahtevno delo, vendar so izračuni natančno predpisani in se zelo redko spreminjajo. Vprašanja, kjer se iščejo neke zakonitosti tržnega obnašanja, pa se neprestano spreminjajo, nemogoče jih je vnaprej predvideti in definirati. Ta vprašanja so v osnovi večdimenzionalna, zato se za potrebe analiz v novejšem času vedno bolj uporabljajo tako imenovani OLAP sistemi, ki omogočajo enostavno in prožno gradnjo vprašanj kar s strani končnega uporabnika.
e) *Odgovor bi rad čimprej*: Kreiranje množice tabel agregatov, ki bi zagotavljale odziv sistema v kratkem času, bi ob milijonskih zapisih in lahko tudi več deset dimenzijah vodilo do tako imenovane eksplozije podatkov. OLAP sistemi omogočajo nadzor nad večdimenzijsko strukturo v smislu načina shranjevanja podatkov, kreiranja agregatov in drugih stvari, ki ob ustrezni strojni računalniški opremi zagotavljajo odziv sistema v nekaj sekundah ne glede na to, koliko je izvornih podatkov in koliko je število kriterijev in mer, ki jih vključujemo v vprašanja.

1.2. Koncept izgradnje rešitve

Koncept računalniške izgradnje rešitve se lahko deli na dva dela:

1.2.1. Gradnja rešitve na strežniški strani

Odvisno od zahtevnosti rešitve se izbira tudi vrsta in število strežnikov, na katerih bodo delovali posamezni sklopi aplikacije.
Če bo rešitev izvedena na osebnih računalnikih, je zaradi jasnosti, stabilnosti in hitrosti sistema smiselno uporabiti vsaj tri različne računalnike:

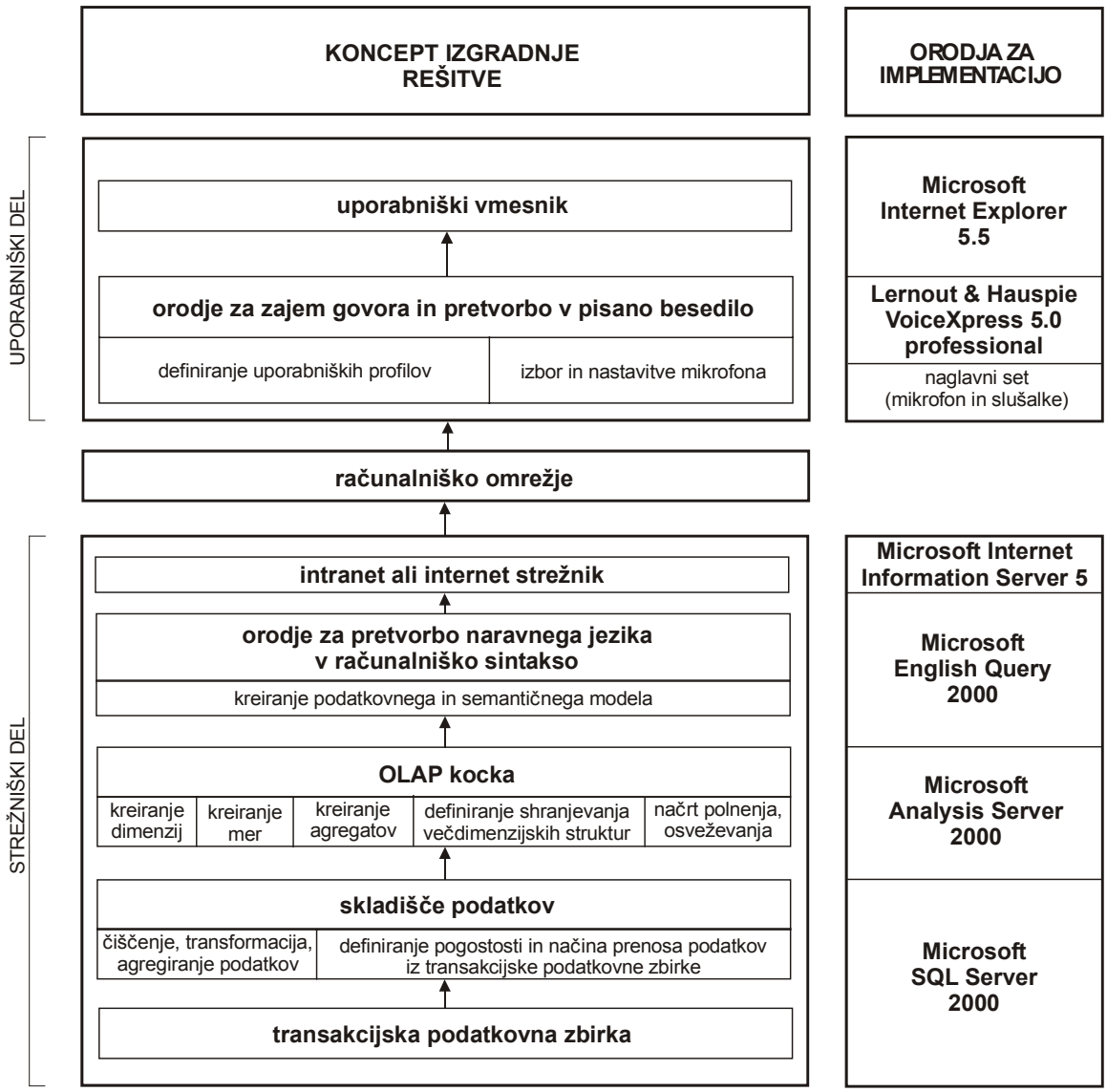
a) Na prvem strežniku bi se izvajala ustrezna replikacija izvornih podatkov različnih transakcijskih podatkovnih zbirk, ki se nahajajo na enem ali več računalnikih v skupno relacijsko podatkovno zbirko. S tem bi se zagotovil zbir vseh potrebnih podatkov na enem mestu. Izdelava kopije transakcijskih podatkov je pomembna tudi zato, da se ne izvajajo razne pretvorbe podatkov na operativnih podatkih in s tem preprečijo nastanek motenj v poslovnem procesu. Na tem strežniku bi se izvajali tudi ustrezni procesi pretvorbe podatkov v smislu izdelave želene vrste, količine in strukture podatkov v skladišče podatkov. V večini primerov je smiselno, da se tudi skladišče podatkov nahaja na istem računalniku, vendar kot baza podatkov ločena od kopije transakcijskih baz.
b) Drugi strežnik bi služil za izdelavo, preračunavanje in arhiviranje večdimenzijskih podatkovnih struktur - OLAP kock. V primeru enostavnih OLAP struktur bi isti računalnik lahko uporabili tudi za izdelavo sistema, ki bi bil sposoben vprašanje, zastavljeno v naravnem človeškem jeziku, pretvoriti v ustrezno računalniško sintakso, s pomočjo katere bi OLAP strežnik lahko poiskal ustrezno informacijo iz večdimenzijskih struktur. Sistem mora biti sposoben tudi obratnega opravila, to je pretvarjanja dobljenih podatkov v obliko, ki ustreza zahtevam končne aplikacije.
c) Tretji računalnik bi opravljal nalogo strežnika, ki bi na ustrezen način pretvoril podatke, dobljene iz OLAP kocke, v obliko, ki bi bila primerna želji končnega uporabnika. Lahko je to ustrezna "strežnik - odjemalec" aplikacija, ali pa neka intranet ali internet aplikacija, kjer bi računalnik opravljal vlogo spletnega strežnika, uporabniki pa bi do programa dostopali kar preko spletnega brskalnika. Lahko bi bil to celo strežnik, ki bi skrbel za ustrezno komunikacijo in pravilno dostavo informacij na razne prenosne naprave, kot so mobilni telefoni ali dlančniki.

1.2.2. Gradnja rešitve na uporabniški strani

Uporabniški del rešitve je potrebno gledati kot dva povezana sklopa:

a) Uporabnik mora imeti pred seboj program, kjer lahko enostavno izbira kriterije, ki jih bo vključil v svoje poizvedbe. Prav tako mora imeti možnost določanja načina prikaza odgovorov na zastavljena vprašanja v tabelarični, grafični, govorni ali kakšni drugi obliki. Kot zadnji (ali prvi) člen v verigi povezanih členov mora biti na strani uporabnika nameščen tudi ustrezen program za računalniško razpoznavanje človeškega govora. Nekateri programi, ki se danes pojavljajo na trgu, so že zelo kvalitetni, še vedno pa jih je potrebno dobro prilagoditi končnemu uporabniku. Potrebna je nastavitev mnogih parametrov, kot so: določanje hitrosti in glasnosti govora, višine glasu, prilagoditev na dolžino presledkov med besedami in druge fonemske posebnosti, ki so značilne za vsakega človeka. Pomembno je, da ima v primeru težav z razpoznavo govora, pri delu v hrupnem okolju in podobno, uporabnik še vedno možnost uporabiti druge načine (tipkovnica, miška) komuniciranja z računalnikom. Pričakovati je, da bodo programi vedno boljši, da se bo komuniciranje med človekom in računalnikom vedno bolj približevalo naravni komunikaciji, ki je značilna za sporazumevanje med ljudmi. To nenazadnje potrjujejo mnogi znani vpeljani sistemi za razpoznavanje govora, npr. v avtomobilih (Motor Trend News, 7.6.2001), v zdravstvu (Speech Technology, 2001), mobilni telefoniji (Dobler, 2000), (Lee, 31.5.2000), ali v zadnjem času tudi pri uporabi v žepnih računalnikih (IBM Corporation, 4. 10.2001), (Lernout & Hauspie Speech Products, 4.11.2001). V Sloveniji je poznan zelo dober program Govorec. To je program za sintezo slovenskega govora, ki so ga razvili v Skupini za inteligentne sisteme na Inštitutu Jožef Stefan (Gams, 2000). Ker je program kompatibilen z Microsoft Windows programi (z modulom Speech API 4.0), bi ga bilo možno uporabiti tudi v konkretnem modelu za pretvorbo odgovora računalnika v sintetizirani govor.

Slika 2: Koncept izgradnje rešitve in uporabljena orodja za implementacijo



Vir: avtor

1.3. Neodvisnost posameznih sklopov

Pri iskanju odgovora je pomembno dejstvo, da je vsaka dokončana vmesna stopnja lahko tudi samostojna rešitev sama po sebi. Tu ni govora o "sistemu črne škatle", kjer bi bilo zaradi prepletenosti komponent možno videti in uporabiti rešitev le na koncu gradnje. Taka prožnost je zelo pomembna zaradi možnih organizacijskih, finančnih in drugih težav, ki se lahko pojavijo med projektom. Projekt se lahko v katerikoli zaključeni vmesni fazi prekine in še vedno ostane rešitev, ki jo uporabnik lahko koristno uporabi v informacijskem procesu:

a) Analiza transakcijskih baz podatkov, ustvarjanje centralne baze podatkov, brisanje nepotrebnih baz podatkov, čiščenje zapisov, definiranje atributov, povezav med podatkovnimi strukturami in podobno, vse to je vedno dobrodošlo za prevetritev ustreznosti informacijskih sistemov.
b) Izgradnja skladišča podatkov predstavlja osnovo za izdelavo analiz, ki so lahko izdelana na najrazličnejše načine, z različnimi orodji. Za analizo podatkov iz skladišč podatkov ni pogoj uporaba OLAP orodij. Marsikdaj je mogoče celo zaslediti enačenje skladišča podatkov z OLAP-om, kar ni pravilno.
c) OLAP tehnologija je v svetu v velikem vzponu (slika 9, str. 27). Večdimenzijske podatkovne strukture so zelo blizu večdimenzijskim odnosom v poslovnem svetu, zato je uporaba OLAP kock zaenkrat ena izmed najboljših rešitev za izdelavo zapletenih analiz na podlagi skladišč podatkov. Danes v svetu obstaja mnogo uporabniških programov, ki so sposobni na uporabniku prijazen način prikazati podatke iz OLAP kock. Kocke, narejene z Microsoft Analysis Serverjem je tako možno pregledovati preko programov kot so: Microsoft Excel 2000, ProClarity Analytic Platform 4.0 (ProClarity, 2001) ali Seagate Info (Crystal Report, 2001).
d) Microsoft English Query 2000 omogoča pretvarjanje vprašanja, ki je podan v naravnem človeškem jeziku, v računalniško sintakso, kar omogoča kreiranje poizvedb po OLAP kockah na za človeka najenostavnejši možen način. Uporaba programa ni pogojena z izključno uporabo OLAP tehnologije, English Query aplikacijo je namreč možno izdelati tudi na podlagi različnih relacijskih baz podatkov.
e) Program za zajem, razpoznavanje in sintezo govora je ravno tako neodvisna komponenta, ki je namenjena pretvorbi človeškega govora v pisano obliko, ki nato služi kot vhodno vprašanje v English Query aplikacijo. Program Lernout & Hauspie Voice Xpress Professional 5.0 omogoča vključevanje razpoznavanja in sinteze govora v razne programe, ki delujejo v Microsoft Windows okolju.

2. SISTEMI ZA PODPORO ODLOČANJU

Danes se pojem *podpora odločanju* (Decision Support - DS) zelo pogosto uporablja na različnih področjih sprejemanja odločitev (Decision Making). Čeprav je na prvi pogled pojem zelo jasen, pa neke široko sprejete definicije ni.
Desetletje nazaj se je izraz *podpora odločanju* pojavljal v tesni zvezi z *operacijskimi raziskavami* (Operations Research - OR) in *analizo odločitev* (Decision Analysis - DA), pojavili so se različni *ekspertni sistemi* (Expert Systems - ES), v novejšem času pa je možno zaslediti mnogo *sistemov za podporo odločanju* (Decision Support Systems - DSS). V Sloveniji je o tem pojmu, orodjih in pričakovanjih razmišljal že v osemdesetih letih prof. dr.Janez Grad (Grad, A.Jenkins, 1987). Vse bolj popularen je tudi izraz *poslovna inteligenca* (Business Intelligence - BI). Včasih zelo uporabljen *direktorski informacijski sistem* (Executive Information Systems - EIS) se danes zelo redko uporablja.

Zmeda se pojavlja že pri kraticah. "DS" lahko pomeni "Decision Support", "Decision Systems", "Decision Solutions" ali celo "Decision Sciences". Podobno nejasnost je v zadnjem času možno zaslediti pri pojmu *poslovna inteligenca*, posebno pri kratici EBI ali e-BI, kjer lahko "e" pomeni "enterprise" ali "electronic".

Problem se pojavlja tudi pri samoopredelitvi dejavnosti posameznih konkurenčnih podjetij, kjer je opaziti, da se podjetja ne morejo dogovoriti glede izrazoslovja (slika 3).

Slika 3: Primer opredelitve dejavnosti nekaterih pomembnejših podjetij, ki se ukvarjajo z informacijsko tehnologijo

PODJETJE	OPIS DEJAVNOSTI
Adaytum	A leading provider of Web-based business planning solutions
Cognos	The world's largest and most successful business intelligence company
Oracle Express	A powerful, multidimensional OLAP analysis environment
SAS Institute	The leader in decision support and data warehousing, providing integrated enterprise information-delivery solutions and e-business solutions
WhiteLight Systems	The leading provider of next-generation analytic applications

Vir: The OLAP Report, 2001

Obstaja veliko različnih opredelitev izraza *podpora odločanju*. Dve osnovni definiciji sta:

- Pogled na podatke iz različnih pogledov kot pomoč pri sprejemanju odločitve (Data Warehouse forum, 16.6.1998).
- Podpora odločanju pomeni pomoč pri sprejemanju dobrih odločitev z razumevanjem učinkov vseh možnosti. Pomaga nam odgovoriti na vprašanje: "Kaj se bo zgodilo, če...?", za različne scenarije (Silsoe Research Institute, 2001).

Pomembno je torej pravilno razumeti bistvo izraza *podpora odločanju*, ki pomeni pomoč človeku, da iz množice možnosti lažje sprejme dobro odločitev. V praksi so namreč poznani različni sistemi, ki se na podlagi vnaprej vgrajenih pravil odločajo sami. To so lahko različne naprave, npr.: stikala v računalniških omrežjih, krmilniki, roboti, ali pa neki avtonomni računalniški programi.

Glede pogostosti povezav posameznih informacijskih področij s pojmom *podpora odločanju* je za trenutno obdobje možno reči, da se verjetno največkrat uporablja v povezavi s skladišči podatkov (Data Warehousing - DW) in OLAP kockami.

3. ORODJA ZA IZDELAVO REŠITVE

3.1. Strežnik Microsoft SQL Server 2000

3.1.1.Osnovne značilnosti

Microsoft SQL Server 2000 strežnik je Microsoftov sistem upravljanja relacijskih baz podatkov (RDBMS). Podatki so organizirani v dvodimenzionalne tabele. Ena tabela je sestavljena iz enega ali več stolpcev, ki predstavljajo atribute in ene ali več vrstic, ki predstavljajo dejanske vrednosti teh atributov. Več tabel je med seboj povezanih z relacijami. Operacije, ki jih izvaja uporabnik, so največkrat usmerjene v obdelavo relacij v smislu kreiranja novih tabel in pogledov, da se ugotovijo nove zakonitosti določene zbirke. Tok podatkov v tako podprtem informacijskem sistemu je običajno urejen na način "strežnik-odjemalec". To pomeni, da je zbirka podatkov ločena od uporabnika. Odjemalec s pomočjo uporabniškega vmesnika dostopa do strežnika, ta pa s pomočjo ustreznih servisov do fizičnih podatkov. Na strani odjemalca so lahko nameščeni različni programi, na strani strežnika pa centraliziran sistem zapisa in nadzora dostopa do podatkov.

Arhitekturo strežnika sestavlja več povezanih sklopov, ki delujejo kot celota:

3.1.1.1. Podatkovna arhitektura

Podatki v bazah so organizirani v logične komponente, ki so na pregleden način vidne uporabnikom. Baze so fizično implementirane kot dve ali več datotek na trdem disku. Uporabnik dela predvsem z logičnimi komponentami, kot so: tabele, pogledi, procedure, uporabniki. Z načini fizične implementacije se ponavadi ukvarja predvsem skrbnik baz podatkov.

Vsak strežnik vsebuje 4 sistemske baze:

- *master*: V njej so zapisane glavne sistemske informacije o strežniku. Beležijo se zagonske in ostale nastavitve za strežnik, uporabniki in njihove pravice, imena in zgradba uporabniških podatkovnih zbirk in podobne stvari, ki se tičejo podatkovnega strežnika. Je najpomembnejša sistemska baza, zato je za primer okvar vedno potrebno imeti na voljo svež arhiv *master* baze. Zanimivo je, da so te sistemske informacije ravno tako zapisane v relacijski obliki - v tabelah. Do teh podatkov je možno dostopati na enak način kot do uporabniških podatkov.

- *model*: Služi kot pripravljena osnova - predloga za kreiranje novih uporabniških baz. Vsaka nova podatkovna baza, ki jo uporabnik ustvari, je na začetku identična sistemski bazi *model* po velikosti, nastavitvah in ostalih lastnostih.

- *tempdb*: To bazo strežnik uporablja za shranjevanje začasnih tabel in procedur, ki so potrebne za uspešno procesiranje različno zahtevnih poizvedb po podatkovnih virih. Je globalna baza, v njej se arhivirajo začasni podatki za vse uporabnike. Ni je potrebno arhivirati, po ponovnem zagonu strežnika se izdela nova, prazna *tempdb* baza.

- *msdb*: Namenjena je za shranjevanje opravil, operaterjev, alarmov, obvestil in drugih nastavitev, ki so potrebna za nadziranje delovanja strežnika. Najpogostejša so časovno pogojena opravila za arhiviranje podatkovnih virov, za optimizacijo indeksiranih polj v tabelah, za različen uvoz, pretvorbo in izvoz podatkov, proženje in obvladovanje alarmov, ki se pojavljajo ob različnih scenarijih delovanja strežnika in nastavitve imen ter pravic operaterjev, ki skrbijo za vse skupaj.

V primeru, da se uporablja podvojevanje podatkov na druge strežnike, se ustvari tudi sistemska baza *distribution*, kjer se shranjujejo stvari, kot so: definicije "člankov" in "publikacij", ki predstavljajo izsek podatkov, ki jih je potrebno prenesti na ostale podatkovne strežnike, status sinhronizacije in vodenje zgodovine prenosov.

Na strežniku se poleg sistemskih nahaja tudi ena ali več uporabniških baz.

3.1.1.2. Administracijska arhitektura

Strežnik vsebuje niz orodij, ki omogočajo učinkovit nadzor nad delovanjem, samodejno odpravo napak in avtomatiziranje mnogih ponavljajočih se opravil, kar zelo olajša delo skrbnikom baz podatkov zaradi samodejnega upravljanja velikosti baz, nadzora nad dodeljevanjem in sproščanjem delovnega spomina, različnih grafičnih orodij za uvoz, izvoz in transformacijo podatkov, kreiranje baz, uporabnikov, zaščit in opravil (Knight, 21.5.2001).

3.1.1.3. Aplikacijska arhitektura

Aplikacije uporabljajo dve komponenti za dostop do baze podatkov:

a) Vmesnik uporabniškega programa (Application Programming Interface - API)

Definira način priklopa aplikacije na bazo podatkov in prenosa ukazov do baze. Objektni model API je večinoma neodvisen od programskega jezika, v katerem je napisana aplikacija in vključuje niz objektov, lastnosti in vmesnikov. Preko API vmesnika lahko aplikacija dostopa do podatkov in jih spreminja. Prav tako lahko z različnimi ukazi kreira in spreminja različne objekte, odvisno od pravic, ki jih ima posamezen uporabnik aplikacije.

SQL Strežnik podpira različne vmesnike:

- ActiveX Data Objects (ADO)
- OLE DB
- Open Database Connectivity (ODBC) in objektni API, zgrajeni preko ODBC
- Remote Data Objects (RDO) in Data Access Objects (DAO)
- Embedded SQL for C (ESQL)

b) Enolični lokator vira (Uniform Resource Locator - URL)

Je niz znakov, ki ga neka internet aplikacija uporablja za dostop do virov na internetu ali intranetu. Microsoft SQL Server 2000 vsebuje dinamično knjižnico (Dynamic-Link Library - DLL), ki jo aplikacije, ki delujejo na strežniku Microsoft Internet Information Services (IIS), uporabljajo za kreiranje URL naslova, preko katerega dostopajo do podatkov na SQL strežniku.

Strežnik podpira dva jezika za izmenjavo podatkov:

a) Internetne aplikacije lahko uporabljajo XPath poizvedbe iz XML shem

Strežnik podpira del XPath programskega jezika, ki je definiran s strani združenja za definiranje standardov preko interneta (World Wide Web Consortium - W3C). XPath je grafični navigacijski jezik, ki zna delati s segmenti XML dokumentov (Henderson, 21.12.2001). Najprej se s pomočjo XML Data Reduced - XDR jezika kreira načrt za definiranje XML pogleda podatkov iz ene ali več relacijskih baz podatkov. Nato se lahko uporabijo Xpath poizvedbe za pridobivanje podatkov iz izdelanih shem. Xpath poizvedbe se uporabljajo v URL ali ADO API, podpirajo pa tudi OLE DB API.

b) Transact-SQL jezik

Transakcijski strukturirani poizvedovalni jezik (Transact SQL - T-SQL) je programski jezik, ki je namenjen za programiranje in kreiranje raznih pogledov iz zbirk podatkov. Organizaciji ANSI in ISO sta definirali standard za SQL. Glavni standard je bil definiran leta 1992, zato se govori tudi o ANSI SQL-92 jeziku. Obstajajo pa tudi različni dodatki, nove funkcionalnosti, ki nadgrajujejo standard iz leta 1992 (Henderson, 18.2.2000).
Vsi programi, ki komunicirajo z SQL strežnikom, uporabljajo T-SQL programski jezik, ne glede na vrsto uporabniškega vmesnika določene aplikacije.

Primeri uporabe, kjer se kreira T-SQL sintaksa:

- Programi narejeni s programskimi jeziki Microsoft Visual C++, Microsoft Visual Basic, Microsoft Visual J++, ki uporabljajo vmesnike uporabniškega programa, kot so: ADO, OLE DB in ODBC.
- Transact-SQL skripti, ki jih uporabljajo različna orodja, npr. *osql*.
- Internet strani, ki pridobivajo podatke iz baz podatkov na SQL strežniku.
- Distribuirane baze podatkov, ki so izdelane na podlagi podvojevanj, ali poizvedbe iz različnih distribuiranih, dislociranih baz podatkov.
- Skladišča podatkov, kjer se iz transakcijskih baz podatkov izluščijo pomembni podatki za izdelavo različnih analiz za podporo odločanju.

3.1.1.4. Arhitektura podvajanja

Arhitektura podvajanja zajema niz rešitev za kopiranje, distribucijo in transformacijo podatkov med različnimi podatkovnimi strežniki. Obstajajo učinkovite metode in opcije za različne modele podvajanja, nadzora in ostale prožnosti konsistentnega prenosa podatkov.

3.1.2. Skladišče podatkov

3.1.2.1. Značilnosti

Skladišče podatkov (Data Warehouse -DW) je zbirka podatkov, ki je posebej strukturirana za povpraševanje in analiziranje. Praviloma zajema zgodovinske podatke o poslovanju nekega podjetja.
Nasprotje temu so poznani operacijski sistemi, ki uporabljajo transakcijsko zbirko podatkov (Online Transaction Processing - OLTP). To so podatki o trenutno izvajajočih se poslovnih dogodkih, ki imajo krajšo življenjsko dobo.

Zakaj naj bi se neko podjetje odločilo za gradnjo podatkovnega skladišča? Zato, ker to prinaša strateške prednosti pred konkurenco. Prednosti so vidne na mnogih področjih (Microsoft Corporation, 15.7.2001):

a) Zmožnost dostopa do vseh pomembnih podatkov v podjetju. Velikokrat je problem priti do podatkov, ki nastanejo v posameznih službah v podjetju. Informacijski sistemi v podjetjih so velikokrat ločeni, razdrobljeni med oddelke in neskladni med seboj. S pomočjo ustreznih transformacij pa se v skladišče podatkov prenesejo vsi pomembni podatki na eno mesto, v enotno obliko.
b) Odkrivanje podvojenih opravil: Z analizo toka podatkov mnogokrat pride do ugotovitev o podvajanju izračunov, vodenja evidenc, poročil. Poznani so primeri, ko npr. kadrovska služba vodi povsem ločeno kadrovsko evidenco od evidence v finančni službi.

c) Konsistentnost podatkov iz različnih oddelkov v centralni sistem. Podatki so strukturirani tako, da so med seboj primerljivi.
d) Zmanjša se redundanca - podvojevanje podatkov in analiz. Spisek vrst izdelkov lahko npr. vodijo tako v prodaji kot nabavi, informatiki.
e) Odkrivajo se nove možnosti celovitega analiziranja poslovnih procesov, ki zaradi razdrobljenosti podatkovnih virov prej niso bile znane.
f) Zmanjševanje stroškov zaradi lažjih, enotnih in bolj učinkovitih načinov analiziranja podatkov.

Skladišče podatkov zagotavlja pomoč vodstveni strukturi pri odločanju (Inmon, 1993). Omogoča hitro in enostavno analizo prodaje, ugotavljanje trendov, določanje odstopanj, odkrivanje vzorcev obnašanja nekega sistema, odnosov med posameznimi dimenzijami. Uspešnost izgradnje nekega analitskega sistema, katerega jedro je dobro skladišče podatkov, je odvisna od sistematičnega pristopa k izgradnji. Faze izgradnje so lahko sledeče (Microsoft Corporation, 15.7.2001):

a) Določitev zunanjih faktorjev, ki vplivajo na podjetje (konkurenca, gibanje cen na trgu, lastnosti kupcev, vladni ukrepi, spremembe tehnologij izdelave in procedur).
b) Definiranje ciljev podjetja (povečevanje konkurenčnosti, iskanje tržnih niš, optimiranje procesov, načini zniževanja stroškov, povečevanje zadovoljstva delavcev.
c) Dogovori z vodstvom podjetja glede transparentnosti sistema, ugotavljanje potreb po analizah, ugotavljanje informacijskih lukenj.
d) Pogovori z informacijskimi strokovnjaki glede računalniške strukture v podjetju, kakšna je strojna in programska oprema, ugotavljanje neskladnosti med seboj, stanje računalniškega omrežja, ki bi vse računalnike povezoval med seboj, računalniška izobraženost zaposlenih.
e) Pregled dokumentacije o službah, procesih, izdelkih, informacijskih tokovih in podobno.
f) Analiza obstoječega stanja, ustvarjanje celotne slike, določanje poslovnih prioritet, to je, katero področje je najpomembnejše za izdelavo dela skladišča podatkov, ki bi po uvedbi lahko dalo največji učinek, definiranje povezav, soodvisnosti med tehničnimi, organizacijskimi in prodajnimi področji, določanje upravičenosti investicije.
g) Določanje plana poslovnih področij in prioritete priprav podatkov za vključitev v skupno skladišče podatkov.
h) Definiranje konkretnega terminskega, človeškega in stroškovnega plana izvedbe projekta.
i) Določanje udeležencev in vrste vlog, ki jih bodo igrali pri zajemanju, prenosu, arhiviranju in analiziranju podatkovnih skladišč.
j) Prvo preverjanje dobljenih rezultatov s strani vodstva podjetja, popravki.
k) Dograjevanje sistema do želenega rezultata.

Skladišča podatkov so med seboj lahko zelo različna, vsem pa so skupne naslednje značilnosti (Microsoft Press, december 1999):

a) Podatki so zbrani iz drugih virov (transakcijskih baz podatkov ali predhodno zgrajenih podatkovnih skladišč).
b) Podatki morajo biti prečiščeni in konsistentni, preden se zapišejo v skladišče podatkov.
c) Podatki so agregirani. Ponavadi ne vsebujejo toliko podrobnosti kot transakcijsko usmerjeni sistemi.
d) Podatki imajo daljšo življenjsko dobo. Transakcijski sistemi običajno shranjujejo podatke samo do zaključka neke transakcije, v skladiščih podatkov pa so podatki praviloma shranjeni za obdobje več let.
e) Podatki so shranjeni na način, ki omogoča primerno poizvedovanje in analizo.
f) Podatki so ponavadi namenjeni samo za branje.

3.1.2.2. Koga vključiti v gradnjo

Pri gradnji skladišča podatkov je potrebno biti zelo pozoren na izbor oseb, ki bodo vključene v projekt. Podatkovni tok je lahko zelo zapleten, podatki zapisani na različnih mestih, na različnih sistemih, ki med seboj niso združljivi, podatki so lahko podvojeni, neažurirani.

Za kvalitetno gradnjo skladišča je potrebno izbrati vsaj naslednje ljudi:

a) Naročnik, sponzor projekta, ki pozna podjetje. To je ponavadi vodstvo podjetja, ki najbolje pozna organizacijo in razpolaga z analizami in poročili, ki so zelo dobra osnova za kreiranje ciljev izgradnje skladišča podatkov.
b) Skrbnik baz podatkov (Database Administrator - DBA): Oseba, ki najbolje pozna podatkovni tok v podjetju. Pozna tudi kvaliteto podatkov, to je, kateri podatki so zastareli, kateri so ažurirani, kateri se nahajajo v centralni bazi podatkov in katere bo potrebno prenesti iz lokalnih podatkovnih virov.
c) Programer: Programer aplikacij v podjetju prav tako dobro pozna podatkovni tok, še posebno izvor posameznega podatka (kdo ga vnese v sistem) in načine preračunov, preden se podatki zapišejo v posamezno bazo podatkov.
d) Zunanji svetovalec: To mora biti oseba, ki ima veliko izkušenj pri izdelavi skladišč podatkov. Mora biti človek, ki lahko z razdalje vodi projekt in učinkovito usklajuje vse vključene komponente v projekt izgradnje. Sposoben mora biti misliti zelo široko in po potrebi znati vključiti tudi ostale strokovnjake, kot so: organizatorji dela, tehnologi, svetovalci za izgradnjo računalniškega omrežja, skrbniki varnosti podatkov, skrbniki spletnih strežnikov.

3.1.2.3. Podatkovni trg

Majhno skladišče podatkov ali izsek iz skladišča podatkov se imenuje podatkovni trg (Data Mart). Uporablja se kot zbir podatkov za nek funkcionalni del podjetja, za razliko od skladišča podatkov, ki se kreira na nivoju celega podjetja. Poizvedbe, ki se izvajajo na podlagi podatkovnih trgov, so hitrejše, ker je manj izvornih podatkov. Tipični primeri kreiranja podatkovnega trga so za prodajno področje, finančni oddelek, za potrebe uprave in podobno. Podatkovni trgi lahko delijo podatkovno skladišče tudi na geografski osnovi. Primer: Neko trgovsko podjetje lahko deli podatkovno skladišče na več regijskih centrov.

Poznani so odvisni in neodvisni podatkovni trgi:

a) Odvisni: Kljub različni funkcionalnosti podatkovnih trgov mora biti organizacija podatkov, shema in format podatkov, hierarhična zgradba dimenzij in način osveževanja enaka v vseh primerih. Le tako je možno uspešno iskati zakonitosti poslovnih procesov na nivoju celotnega podjetja preko skupnih dimenzij posameznih podatkovnih trgov. Posamezni oddelki v podjetju niso čisto ločeni eden od drugega, temveč obstajajo neke skupne dimenzije (npr. čas, kupec, poslovna enota), po katerih je možno iskati neke soodvisnosti.

b) Neodvisni: Kreirajo se za točno določen namen in samo za ta namen, brez potrebe, da se povezujejo z drugimi podatkovnimi trgi. Tak način je manj pogost, saj zaradi svoje specifičnosti zahteva posebno načrtovanje, kar lahko precej poveča stroške izgradnje.

3.1.3. Orodja za transformacijo podatkov

Potrebno je opraviti štiri nivoje preverjanja ustreznosti podatkov, preden se zapišejo v skladišče podatkov:

a) Preverjanje podatkov v transakcijskih bazah: Pojavljajo se različne nepravilnosti, kot so: neusklajene enote mere, podvajanje zapisov, podvajanje šifer, osiroteli zapisi. Če je le mogoče, se je najbolje lotiti nepravilnosti že na izvoru, to je v transakcijski bazi podatkov. Vse naknadne transformacije podatkov so praviloma veliko bolj zapletene.
b) Prenos podatkov v začasno bazo podatkov: V praksi se je pokazalo, da je najbolje najprej narediti prenos transakcijskih baz v skupno začasno bazo, kjer se nato izvajajo različne transformacije podatkov brez bojazni, da bi to vplivalo na izvorne transakcijske podatke. Tak prenos je včasih zelo zahteven, saj so baze podatkov lahko shranjene na različnih sistemih, npr. VAX RMS, dBase, Access, Oracle, Paradox, Microsoft SQL Server, Centura SQL Server, sistemi mnogokrat niso povezani v skupno računalniško omrežje in podobno.

c) Čiščenje podatkov: Zajema opravila, s katerimi je možno skrčiti bazo podatkov na tiste zapise, ki so res pomembni za uporabo v analitičnih sistemih. Taka opravila so:

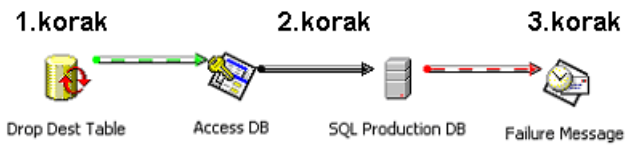
- Brisanje atributov, ki se pri strateških analizah ne potrebujejo, npr. različne podrobnosti o računu.
- Krčenje velikosti posameznih polj, še posebno v tabeli dejstev. Potrebno je izbrati tako velikost podatkovnih tipov, ki zasedejo čim manj prostora. Za primer si je možno ogledati Microsoft SQL strežnik in podatkovne tipe, ki jih podpira. Če se želi v neki koloni imeti zabeležene številčne vrednosti od 1 do 100, je primerneje uporabiti *tinyint* podatkovni tip, kot pa *int*. *Tinyint* tip je velikosti 1 byte, kar pomeni, da se s takim podatkovnim tipom lahko zapišejo števila od 0 do 255, kar v konkretnem primeru popolnoma zadostuje. Ker vsak *int* zapis zavzema 4 byte, je samo s pravilno izbiro podatkovnega tipa možno prihraniti 3 byte prostora v bazi podatkov za vsako vrstico v tabeli.
- Združevanje tabel: Povezovanje več tabel pri poizvedbah zahteva precej časa. Pri bolj enostavnih projektih je velikokrat bolje uporabiti manj tabel, tudi na račun nepopolne normalizacije.

d) Pretvorba podatkov: Zajema različna opravila, kot so: sprememba zapisov, združitev ali razgradnja zapisov, sprememba relacij med podatki, drugačna zgradba indeksov, preračunavanje določenih vrednosti, transformacije podatkovnih tipov, npr. iz številčnega tipa v tekstovni tip.

3.1.3.1. DTS orodja

Pri zagotavljanju ustreznih podatkov so zelo dobrodošla orodja za pretvorbo podatkov (Data Transforation Services - DTS), ki so vključena v strežnik Microsoft SQL Server 2000. Program, v katerem je možno v grafični obliki načrtovati ustrezne podatkovne vire, se imenuje DTS Designer.

Slika 4: Primer grafičnega prikaza uporabe DTS orodij

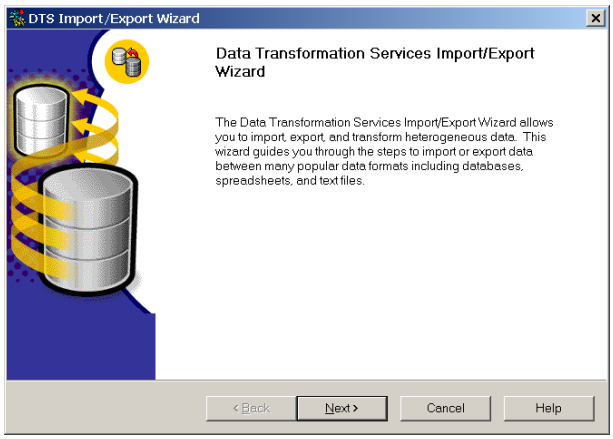


Vir: Ekranska slika, Microsoft SQL Server 2000, Books Online

Na sliki 4 je viden enostaven primer uporabe DTS orodij. Opravilo se samodejno izvaja v treh korakih:

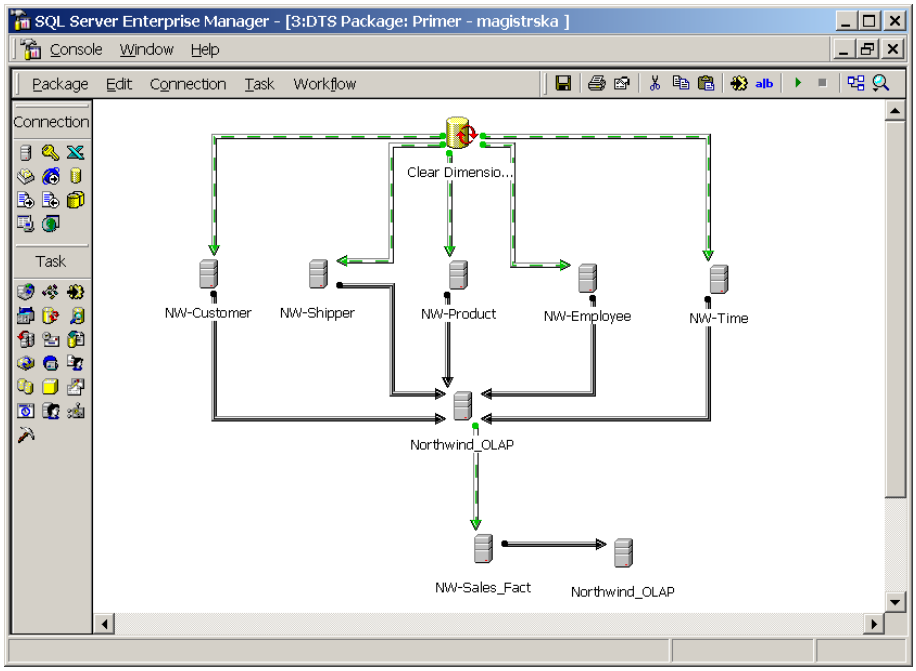
a) V prvem koraku se tvori povezava do ustrezne podatkovne baze na SQL strežniku in počisti vse zapise v ciljni tabeli.
b) Če je bilo brisanje zapisov uspešno, se bo izvedel drugi korak. To ponazarja zelena puščica, ki pomeni, da se naslednji korak lahko izvede le v primeru, če se je predhodno opravilo uspešno izvedlo. V drugem koraku se bodo podatki iz Microsoft Access baze prenesli v ustrezno tabelo v Microsoft SQL Server bazi. Črna puščica definira povezavo med izvorom in ciljem in ne določa nobenega pogoja za povezavo. Neko opravilo se bo torej izvedlo ne glede na končni rezultat.
c) Tretji korak se bo izvedel le v primeru, če bo kopiranje podatkov neuspešno. To je ponazorjeno z rdečo puščico. V tem primeru se bo preko elektronske pošte obvestil skrbnik baz podatkov.

Slika 5: Za enostavno uvažanje in izvažanje podatkov je na voljo tudi DTS "čarovnik":



Vir: Ekranska slika, Microsoft SQL Server 2000, DTS Export/Import Wizard

Slika 6: Grafični prikaz polnjenja skladišča podatkov iz transakcijskih podatkov



Vir: Ekranska slika, Microsoft SQL Server 2000 Enterprise Manager, Local Pacages

Na levi strani slike 6, ki predstavlja delovno okolje DTS Designerja, sta vidni dve glavni skupini komponent, ki se lahko vključita v diagram. Ena skupina zajema ikone, ki predstavljajo različne povezave do baz podatkov (Connection), druga skupina pa predstavlja opravila (Task). Ustrezno izbrane ikone se na diagramu lahko povežejo z različno obarvanimi puščicami in pravilno nastavijo posamezne lastnosti objektov. Tako pripravljen "transformacijski paket" opravil je za skrbnika baz podatkov pregleden in jasen.
Na sliki 6 je prikazan eden izmed možnih postopkov polnjenja podatkov iz transakcijskih baz podatkov v skladišče podatkov z uporabo DTS orodji:

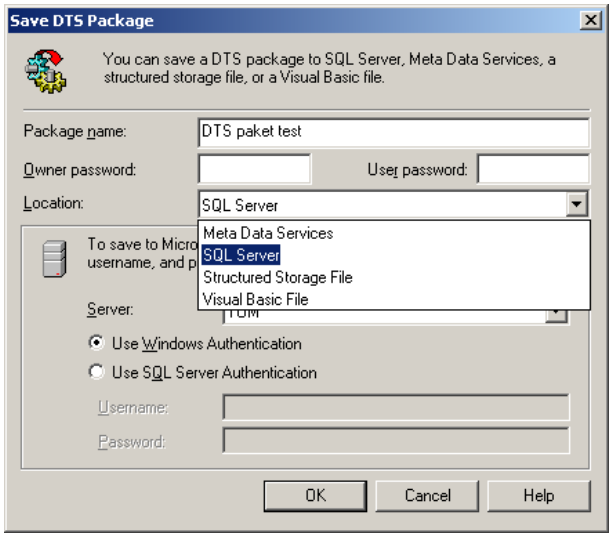
a) Najprej se izbrišejo vsi zapisi v petih ciljnih tabelah v skladišču podatkov, kar ponazarja ikona v obliki rumenega valja na vrhu diagrama. Primer torej ponazarja situacijo, kjer so že izdelane tabele z ustrezno strukturo v skladišču podatkov. V tem koraku se izbrišejo le dejanske vrednosti v tabelah, ime, struktura in ostale lastnosti tabele pa ostanejo.
b) Če predhodno opravilo uspe, se napolnijo ustrezne tabele - šifranti s podatki iz začasne baze podatkov, ki predstavlja kopijo transakcijske baze. Pogoj takega polnjenja je torej odvisen od predhodno uspešno izvedenega čiščenja tabel, kar je predstavljeno z zelenimi puščicami.
c) Le v primeru, če so bila vsa polnjenja izvedena, se izvede še polnjenje tabele dejstev. To nakazujejo črne puščice, ki kot pogoj za izvedbo neke akcije zahtevajo izvedbo predhodno definiranih opravil. Za razliko od zelene puščice, ki zahteva uspešnost izvedbe nekega definiranega opravila (On Success), pa je pri uporabi

črne puščice dovolj zaključek opravila (On Completition), ne glede na to, kakšen je bil rezultat nekega opravila.

Ko so postopki transformacije ustrezno pripravljeni, jih je kot paket možno takoj izvesti in tudi shraniti za kasnejšo ročno ali samodejno izvedbo. Tak paket je možno shraniti na različne načine (Chaffin, Knight, Robinson, oktober 2000):

a) V SQL strežnik, kjer se paketi beležijo v *msdb* bazi. Tak način omogoča shranjevanje paketov v katerikoli SQL strežnik v omrežju, omogoča kvaliteten nadzor nad vrstami, verzijami in izvajanjem paketov. Pri takem načinu shranjevanja je možno enostavno vključevanje različnih paketov in verzij posameznega paketa v verigo. Tak način je privzet, je najbolj običajen način shranjevanja DTS paketov.
b) V področje Meta Data Services: To je področje meta podatkov oziroma podatkov o podatkih, ki so shranjeni v posebnem skladišču (Repository). Pri tem načinu je omogočeno spremljanje zgodovinskih dejstev o podatkih, ki so bili uporabljeni pri izvedbi določenega DTS paketa. V določenih primerih je namreč pomembno, da skrbnik podatkovne zbirke lahko ugotovi (npr. pri ustvarjanju skladišča podatkov), katera izvedba določenega DTS paketa je "priskrbela" določen podatek.
c) V strukturirano arhivsko datoteko (Structured Storage File): Ta možnost omogoča shranjevanje paketov in struktur paketov, ki lahko vsebujejo več paketov in verzij paketov, v eno samo datoteko, ki jo je možno kopirati, premikati in brisati preko disket, priponk v elektronski pošti ali računalniškega omrežja, brez potrebe po predhodnem shranjevanju v SQL strežnik.
d) Kot Microsoft Visual Basic datoteko: Ta opcija shrani paket, ki je bil kreiran v DTS Designerju, spremeni v Visual Basic programsko kodo, ki jo je možno vključevati in spreminjati v razvijalskih orodjih.

Slika 7: Različni načini shranjevanja DTS paketov



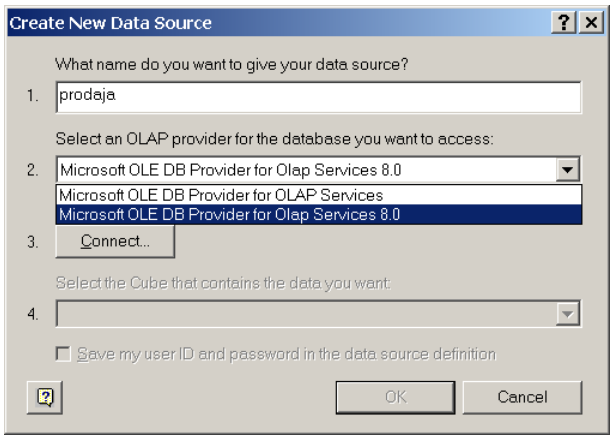
Vir: Ekranska slika, Microsoft SQL Server 2000 Enterprise Manager, DTS Designer

3.2. Microsoft SQL Server 2000 Analysis Services

Microsoft® SQL Server 2000 Analysis Services so orodja, ki se pri gradnji rešitev uvrščajo med podatkovnim in aplikacijskim strežnikom (Jacobson, 2000). Namenjena so kreiranju zapletenih analitičnih poizvedb (On-Line-Analytical-Processing - OLAP) in rudarjenju podatkov (Data Mining). Analysis Services vsebujejo strežnik Analysis Server, ki je zgrajen za namene upravljanja večdimenzijskih podatkovnih struktur in zagotavljanju hitrega dostopa uporabnikov do informacij iz večdimenzijskih struktur.
Analysis Services praviloma organizirajo podatke iz skladišč podatkov v OLAP kocke. Ob ustreznih predhodno izračunanih agregiranih vrednostih so uporabnikom sposobni zagotavljati hitre odgovore na zapletene analitične poizvedbe. Omogočajo tudi kreiranje modelov za rudarjenje podatkov tako na podlagi relacijskih kot večdimenzijskih baz podatkov. Taki modeli so primerni za iskanje določenih zakonitosti v podatkovnih strukturah, katerih uporabniki sami mogoče sploh niso poznali.

Na strani uporabnika obstajajo mnoga orodja, ki znajo uporabljati Microsoftov OLE DB gonilnik za OLAP 8.0. Ta orodja so sposobna črpati podatke iz Analysis Serverja in jih prikazati na uporabniku prijazen način. Možno je izdelati tudi lokalne kocke in izvajati poizvedbe brez stalne povezave do strežnika.

Slika 8: Primer definiranja povezave do OLAP kocke v programu Microsoft Excel 2000



Vir: Ekranska slika, Microsoft Excel 2000, "Čarovnik" za vrtilne tabele in vrtilne grafikone

Podobno kot pri upravljanju z Microsoft SQL strežnikom, tudi tu obstajajo zelo dobra grafična orodja za nadzor nad delovanjem Analysis Services, od "čarovnikov", urejevalnikov, do različnih orodij za shranjevanje podatkov, kreiranje uporabnikov, zaščit in optimizacijo delovanja.

3.2.1. OLAP

Je način organizacije velikih poslovnih zbirk podatkov. OLAP zbirke podatkov so zasnovane na način, ki ustreza posameznikovemu načinu analiziranja in upravljanja s podatki, tako da za potrebna poročila potrebuje čim manj časa in truda (Microsoft Corporation, 2000b).

Uspešni OLAP programi povečajo produktivnost vodstva podjetja, razvijalcev in cele organizacije. Vgrajena prožnost OLAP sistemov zagotavlja poslovnim uporabnikom večjo samostojnost. Vodje niso več odvisni od informatikov za kreiranje in dopolnitev računalniških programov, poročil, povezav. Še pomembneje je, da menedžerji lahko raziskujejo nek problem, ki bi ga bilo z manj učinkovitimi in prožnimi sistemi nemogoče modelirati. Hitri dostopi do želenih informacij pa v končni fazi pomenijo bolj učinkovit proces odločanja.

Pojem OLAP (On-Line-Analytical-Processing) je bil prvič definiran leta 1993 s strani dr. Teda Codda in njegovih sodelavcev, koncept pa je bil znan že mnogo prej. Opredelitev pojma OLAP so zapisali v raziskovalnem delu z naslovom: "Providing OLAP to User-Analysts: An IT Mandate".
Dr. Codd, sodelavec podjetja IBM, je znan tudi kot izumitelj modela relacijskih baz podatkov (McEwan, 2.7.2001). Bil je zelo cenjen raziskovalec baz podatkov med leti 1960 in 1980. V svojem raziskovalnem delu iz leta 1973 je opredelil dvanajst pravil za definiranje OLAP izdelkov, ki so nekoliko sporna, saj je bilo delo sponzorirano s strani podjetja Arbor Software Corporation (Groves, 26.1.2000).
Nekateri celo menijo, da je mnogo funkcij definiralo podjetje in ne dr. Codd. Tako se danes raziskovalno delo, ki je prvo definiralo pojem OLAP, smatra za publikacijo podjetja Arbor Software Corporation in ne neodvisno akademsko delo.

3.2.1.1. Dvanajst pravil za določanje OLAP izdelkov

Značilnosti, ki jih mora izpolnjevati OLAP aplikacija (Codd, 1993):

1. Večdimenzijski shematski prikaz

Uporabniški pogledi na zapletena dogajanja v podjetju so v osnovi večdimenzionalni. Zato naj bi OLAP baze podatkov omogočale večdimenzijski prikaz podatkov na prožen način, od najrazličnejših prikazov v vrtilnih tabelah, pogledov z več strani, z enostavnim segmentiranjem in agregiranjem podatkov.
Nekateri OLAP proizvajalci so bili skeptični do tega pravila, saj so menili, da so take večparametrske predstavitve lahko izvedene tudi brez potrebe po večdimenzijskem arhiviranju podatkov.

2. Odprtost

Uporabnikom končnih aplikacij ni potrebno vedeti, kakšen sistem stoji v ozadju. Zgradba OLAP sistemov mora biti povsem odprta, kar pomeni, da jo uporabniki lahko enostavno uporabljajo v različnih tabelaričnih ali grafičnih programih, povsod, kjer je zahtevana analiza podatkov, ne da bi morali vedeti, ali je v ozadju OLAP sistem, kje se nahaja izvor podatkov in podobno.
Tudi to pravilo so si proizvajalci večkrat razlagali po svoje. Celo testiranje izdelkov Essbase in kasneje TM/1 s strani podjetja, ki je sponzoriralo dr. Coddovo definiranje OLAP pravil, je le delno izpolnjevalo drugo pravilo, saj izdelka nista zagotavljala neposrednega dostopa do heterogenih zunanjih podatkovnih virov.

3. Dostopnost

Orodje, ki ga uporablja uporabnik, mora imeti možnost dostopa do najprimernejšega modela podatkov, ki bo zagotavljal ustrezen odgovor na določeno poizvedbo. OLAP sistem mora biti tak, da uporabniku zagotavlja le trenutno želene podatke na način, ki je uporabniku prijazen, in ne prikaz vseh mogočih podatkov, ki so za trenutno poizvedbo nepomembni.

4. Nespremenjena odzivnost poročanja

Če se poveča število dimenzij in nivojev agregiranja, uporabnik tega ne sme občutiti. Odzivnost sistema mora biti enaka. OLAP sistem mora biti toliko robusten, da se je sposoben prilagoditi na različne spremembe v organizacijskih strukturah.
Občutna sprememba pri odzivnosti sistema ob povečanju dimenzij bi pri uporabniku lahko pomenila spremembo v načinu razmišljanja. Pri odločitvah bi lahko začel poenostavljati probleme in zmanjševati število odvisnih faktorjev pri analiziranju problemov.

5. Zgradba "strežnik - odjemalec"

OLAP orodja morajo biti sposobna delovati v "strežnik-odjemalec" okolju. Ne samo to, različne uporabniške aplikacije naj bi imele možnost enostavnega dostopa do OLAP strežnika.
To je verjetno eno izmed najtrših pravil dr. Codda, saj le redki izdelki, ki so do danes prišli na trg, zadostujejo temu pogoju. Verjetno je dr. Codd, ne da bi uporabljal ta izraz, mislil na dostop preko interneta, ali pa na vedno bolj uveljavljen "OLE DB for OLAP" standard.

6. Enakovredna dimenzionalnost

Vsaka podatkovna dimenzija mora biti enakovredna v strukturi in računskih zmožnostih. Pri kreiranju poročil in enačb morajo vse dimenzije imeti enako težo.

To je mogoče najbolj sporno izmed vseh dvanajstih pravil. Danes je običajno, da je v primeru uporabe OLAP orodij v več aplikacijah za splošne namene možno uporabiti orodje, ki izpolnjuje to pravilo, za posebne primere pa se uporablja orodje, ki omogoča spreminjanje osnovne strukture, izračunov in dodatne operacijske zmožnosti posameznim dimenzijam.

7. Dinamično upravljanje z redkimi matrikami

Običajen večdimenzijski model pogosto vsebuje več milijonov celic, od katerih so mnoge vedno prazne in so lahko zelo različno razporejene v matrikah, ki jih tvorijo dimenzije. OLAP sistem mora znati graditi optimalen model glede porabe delovnega spomina in prostora na disku.

8. Večuporabniška podpora

OLAP orodja morajo podpirati in spodbujati skupinsko delo, izmenjavo idej in analiz med uporabniki, zato je pomembno, da je zagotovljena večuporabniška podpora.

9. Neomejene meddimenzijske operacije

Pravila za način prikaza podatkov na kateremkoli nivoju v hierarhiji morajo biti vedno definirana, morajo vedno veljati, ne glede na to, kateri del podatkov je izbran.

10. Intuitivno upravljanje s podatki

Uporabniški vmesnik mora biti zgrajen tako, da omogoča neposredno spreminjanje prikaza podatkov.
Dr. Codd pravi, da mora biti spreminjanje pogleda podatkov tako, da za navigacijo po pregledih ni potrebna uporaba različnih menijev ali večkratno "sprehajanje" preko ekrana računalnika. Današnji programi po večini izpolnjujejo ta pogoj z uporabo dvojnega klika miške na določeno celico in ukaza "povleci - spusti".
To pravilo se marsikomu ne zdi tako pomembno za določanje primernosti OLAP orodij. Končni uporabniki so si med seboj zelo različni, zato je težko reči, kateri vmesnik je najprimernejši.

11. Prilagodljivo poročanje

OLAP orodje mora uporabniku omogočiti izdelavo najrazličnejših prikazov podatkov na način, ki ga želi in potrebuje.

12. Neomejeno število dimenzijskih in agregatnih nivojev

OLAP orodja ne smejo biti omejena s številom dimenzij, ki jih je možno vključiti v model.

Leta 1995 je dr. Ted Codd dodal še šest pravil in vse skupaj uvrstil v štiri skupine:

a) osnovne lastnosti
b) posebne lastnosti
c) pregledne lastnosti
d) dimenzijski nadzor

Zmede in nejasnosti pa s tem še ni bilo konec. Večina ljudi si težko zapomni dvanajst oziroma osemnajst pravil, še težje se je po njih ravnati in razločevati razlike, kar se vidi tudi pri poimenovanju podjetij (glej sliko 3, na str. 7). Nekatera podjetja so se tako promovirala kot izdelovalci OLAP izdelkov, pa to niso bila, nekatera podjetja so imela celo programe z značilnostmi OLAP izdelkov, ampak vgrajene v drugače poimenovane izdelke. Ravno tako v združenju proizvajalcev OLAP izdelkov (OLAP Council, 1999) še zdaleč niso bila vključena vsa glavna OLAP podjetja, ki so bila prisotna na trgu.

Leta 1995 sta se gospoda Nigel Pendse in Richard Creeth povezala z neodvisnimi raziskovalci iz podjetja Business Intelligence (Business Intelligence, 2001). Hoteli so definirati neka nova, vendar enostavna pravila za določanje OLAP izdelkov, ki bi bila res neodvisna od proizvajalcev. Rezultat tega je takoimenovani FASMI (Fast Analysis of Shared Multidimensional Information) test, ki je od leta 1995 ostal nespremenjen in velja še danes.

3.2.1.2. FASMI test

Značilnosti FASMI testa (Pendse, 2001):

a) Hitrost (Fast)

Hitrost pomeni, da mora OLAP sistem vrniti uporabniku večino rezultatov v času do petih sekund. Pri tem poizvedbe pri enostavnih analizah ne smejo trajati dalj kot sekundo in le redke, kompleksne poizvedbe lahko trajajo več kot dvajset sekund. Neodvisna raziskava, opravljena na Nizozemskem, je pokazala, da končni uporabniki po tridesetih sekundah razumejo poizvedbo kot neveljavno in pritisnejo *Ctrl+Alt+Delete* tipke za končanje neuspešne poizvedbe, če se na ekranu posebej ne izpiše opozorilo, da bo poizvedba trajala dalj časa. Vendar tudi v primeru, da se na ekranu izpiše opozorilo o daljšem čakanju, uporabniki zgubijo koncentracijo in potrpljenje, kar lahko vpliva na kvaliteto odločanja.
Tako hitrost je ob veliki količini podatkov težko zagotavljati, zato se proizvajalci zatekajo k različnim metodam pospešitev, od posebne strojne opreme, posebno strukturiranih agregatov, do različnih načinov shranjevanja podatkov.

b) Analiza (Analysis)

Pomeni, da mora biti OLAP sistem sposoben zajeti kakršnokoli poslovno logiko in statistično analizo, ki jo zahteva nek uporabnik ali aplikacija, pri tem pa ostati čimbolj enostaven za uporabnika. Čeprav je potrebno sprogramirati določene funkcionalnosti neke analize, morajo biti OLAP sistemi narejeni tako, da različne analize lahko nastavi uporabnik sam, brez potrebe po znanju računalniških programskih jezikov. Nekateri izdelki, npr. Oracle Discoverer so "padli" ravno pri tej zahtevi.

c) Skupna raba (Shared)

OLAP sistem mora vsebovati vse potrebne varnostne mehanizme za skupinsko analitično delo, še posebno, če je več uporabnikom omogočeno popravljanje podatkov za namene "kaj-če" analiz.
Veliko programov je zgrajenih pomanjkljivo, ker so proizvajalci predvidevali, da bodo podatki v OLAP sistemih namenjeni le za branje, zato so vgrajevali le enostavne načine zaščite. Celo zelo dodelani večuporabniški programi, ki omogočajo povratno pisanje v OLAP kocke, imajo še dokaj okoren sistem zaščite. Tak je npr. tudi Microsoft SQL Server 2000 Analysis Services.

d) Večdimenzionalnost (Multidimensional)

To je osnovna zahteva za OLAP sisteme. Če bi zahtevali opis OLAP sistemov z eno besedo, bi bila to večdimenzionalnost. Sistem mora zagotavljati večdimenzijsko poizvedbo po podatkih, vključno s podporo hierarhične ureditve (vrtanje v globino), kar je najbolj logičen način analiziranja poslovnih procesov in organizacije.
Ne predpisuje se natančno število dimenzij, ki jih mora sistem podpirati, ker so analize grajene za zelo različne namene. Pravilo je, da mora OLAP sistem zagotavljati minimalno toliko dimenzij, da je zagotovljen popoln večdimenzijski konceptualni pregled podatkov.

e) Informacije (Information)

Upošteva se vse izvorne podatke in vse pridobljene informacije z neko analizo. Pri tem pogoju se meri zmožnost posameznega izdelka v smislu količine izvornih podatkov, ki jo je sposoben obdelati, in ne kapaciteta podatkov, ki jo je sposoben shraniti.
Tu so lahko razlike zelo velike. Največji OLAP izdelki so sposobni obdelati vsaj tisočkrat več izvornih podatkov kot majhni. Pri ugotavljanju te zahteve se upoštevajo različne stvari, kot so: podvajanje podatkov, potreben spomin (RAM), izkoristek trdega diska, učinek - storilnost, integracija s skladišči podatkov in podobno.

3.2.1.3. Kdaj uporabiti OLAP

Tipična področja uporabe OLAP aplikacij (OLAP Council, 1997):

a) finančno modeliranje (priprava proračuna, planiranje)
b) napoved prodaje
c) analiza dobičkonosnosti kupcev in izdelkov
d) prikaz izjem
e) razporeditev virov in načrtovanje kapacitet
f) analiza sprememb
g) planiranje napredka (razlike med planom in realizacijo)
h) analiza tržnih deležev
i) nadzor kakovost

Nekaj primerov zapletenih vprašaj, ki se pogosto porajajo pri strateških analizah in na katere je možno dobiti ustrezen odgovor ob pomoči OLAP orodij:

a) Kakšna je bila količina prodaje jabolk na Štajerskem v prvi polovici jeseni ?
b) Kateri izdelki so bili v lanskem letu na novo prevzeti v skladišče s strani dobavitelja ABC, ločeno po poslovnih enotah ?
c) Naštej mi prvih 10 delavcev, ki smo jih v letu 1997 zaposlili v novi proizvodni enoti.
d) Kakšni so bili stroški oglaševanja izdelka S345 od leta 1998 do danes, v odvisnosti od medija v katerem smo oglaševali ?

Značilnosti podatkov v OLAP sistemih:

a) Podatki so konsolidirani: Podatki so zbrani z vseh lokacij podjetja na enem mestu.
b) Podatki so konsistentni: Vsi uporabniki dobijo enak rezultat na enako zastavljeno vprašanje. To je zelo pomembno, ker se v podjetjih večkrat dogaja, da posamezne službe pri preračunavanju podatkov uporabljajo različne računalniške programe. Veliko analiz se dandanes izvaja s programi za delo z razpredelnicami, npr. z Microsoft Excelom, kjer različni uporabniki pri analizi izvornih podatkov uporabljajo različne enačbe, različno zaokrožujejo posamezne rezultate, zato so končne vrednosti enakih analiz velikokrat precej različne.
c) Podatki so zgodovinski: Podatki v OLAP sistemih predstavljajo zgodovinska in ne trenutna dogajanja.
d) Podatki so praviloma namenjeni samo za branje, razen v nekaterih posebnih primerih ("kaj-če" analize).
e) Podatki so subjektno orientirani: Podatki v posamezni kocki morajo biti prečiščeni, brez balasta, točno je potrebno vedeti, katere odgovore se želi dobiti z gradnjo določene kocke, vključujejo se le pomembni podatki, ki pomagajo vodstveni strukturi pri odločanju.

Prenos podatkov iz OLAP zbirk v uporabniške programe:

OLAP zbirke podatkov so zasnovane tako, da pospešijo pridobivanje podatkov. Povzete vrednosti praviloma namesto odjemalca (npr. Microsoft Excel ali ProClarity Info) izračuna OLAP strežnik, zato se, kadar se ustvarja ali spreminja poročilo, po računalniškem omrežju pošlje manj podatkov. Tak pristop omogoča lažje upravljanje z večjimi količinami izvornih podatkov.

Kdaj graditi OLAP sistem ?

OLAP sistem je primeren:

a) Če je organizacija kompleksna.
b) Če je na voljo dovolj dobrih, zanesljivih, verodostojnih podatkov.
c) Kjer vlada velika dinamika dela.
d) Kjer se organizacija poslovanja hitro spreminja.

OLAP ni primeren:

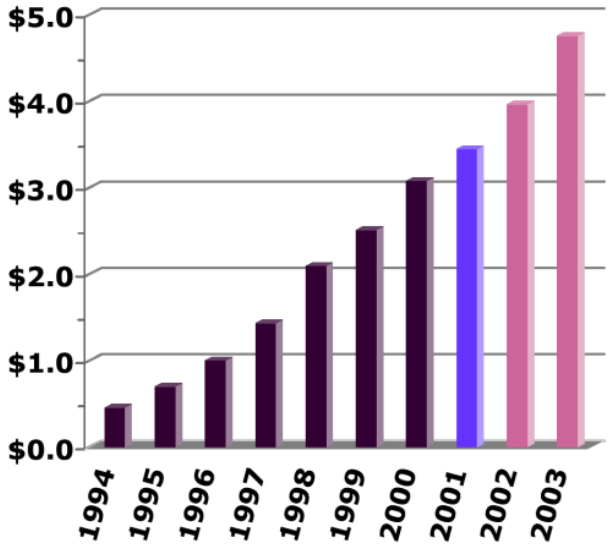
a) Za podporo produkcijskim, operativnim procesom.
b) Za poročila, ki vsebujejo malo različnih dimenzij in malo količino podatkov.

3.2.1.4. Analiza tržnega deleža OLAP sistemov

Predvideva se, da je svetovni delež OLAP sistemov v letu 1996 zavzemal okoli 1 milijardo dolarjev, 1,4 milijarde dolarjev v letu 1997, nekaj čez 2 milijardi dolarjev leta 1998, 2,5 milijarde dolarjev leta 1999 in nekaj čez 3 milijarde dolarjev v letu 2000.

Analizo je izvedel g. Nidgel Pendse s sodelavci (Pendse, 2001). Rezultati niso podani na podlagi števila aktivnih uporabnikov ali števila podjetij, ki se ukvarjajo s to tehnologijo, rezultati in napovedi temeljijo na podatkih o stroških končnih uporabnikov, ki nastanejo zaradi nakupa OLAP rešitev.

Slika 9: Analiza rasti vrednosti OLAP svetovnega trga in napovedi



Vir: Pendse, 2001

Uporabljen je bil FASMI test za ugotavljanje ustreznosti produktov kot OLAP sistemov, ki so jih vključili v raziskavo. Vseeno pa je točne vrednosti težko določiti. Za to je več razlogov. Glavna sta dva:

a) Veliko proizvajalcev programske opreme ima OLAP rešitve vgrajene v večje programske sklope in možno je, da kljub temu, da imajo OLAP tehnologijo nameščeno, uporabniki teh programov ne uporabljajo. Tako določeni proizvajalci ne morejo ali nočejo izdati informacij o uporabi OLAP tehnologije. Tudi Microsoft ne prodaja OLAPa v obliki samostojnega izdelka, ampak se Analysis Services dobi kar skupaj z nakupom SQL strežnika. Prav tako z Excelom 2000 uporabnik dobi tudi odjemalsko orodje za pregledovanje OLAP kock v obliki večdimenzijskih vrtilnih tabel.
b) Nekateri proizvajalci prodajajo svoje izdelke neposredno uporabnikom, drugi pa le posrednikom ali podjetjem, ki nato nadgrajujejo osnovne module. Zato je težko vrednotiti vse proizvajalce enakovredno.

V letu 2001 je bila predvidena nekoliko počasnejša rast iz več razlogov:

a) Težko je vsako leto dosegati ogromna povečanja na trgu najrazličnejših informacijskih podjetij in rešitev. V zadnjih letih je namreč na trg prišlo veliko novih "igralcev", zato se na nekaterih področjih že pojavljajo znaki zasičenosti z različnimi izdelki za podporo odločanju. Sedaj je namreč na trgu vsaj 30 velikih podjetij, ki za sebe trdijo, da so ponudniki OLAP rešitev.
b) Konec leta 1998 je na OLAP trg prišel tudi Microsoft, ki je povzročil izredno znižanje cen OLAP produktov, kar pomeni vrednostno znižanje rasti, čeprav

število namestitev še vedno strmo raste. OLAP tehnologija je s pojavom Microsofta tako postala dostopna tudi manjšim podjetjem.

c) Vseeno obstajajo ocene, da bi se do leta 2003 OLAP trg lahko približal vrednosti 5 milijard dolarjev.

Slika 10: Tržni delež petih največjih proizvajalcev OLAP produktov v letu 2000

PODJETJE	TRŽNI DELEŽ (%)
Hyperion Solutions	22,3%
Cognos	12,6%
Microsoft	11,7%
Oracle	10,1%
MicroStrategy	9,2%

Vir: Pendse, 2001

3.2.1.5. OLAP in Microsoft

Obstajajo ocene (Pendse, 2001), da je bilo v svetu do leta 2000 kupljenih okoli 35.000 Microsoft OLAP strežnikov s 600.000 odjemalci, vendar vsi strežniki niso bili implementirani. Do leta 2001 naj bi bila ta številka še dvakrat do trikrat višja. Taka rast prodaje OLAP strežnikov je sigurno posledica dejstva, da je Microsoft ob izidu v paket strežnika SQL Server 7.0 vključil tudi OLAP strežnik, brez dodatnih stroškov za uporabnika. S tem je drage in ekskluzivne OLAP sisteme s strani proizvajalcev, kot so npr. Essbase, MicroStrategy ali iTM1, približal širokim množicam. Danes je možno reči, da velika večina uporablja OLAP rešitve samo zaradi izredno ugodne cene, ki jo je ponudil Microsoft.

Ker se OLAP strežnika ni dalo kupiti kot samostojen izdelek, ampak le skupaj v paketu z SQL Serverjem 7.0, je bilo težko ugotoviti dejansko uporabo OLAP sistemov. Zelo posrečeno se zdi ugotavljanje uporabe na podlagi števila prenosov programskih popravkov za OLAP strežnik z Microsoftove spletne strani. Leta 2000 je namreč podjetje izdalo drugo verzijo popravkov "Service Pack - SP2" za SQL Server 7.0. Datoteka s popravki za SQL strežnik je bila namerno ločena od popravkov za OLAP strežnik. Upravičena so razmišljanja, da je 64.4 MB veliko datoteko preko interneta prenesel le tisti, ki je te popravke res potreboval, torej tisti, ki je imel tudi dejansko implementirano OLAP rešitev. V prvem mesecu je bilo kar 135.000 poizkusov prenosa datoteke s popravki. Tudi če je bilo vmes nekaj neuspešnih, je ocenjeno število delujočih Microsoftovih OLAP strežnikov večje od vseh OLAP strežnikov drugih proizvajalcev skupaj. Potrebno je računati, da je marsikatero podjetje datoteko s popravki preneslo z interneta le enkrat, čeprav imajo mnoga podjetja realiziranih več namestitev OLAP strežnikov. Tak uspeh so Microsoft OLAP Services dosegli v pičlih 15 mesecih od izdaje.

Predvidevanja o več deset tisoč inštalacijah pa niso govorila ničesar o vrsti aplikacij, ki delujejo na podlagi OLAP tehnologije. Ocenjuje se, da je velika večina podjetij še vedno le "tipala" novo tehnologijo in da so imela le redka podjetja vpeljan res delujoč sistem v svoj poslovni proces.
Leta 2000 je Microsoft skupaj s strežnikom SQL Server 2000 izdal tudi nadgrajeno OLAP razvojno okolje, ki so ga poimenovali Analysis Services. Glede na prejšnjo različico je bilo dopolnjenih in popravljenih veliko stvari, kot so: večja hitrost sistema, bolj prožno upravljanje s hierarhičnimi strukturami, večje število dimenzij, izboljšana zaščita, dopolnjena internetna podpora in drugo. Z izboljšavami je strežnik postal primeren za precej širši krog poslovnih področij, vključno z raznimi finančnimi analizami.
Ena izmed večjih pomanjkljivosti Analysis Services so mogoče le zelo osnovna navodila za delo in premalo učnih primerov za prikaz uporabe OLAP tehnologije na različnih področjih. Ta del je tako prepuščen drugim podjetjem, kot je npr. OLAP Train (OLAP Train/Aspirity, 2001). Na voljo je tudi mnogo knjig neodvisnih piscev, kot sta npr.: (Jacobson, 2000) in (Spofford, 2001).

3.2.1.6. Primerjava med OLTP in OLAP bazami podatkov

Slika 11: Primerjava med OLTP in OLAP bazami podatkov

	OLTP	OLAP
sprememba HW, SW	redko	pogosto
poročila	fiksna	se spreminjajo
ažurni podatki	da	ne (zakasnjeni)
zgodovina	kratka	dolga
interakcija uporabnika	transakcija	celotna zbirka
obremenitev HW	stalna	različna
tip podatkov	detajlni	agregati
tip uporabnikov	obračunovalke	vodstvo

Vir: Microsoft Press, december 1999

3.2.1.7. Projekt T^3

7. februarja 2001 so štiri pomembna podjetja iz sveta informacijske tehnologije (Microsoft, Unisys, EMC in Knosys) prvič predstavila informacije o projektu T^3 (Microsoft Corporation, 2000), kjer so dokazali, da Microsoft Analysis Services, ki delujejo na Windows 2000 operacijskem sistemu, lahko zgradijo, ažurirajo in izvajajo različne poizvedbe na MOLAP kockah, ki so mnogo večje, kot so do sedaj mislili, da je možno.

V ta namen so uporabili podatke v relacijski zbirki velikosti 1,2 TB. 8 tabel dejstev je vsebovalo 7,7 milijarde vrstic podatkov o prodaji več kot 700.000 izdelkov v obdobju 5 let.

Uporabili so Unisysov ES7000 računalnik, na katerem so definirali 2 navidezna strežnika. Na enem je bila relacijska zbirka - skladišče podatkov na Microsoft SQL Server 2000 platformi, velika 1,2 TB (brez indeksov), na drugem strežniku pa Microsoft Analysis Services, kjer so zgradili OLAP kocko na podlagi podatkov iz relacijske zbirke.

Pri kreiranju kocke so uporabili tri osnovne dimenzije: izdelek, trg in obdobje. Na podlagi 8 tabel dejstev so zgradili 8 kock, ki so jih združili v eno navidezno kocko.
Uporabljena MOLAP tehnologija pomeni, da tabel ni bilo potrebno indeksirati, niti ni bilo potrebno kreirati dodatnih zbirnih tabel. Vsi izvorni podatki, agregati in interno večdimenzijsko indeksiranje je zasedlo le 471 GB, kar je le 39% velikosti izvornih podatkov relacijske zbirke, pa tam še ni bilo kreiranih indeksov in agregatov. Pokazalo se je, da je taka večdimenzijska struktura zelo optimirana za shranjevanje in daleč od bojazni, da bi lahko prišlo do pojava eksplozije podatkov. Če bi bil projekt izveden na podlagi ROLAP tehnologije, bi bila zbirka podatkov verjetno velika med 3 in 4 TB.

Glede ugotavljanja performanc sta pomembni dve stvari:

a) Koliko časa se procesira kocka: Ves prenos 7,7 milijarde vrstic v Analysis Services iz skladišča podatkov, kreiranje agregatov, indeksiranje in kreiranje ustreznega načina shranjevanja je trajalo 50 ur in 18 minut, kar pomeni procesiranje 153 milijonov vrstic na uro ali 42,4 tisoč vrstic na sekundo. 5 particij se je procesiralo ločeno na 16 procesorjih (60-70% CPU utilizacija)
b) Hitrost izvršitve poizvedbe: simulirali so delo 50 uporabnikov, od katerih naj bi vsak izvršil po 27 različnih poizvedb s povprečnim "razmišljanjem" 30 sekund med poizvedbami. To pomeni 1350 različnih poizvedb (niti ena poizvedba ni bila enaka drugi). Pri takem testiranju je bilo polovico poizvedb izvršenih v manj kot 0,08 sekunde, povprečje je bilo le 1,2 sekunde. Nobena še tako zahtevna poizvedba ni trajala dalj kot 33 sekund. Pri tem pa se je povprečna obremenjenost procesorja gibala okoli 19%, kar kaže še na velike rezerve pri uporabi s strani več uporabnikov.

3.2.1.8. OLAP kocka

Kocke so glavni objekti v OLAP tehnologiji. Kocka je skupek podatkov, ki je ponavadi del skladišča podatkov, organiziran v večdimenzijsko strukturo, definiran z nizom dimenzij in mer.
Uporabniki uporabljajo različna orodja za dostop do Analysis Serverja in izvajajo poizvedbe iz kock, ki so shranjene na strežniku. Večina programov je narejena tako, da uporabnik s pomočjo različnih grafičnih orodij zgradi želeno poizvedbo, na podlagi tega pa

program pošlje zahtevo strežniku, ki iz kocke pridobi ustrezne podatke in jih pošlje nazaj uporabniškemu vmesniku. Tak način omogoča kreiranje poizvedb brez potrebe po pisanju računalniških programov.
Zaradi tega, ker so ploščati nizi zapisov v OLAP kocki pretvorjeni v hierarhično strukturo, je omogočeno kreiranje poročil, ki so osredotočena na želeno raven podrobnosti.

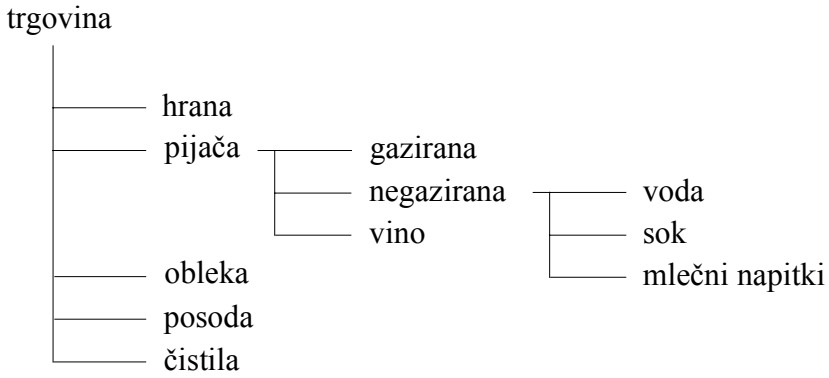
a) hierarhična struktura dimenzij

Poslovne kategorije so po naravi hierarhične, zato se z OLAP sistemi da zelo lepo modelirati poslovni sistem.

Primer:

1. Neka trgovina prodaja različne skupine izdelkov, kot so: hrana, pijača, čistila, obleka in posoda.
2. Določena skupina izdelkov, recimo pijača, je razdeljena na več podskupin, kot so: gazirana pijača, negazirana pijača in vino.
3. Določena podskupina, npr. negazirana pijača, je razdeljena še naprej, na podskupine kot so: voda, sok, mlečni napitek.

Slika 12: Primer hierarhične razdelitve izdelkov:



Vir: Avtor

Taka nivojska delitev, omogoča enostaven pregled agregiranih vrednosti z združevanjem podskupin (drill-up), ali takoimenovano vrtanje v globino (drill-down) podatkov, kjer se da zelo hitro dobiti vrednosti za posamezne osnovne člane dimenzij.

Tovrstno delitev na različne podskupine si je mogoče predstavljati tudi kot narobe obrnjeno drevo, kjer deblo predstavlja vrhnji nivo v hierarhiji in listi osnovni nivo, ali pa odnos v smislu "roditelj-otroci".

Pomembno je, da se pri načrtovanju OLAP kock loči različno zgrajene hierarhične delitve za isto entiteto. Entiteta izdelek je namreč lahko vključena tako v kocko za potrebe prodajnega sektorja kot v kocko, ki je namenjena kot pomoč pri odločanju v proizvodnem sektorju. Hierarhični strukturi sta v obeh primerih različni, zato je potrebno izdelati dve ločeni dimenziji. Primer: Izdelek v prodaji je lahko hierarhično razdeljen na državo, regijo, mesto in prodajnega referenta, nasprotno pa se izdelek v proizvodnji deli na državo, mesto, dobavitelja in proizvajalca.

b) Tabela dejstev

Tabela dejstev je zelo pomemben del kreiranja kocke. V njej so shranjene vse konkretne vrednosti - mere, ki so namenjena raziskavi v nekem analitičnem procesu. Če se iz dimenzij črpa podatke za vprašanja, se v tabeli dejstev dobi odgovore.

Nekaj značilnosti tabel dejstev:

1. V njej se nahajajo ključi, preko katerih se lahko pri poizvedbah dostopa do atributov v dimenzijskih tabelah.
2. V njej se nahajajo mere, ki so ponavadi numerične aditivne narave. Včasih se določene vrednosti pri pregledih na višjih nivojih v hierarhiji ne sme seštevati, ampak mora OLAP strežnik izračunati povprečje. Primer: Če se želi narediti kocko za pregled zaloge nekega izdelka v skladišču v prvem četrtletju, je potrebno izračunati povprečje zalog v januarju, februarju, marcu in aprilu in ne sešteti vseh mesečnih zalog. V nekaterih redkih primerih obstaja možnost, da v tabeli dejstev ni numeričnih vrednosti. Tak primer je npr. kreiranje kocke, kjer se spremljajo pojavi različnih dogodkov v poslovnem procesu. V tem primeru sistem sešteva število pojavov nekega niza znakov.
3. Tabela dejstev je praviloma največja izmed vseh tabel, ki so vključene v OLAP kocke. Obstaja veliko primerov v praksi, kjer tabela dejstev vsebuje tudi več milijonov vrstic podatkov.

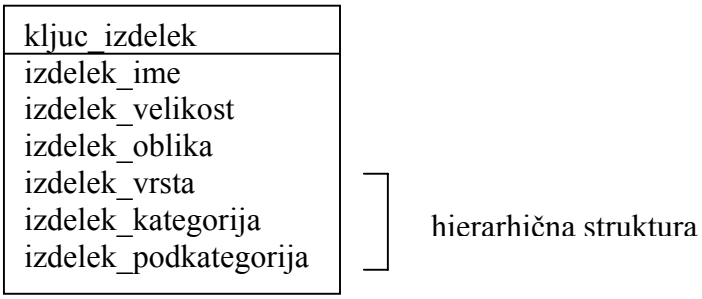
c) Dimenzijske tabele

Poznana sta dva pristopa pri kreiranju dimenzijskih tabel v OLAP kockah:

1. Nenormaliziran pristop:

Dimenzijske tabele vsebujejo atribute, ki opisujejo zapise iz tabele dejstev. Atributi v dimenzijskih tabelah vsebujejo informacije o hierarhični strukturi določene entitete. Vsaka dimenzijska tabela vsebuje informacije le o eni entiteti, npr. tabela *izdelek* vsebuje le podatke o izdelkih, tabela *dobavitelj* pa le o dobaviteljih.

Slika 13: Primer nenormalizirane dimenzijske tabele entitete *izdelek*



Vir: Avtor

V tej tabeli so vrednosti hierarhičnih atributov zapisane v besedilni obliki, vsak atribut predstavlja ime enega od nivojev v hierarhiji.

Slika 14: Primer dejanskih vrednosti v dimenzijski tabeli *izdelek*

kljuc_izdelek	izdelek_ime	izdelek_velikost	izdelek_oblika	izdelek_vrsta	izdelek_kategorija	izdelek_podkategorija
0001	vijak K-30	30	standard	kovinski deli	vijak	križni
0002	vijak K-123	123	standard	kovinski deli	vijak	križni
0003	matica A210-1	45	A210	kovinski deli	matica	jeklena
0004	matica A210-2	33	A210	kovinski deli	matica	jeklena
0005	matica A210-3	78	A210	kovinski deli	matica	pozlačena

Vir: Avtor

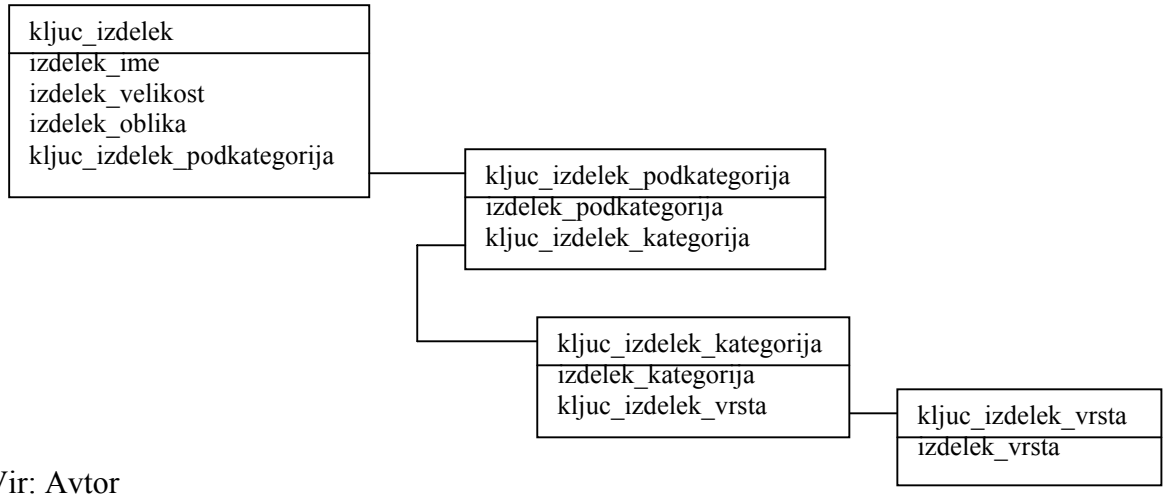
Iz tabele sta razvidni dve značilnosti:

- Taka zgradba je najenostavnejša oblika za kreiranje poizvedb: Vsi podatki o izdelku so shranjeni v eni tabeli, zato je kreiranje poizvedb zelo enostavno.
- Nenormalizirana podatkovna struktura zahteva več prostora na disku in več delovnega spomina kot pri normaliziranih tabelah.

2. Normaliziran pristop:

V tem primeru se za določeno entiteto ne uporabi le ene tabele, ampak se kreira tabele glede na število hierarhičnih nivojev do take mere, da se besedilne vrednosti atributov, ki predstavljajo hierarhično strukturo določene entitete, ne podvajajo. S tabelo dejstev so povezane le primarne dimenzijske tabele. Poddimenzije pa so med seboj povezane preko ključev. Tak način gradnje pomeni manjšo porabo prostora na disku kot pri nenormaliziranem pristopu. Ključi za povezavo poddimenzijskih tabel so v takem primeru običajno umetno zgrajeni.

Slika 15: Primer normaliziranega pristopa pri kreiranju entitete *izdelek*



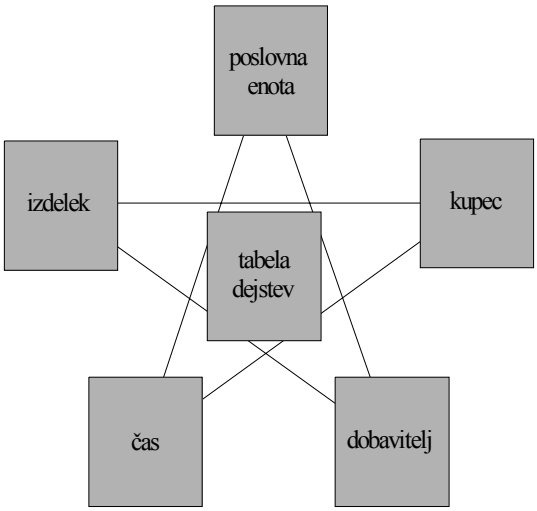
Vir: Avtor

Glede na pristop pri gradnji dimenzijskih tabel sta poznana dva različna načina kreiranja OLAP kock:

1. Konsolidiran pristop:

Tak način omogoča gradnjo tako imenovane zvezdne sheme. Je najpogosteje uporabljen pri gradnji OLAP kock. Pri zvezdni shemi je tabela dejstev postavljena na sredino sheme, nenormalizirane dimenzijske tabele pa na krake zvezde (slika 16). Za krake se torej uporabi tabele, ki vsebujejo tudi atribute, ki predstavljajo hierarhično razdelitev določene entitete.

Slika 16: Primer zvezdne sheme



Vir: Avtor

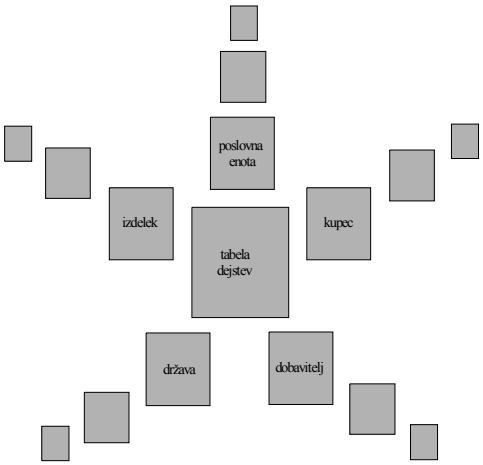
Prednosti, ki jih prinese tak sistem, zasenčijo slabosti:

- Manj je tabel, zato se poizvedbe lažje gradijo.
- Manj je povezav med tabelami, kar lahko pospeši poizvedbe.
- Taka struktura omogoča uporabo bitmap indeksiranja.
- S strani skrbnika baz podatkov je lažje nadzirati in vzdrževati tako zbirko.

2. Shema snežinke:

Je nadgradnja zvezdne sheme. Podatki v dimenzijah so normalizirani, zato je potrebno izdelati več poddimenzij. Nastane toliko tabel, kolikor je hierarhičnih nivojev posamezne dimenzije. Če se zvezdna shema nadgradi s poddimenzijskimi tabelami, dobi shema obliko snežinke. Od tu tudi poimenovanje take zgradbe OLAP kocke. Kljub manjši porabi prostora na disku se tak način redkeje uporablja, ker je kasnejše kreiranje poizvedb iz velikega števila dimenzij precej bolj zahtevno.

Slika 17: Primer sheme snežinka:



Vir: Avtor

d) Agregati:

Microsoft SQL Server 2000 Analysis Services omogoča kreiranje predhodno izračunanih sumarnih podatkov, ki skrajšajo odzivni čas pridobivanja podatkov pri poizvedbah. Strežniku ni potrebno uporabiti vseh osnovnih podatkov za neko preračunavanje, ampak uporabi že predhodno agregirane podatke.
Načeloma več agregatov pomeni večjo hitrost sistema, vendar je potrebno biti pozoren na naslednje faktorje:

1. Prostor na disku: Več agregatov pomeni večjo zasedenost diskov, večje število bralno pisalnih operacij na disk. V nekem trenutku kreiranje novih agregatov ne pomeni več performančnih izboljšav.
2. Izgradnja kocke: V primeru vključevanja novih podatkov, je potrebno kocko na novo sprocesirati. Procesiranje pomeni tudi kreiranje agregatov. Pri velikih zbirkah podatkov je čas izdelave kocke lahko precej dolg.
3. Eksplozija podatkov: Če bi hoteli vnaprej izdelati vse agregate, ki so možni, bi prišlo do ogromnega povečanja zahtev po prostoru na disku in procesorski moči. Prišlo bi do eksponentne rasti velikosti večdimenzijske strukture. Strežnik Analysis Services shranjuje le tiste celice v večdimenzijski strukturi, ki vsebujejo podatke, praznih ne shranjuje. V tej zvezi se pojavljata dva tipa kock:

    - Zgoščene, jedrnate kocke: Odstotek celic, ki vsebujejo podatke v večdimenzijski strukturi, je velik. Zgoščene kocke zavzemajo več prostora kot redke kocke identične zgradbe.
    - Raztresene, redke kocke: Kocke, kjer je odstotek celic v večdimenzijski strukturi, ki vsebujejo podatke, zelo majhen. Take kocke zavzemajo manj prostora na trdem disku kot zgoščene kocke identične zgradbe.

Poleg raznolikega načina shranjevanja podatkov in shranjevanja le tistih celic, ki vsebujejo vrednosti, Analysis Services uporablja inteligenten algoritem za kreiranje optimalnega števila in oblike agregatov, da lahko ob kakršnikoli poizvedbi s strani uporabnika zadovolji FASMI zahtevam. Na ta algoritem lahko uporabniki vplivajo samo s parametri, ki so navedeni v "čarovniku" *Storage Design Wizard*, to je:

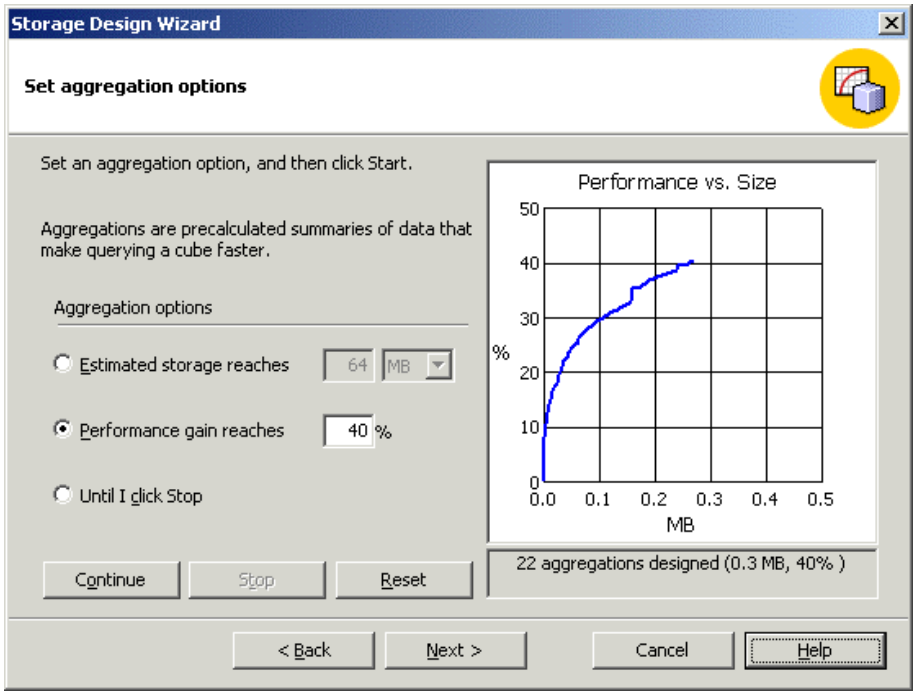
- *Estimated storage reaches*: V primeru omejenega prostora na disku se lahko določi največjo dovoljeno velikost (v MB ali GB), ki naj bo namenjena za agregate.
- *Performances gain reaches*: Pospešitev sistema, ki pomeni odstotek izboljšanja hitrosti poizvedbe glede na maksimalni in minimalni poizvedovalni čas.
- *Until I click stop*: Omogoča ročno določanje števila agregatov in zasedenosti prostora na disku na podlagi opazovanja krivulje *Performance vs. Size*. Na grafu je lepo vidno, kdaj bo večje število agregatov pomenilo velik prispevek pri hitrosti (strma krivulja) in kdaj nadaljnje povečanje povzroča le več zasedenega prostora na disku ob minimalni pospešitvi.

Pospešitev sistema se izračuna po sledeči enačbi:

*pospesitev = 100 * (maxcas - zeljencas) / (maxcas - mincas)*

pospesitev = željen odstotek pospešitve sistema
maxcas = maksimalni poizvedovalni čas
mincas = minimalni poizvedovalni čas
zeljencas = željen odziv sistema po rezultatu

Slika 18: Primer kreiranja agregatov pri izboru pospešitve 40 %



Vir: Ekranska slika, Analysis Manager, Concepts & Tutorial

Primer:

Vzemimo OLAP model, kjer smo pri izvajanju poizvedb izmerili povprečen odzivni čas sistema 22 sekund, pri čemer je moral računalnik vsako poizvedbo preračunati na podlagi izvornih podatkov, ker model ni vključeval agregiranih, sumarnih tabel. S pomočjo "čarovnika" za kreiranje agregatov (*Storage Design Wizard)* smo ugotovili, da bi povprečen odziv sistema pri največjem številu vnaprej izdelanih agregatov trajal 2 sekundi.

Vprašanje, ki se zastavlja, je, kolikšna bi morala biti izbrana pospešitev sistema v *Storage Design Wizardu,* če želimo, da se odzivni čas sistema skrajša na 14 sekund ?

maxcas = 22 sekund
mincas = 2 sekundi
zeljencas = 14 sekund
---
pospesitev = ?

*pospesitev = 100 * (maxcas - zeljencas) / (maxcas - mincas)*

*pospesitev = 100 * (22 - 14) / (22 - 2)*

pospesitev = 800 / 20 = 40 %

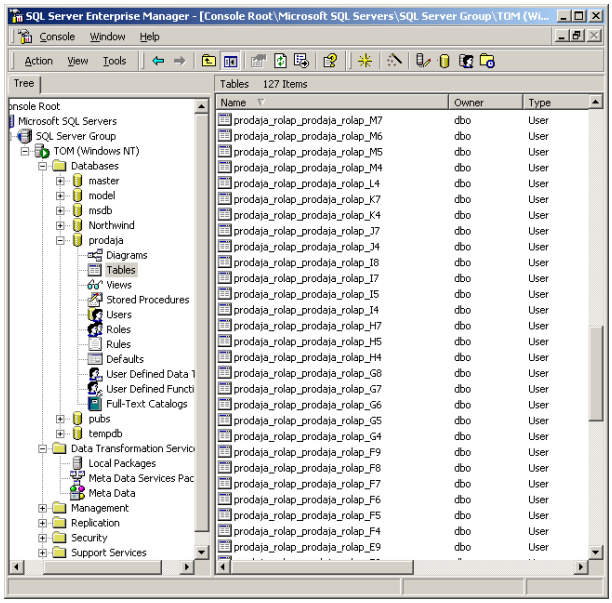
Potrebno je izbrati 40% pospešitev sistema, če hočemo, da bo poizvedba v povprečju trajala 14 sekund.

Na sliki 18 je lepo vidno, da je algoritem za kreiranje optimalnega števila in oblike agregatov izračunal, da je za konkreten primer potrebno izdelati 22 agregatov, ki bodo zasedli 0,3 MB prostora na disku, oziroma da se doseže želeno 40 % pospešitev poizvedb.

Možnost, kjer bi lahko razbrali strukturo agregiranih tabel in s tem algoritma za optimizacijo, je, da se OLAP kocka shrani v takoimenovanem ROLAP načinu, torej v relacijskem načinu, kjer sistem zgradi sumarne tabele v dvodimenzionalni obliki. V tem primeru Analysis Server shrani agregirane tabele, ki so določene s programom *Storage Design Wizard*, kar v tisto bazo podatkov, kjer se nahajajo tudi izvorni podatki dimenzij in mer.

Na sliki 19 se vidi primer s strani sistema določenih agregiranih tabel (98 tabel), ki so nastala ob ROLAP načinu shranjevanja OLAP kocke *prodaja_rolap.*

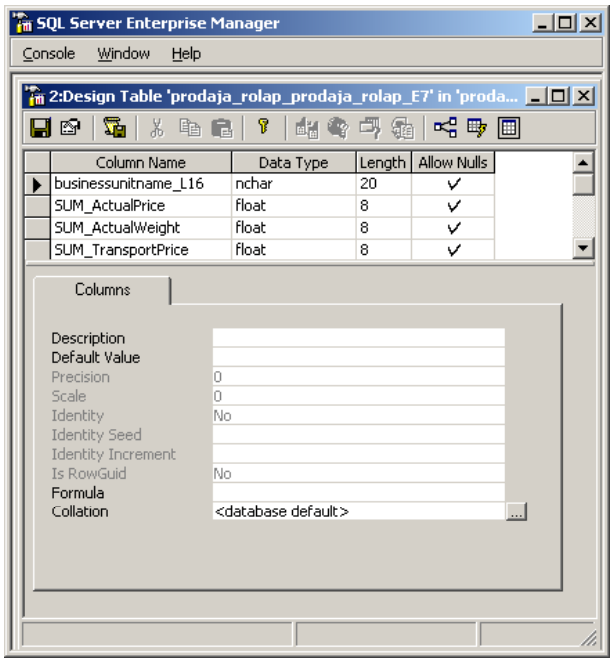
Slika 19: Prikaz sumarnih tabel pri ROLAP načinu shranjevanja OLAP kock



Vir: Ekranska slika, Microsoft SQL Server, Enterprise Manager

Nazive sumarnih tabel, ki so vidni v Microsoft SQL Serverju, je določil sistem sam, prav tako tudi imena atributov znotraj posamezne tabele v obliki, ki je vidna na sliki 20.

Slika 20: Primer s strani sistema določenih imen atributov v sumarnih tabelah pri ROLAP načinu shranjevanja OLAP kock



Vir: Ekranska slika, Microsoft SQL Server, Enterprise Manager

3.2.1.9. MOLAP, ROLAP, HOLAP

Microsoft Analysis Services omogočajo izbiro različnih načinov shranjevanja podatkov, agregatov in struktur pri kreiranju OLAP kock:

- Večdimenzijski OLAP (Multidimensional OLAP - MOLAP):

Pri tem načinu so tako podatki kot agregati shranjeni v večdimenzijskih strukturah. Te strukture se nahajajo ločeno od skladišča podatkov. Je privzet način shranjevanja pri kreiranju kock, primeren predvsem za pogoste poizvedbe. Tak način ima več prednosti:

a) Ne shranjujejo se prazne celice, ki nastanejo v primeru redkih matrik večdimenzijskih baz podatkov, ko je le nekaj celic zapolnjenih z vrednostmi. Ostali podatki v kocki so shranjeni na podlagi posebnega kompresijskega algoritma, kar namesto eksplozije podatkov pomeni celo velik prihranek prostora na disku v primerjavi z zapisom relacijskih struktur.
b) Operacije povezovanja tabel preko ključev, ki so časovno zelo potratne, niso potrebne, ker so podatki v večdimenzijskih strukturah posebno indeksirani (bitmap indeksiranje).
c) Strežniku praviloma ni potrebno zaklepati podatkov, ker so podatki določeni samo za branje.

d) Vsi podatki so shranjeni na enem mestu. Preko Microsoft Excela jih je možno shraniti tudi v datotečni obliki. Vse izvorne in agregirane podatke, shranjene v eni sami datoteki, je možno enostavno prekopirati na stran uporabnika in jih uporabljati za "off-line" analizo - brez potrebe po zagotovitvi stalne povezave s strežnikom.

Slabosti:

Izvorni podatki, agregati, indeksi in informacije o hierarhični strukturi, ki so shranjeni v MOLAP kocki, zaradi zelo dobrega kompresijskega algoritma zavzemajo manj prostora na trdem disku kot zgolj izvorni podatki, ki so shranjeni v skladišču podatkov v relacijski obliki. Kljub temu se lahko smatra kot slabost dejstvo, da morajo biti isti izvorni podatki zapisani dvakrat - enkrat v relacijski, drugič v večdimenzionalni obliki. Ob velikih količinah izvornih podatkov lahko tako organizirani sistemi zavzamejo precej prostora na trdem disku. Poleg tega pa so potrebna posebna orodja za pregledovanje in obvladovanje večdimenzijskih podatkov.

- Relacijski OLAP (Relational OLAP - ROLAP)

Tako agregati kot detajlni podatki so shranjeni v relacijski zbirki podatkov. Prednost takega načina shranjevanja je, da se lahko izkoristi prednosti relacijske strukture podatkov, kot je npr. dober nadzor nad vsebino podatkov in agregatov, kar pri večdimenzijski obliki ni možno, enostavno shranjevanje in kreiranje poizvedb. Tak način je sicer počasnejši glede odzivnosti od MOLAP ali HOLAP sistema, je pa primernejši v primeru velikih količin izvornih podatkov.
Tak način je smiselno uporabiti v primeru izdelave poizvedb, ki niso zelo pogoste.

Primer: Če večina uporabnikov pregleduje podatke le za eno leto nazaj, se lahko starejšo zgodovino shrani v ROLAP strukturo. Prednost je tudi v uporabi podatkov iz različnih relacijskih zbirk, tudi zbirk podatkov, ki niso del Microsoftovih produktov (npr. zbirke podjetja Oracle), kar je lahko velika prednost v heterogenih okoljih.

- Hibridni OLAP (Hybrid OLAP - HOLAP)

Osnovni podatki so zabeleženi v relacijski zbirki, agregati pa v večdimenzionalnih strukturah. Je nekakšna kombinacija obeh prej omenjenih načinov. Najpogosteje se uporablja v primerih, ko uporabniki največ povprašujejo po agregiranih vrednostih, npr. po letni ali četrtletni prodaji. Ti podatki so hitro vidni, ker so agregati shranjeni v zmogljivi večdimenzijski strukturi, dnevne in tedenske vrednosti pa so shranjene v ROLAP načinu, s čimer je zagotovljeno, da ni podvajanja shranjevanja izvornih podatkov.

Primerjava med različnimi načini shranjevanja je vidna na sliki 21.

Slika 21: Primerjava med načini shranjevanja

	MOLAP	ROLAP	HOLAP
mesto osnovnih podatkov	kocka	relacijska tabela	relacijska tabela
mesto agregatov	kocka	relacijska tabela	kocka
hitrost poizvedbe	hitra	počasna	srednja
poraba diskovnega prostora	velika	nizka	srednja
vzdrževanje podatkov	zahtevno	enostavno	srednje

Vir: Microsoft Press, december 1999

3.2.2. Večdimenzijski poizvedovalni jezik MDX

Programski jezik za kreiranje poizvedb in upravljanje večdimenzijskih struktur (OLAP kock) se imenuje Multidimensional Expressions (MDX). Uporablja se lahko za OLAP kocke shranjene na Analysis Services strežniku ali za kocke, shranjene lokalno pri uporabniku (Dewson, junij 2000).

3.2.2.1. MDX nasproti Transact-SQL

MDX je s T-SQL podoben toliko, da oba programska jezika uporabljata podoben format (SELECT…FROM…WHERE), ampak podobnost se tu konča. Ne sme se ju direktno primerjati. MDX je potrebno obravnavati kot povsem nov programski jezik .

3.2.2.2. Večdimenzijske sheme

Večdimenzijske sheme predstavljajo organiziranost večdimenzijskih baz podatkov (Spofford, 2001). Delijo se na več objektov: kocke, dimenzije, člane in celice.

a) Kocka: Predstavlja set med seboj povezanih osi, ki tvorijo n-dimenzijsko mrežo. Vsak presek (celica) v taki mreži je natančno definiran z nizom koordinat. Vrednost koordinat je enaka članom dimenzij, ki so projicirane na os.

b) Dimenzija: Je strukturna lastnost kocke. Predstavlja organizirano hierarhijo kategorij oziroma nivojev, ki predstavljajo podatke, npr. geografska dimenzija lahko vsebuje nivoje, kot so: država, regija, občina, mesto.
Glede postavitve dimenzij v smislu izdelave analiz obstajajo:

- Tiste, ki se projicirajo na os.
- Tiste, ki predstavljajo filter, oziroma omejitev nabora projiciranih dimenzij.

V zvezi z dimenzijami se pojavljajo novi pojmi, kot so:

- Dimenzionalnost: Definirana je s številom osi, ki jih uporabnik želi prikazati v kocki.
- Projiciranje dimenzije: Na eno os se lahko projicira ena ali več dimenzij.
- Ugnezdene dimenzije: Če se na posamezno os projicira več kot ena dimenzija.

c) Člani: V večdimenzijski shemi predstavljajo posamezno diskretno vrednost v dimenziji. To so različne realne vrednosti, ki predstavljajo zalogo vrednosti posamezne dimenzije.

   Člani dimenzij, ki se projicirajo na os, tvorijo set. Z MDX sintakso se to zapiše kot: *{[Maribor.pijaca], [Maribor.hrana]}*

d) Celica: Predstavlja presek posameznih koordinat osi. V celici je shranjena tista informacija, ki uporabnika zanima za vrednotenje. Primer: Vrednost celice *150* na presečišču koordinat *[Ljubljana].[kruh].[januar]* pomeni, da je bilo januarja v Ljubljani prodano 150 ton kruha.

3.2.2.3. Kreiranje MDX izjave

Podobno kot pri T-SQL kreiranju poizvedb, si je tudi pri MDX izjavah potrebno predstavljati rezultat, ki se želi dobiti z neko poizvedbo.

MDX izjava v osnovi vsebuje naslednje elemente:

- Ime kocke, s katerim se določi okvire poizvedbe,
- Število osi,
- Dimenzije, ki se projicirajo na osi in način ugnezdenja dimenzij,
- Člane, ki se iz določene izbrane dimenzije želijo vključiti v analizo (posamezen hierarhični nivo in člani s tega nivoja),
- Člane posamezne dimenzije, ki se ne projicirajo na os, predstavljajo filter - natančnejši okvir za prikaz projiciranih dimenzij.

Mere se obravnavajo kot posebna dimenzija, ki jo je ravno tako možno projicirati na eno izmed osi, ali pa se ena izmed mer obravnava kot filter.

a) Definiranje osi:

V MDX poizvedbah se lahko uporabi do štiriinšestdeset osi, vendar se v praksi redko uporablja več kot štiri. Dve ali tri osi so najpogostejše. Če se podatki želijo prikazati v klasičnem dvodimenzionalnem pogledu z razpredelnico in grafom, potem se uporabita le dve osi. Vsaka os je definirana s številko: AXIS(0), AXIS(1), ..., AXIS(n). Zaradi preglednosti za prvih pet osi obstajajo logična imena:

Slika 22: Prvih pet osi je definirano z logičnimi imeni

IME	OS	OPIS
COLUMNS	AXIS(0)	x-os v dvodimenzionalni tabeli
ROWS	AXIS(1)	y-os v dvodimenzionalni tabeli
PAGES	AXIS(2)	z-os v trodimenzionalnem prikazu kocke
SECTIONS	AXIS(3)	
CHAPTER	AXIS(4)	

Vir: Microsoft Press, december 1999

b) Osnovna sintaksa MDX poizvedbe:

SELECT <definiranje_osi> [, <definiranje_osi>...]
FROM <definiranje_kocke>
WHERE <definiranje_filtra>

Primer:

SELECT [izdelek].Children ON COLUMNS
NON EMPTY [naziv_trgovine].Members ON ROWS
FROM prodaja
WHERE [spol].[Z]

Pomen sintakse zgornjega primera:

- Zaradi hierarhičnega odnosa otrok-roditelj se z navedbo funkcije *[izdelek].Children* določi prikaz naslednikov vrhnjega nivoja v hierarhični strukturi.
- Ukaz ON COLUMNS pomeni postavitev *[izdelek].Children* v stolpce, oziroma na x-os.
- Funkcija *[naziv_trgovine].Members* pomeni izbor vseh članov dimenzije *naziv_trgovine,* vendar samo tiste (ukaz *NON EMPTY*), ki imajo v kocki *prodaja* zabeleženo prodajo.

- *[naziv_trgovine].Members:* Nazivi trgovin so postavljeni v vrstice (*ON ROWS*) oziroma na y-os.
- FROM *prodaja:* Uporabijo se podatki iz večdimenzijske strukture - kocke z imenom *prodaja*
- WHERE *[spol].[Z]*: Z ukazom WHERE se določijo člani neke dimenzije, ki pomenijo okvir - filter za prikaz projiciranih dimenzij. V zgornjem primeru bo pregled števila prodanih enot različnih izdelkov po posameznih trgovinah veljal le za prodajalke - za osebe ženskega spola *[spol].[Z]*.
- Če mere (*Measures*) niso posebno navedene, se pri poizvedbi pokažejo vrednosti prve navedbe iz niza mer (v prikazanem primeru je to število prodanih enot). Če bi želeli videti vrednostni prikaz prodaje, bi filter prikaza dopolnili z ukazom: WHERE *([spol].[Z], [Measures].[vrednost_prodaje]).*

Slika 23: Rezultat primera bi tabelarično izgledal takole:

	hrana	pijača	ostalo
trgovina1	345	1254	3567
trgovina3	555	1100	5678
trgovina7	78	213	891

Vir: Avtor

c) Izvedeni član:

Iz osnovnih članov posameznih dimenzij je možno kreirati tudi druge - izvedene, izračunane člane.

2 primera možne uporabe izvedenih članov:

- WITH MEMBER *[cas].[pomlad]* AS *'Sum({[mesec].[3]:[mesec].[5]})'* je sintaksa za kreiranje izvedenega člana *pomlad*, ki je definiran kot vsota od 3. do 5. meseca, torej od marca do maja. V praksi se večkrat pojavi primer, ko osnovni časovni hierarhični nivoji (leto, kvartal, mesec, teden, ...) ne zadoščajo za kreiranje želenih analiz. V tem primeru se s pomočjo izvedenih članov vnaprej izdelajo časovni intervali, kot so npr.: pomlad, polletje, delovni teden, popoldan.

- WITH MEMBER *[Measures].[cena_na_kg]* AS *'[Measures].[prodaja_vrednost] / [Measures].[prodaja_kolicina]'* je primer sintakse za kreiranje izvedene mere *cena_na_kg*, kjer je uporabljena kar matematična operacija deljenja prodajne vrednosti s količino.

3.3. Microsoft English Query 2000

3.3.1. Osnovne značilnosti

Microsoft English Query (EQ) omogoča pridobiti uporabniku določeno informacijo iz zbirke podatkov z uporabo naravnega angleškega jezika namesto formalnega računalniškega jezika za kreiranje poizvedb.
EQ aplikacijo je možno vključiti v katerikoli program, ki podpira COM objekte. Lahko je spletna aplikacija, ki je nameščena na spletnem strežniku Microsoft Internet Information Server verzije 3.0 ali novejšem.
English Query 2000 urejevalnik je del razvojnega orodja Microsoft Visual Studio verzije 6.0 (Richard, april 2002).
EQ verzija 2000, ki je del SQL strežnika 2000, je popolnoma integrirana z Analysis Services, kar pomeni, da je možno kreirati projekte, ki omogočajo tudi pretvorbo naravnega angleškega jezika v večdimenzionalni poizvedovalni jezik MDX. S tem je možno dobiti rezultate v obliki večdimenzionalnih struktur ali klasični dvodimenzionalni tabeli tudi za zahtevne podatkovne strukture v zelo kratkem času. V prejšnji verziji programa Microsoft English Query te večdimenzijske podpore ni bilo, ni bilo možno izdelati EQ modela, ki bi znal spreminjati vprašanje, zastavljeno v naravnem angleškem jeziku, v MDX računalniško obliko. Prejšnja verzija je poznala le pretvorbo v T-SQL programski jezik.

3.3.2. Zgradba English Query aplikacije

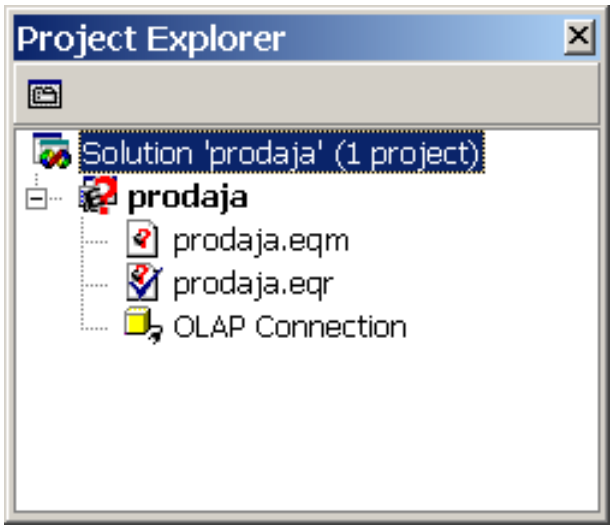
Neka rešitev (solution), ki je shranjena v datoteki s privzeto končnico (*.sln), je lahko sestavljena iz enega ali več med seboj povezanih projektov. Projekti, ki imajo privzeto končnico (*.eqp), so nadalje lahko sestavljeni iz ene ali več komponent. Glavni sta dve skupini komponent:

- Moduli, ki imajo privzeto končnico (*.eqm). Vsak modul je sestavljen iz EQ modela, ki vsebuje strukturo relacijske ali večdimenzijske zbirke podatkov in definicije semantičnih objektov, kot so entitete, relacije in vnosi v slovar. Namen gradnje projekta iz več modulov je ta, da se lahko posamezne module kot samostojne dele prenaša iz enega v drug projekt, kar omogoča lažje načrtovanje EQ aplikacij.

- Regresijske datoteke, ki imajo privzeto končnico (*.eqr). To so datoteke, kjer so v besedilni ali XML obliki zapisana vprašanja, ki jih je uporabnik zastavljal pri testiranju pravilnosti delovanja modela. Te datoteke je možno z urejevalniki besedil tudi ročno popravljati. XML oblika datotek zelo olajša povezljivost z ostalimi programi (Hunter, november 2001).

Preverjeno delujoč projekt je na koncu potrebno povezati in zgraditi v EQ aplikacijo (*.eqd). Ta je nato pripravljena za vključitev v različne aplikacije.

Slika 23: Primer zgradbe English Query rešitve



Vir: Ekranska slika, Microsoft English Query, Project Explorer

3.3.2. Osnovni koraki za izdelavo EQ aplikacije

Kreiranje English Query aplikacije se lahko strne v pet osnovnih korakov:

1. Najprej je potrebno ugotoviti, katera so verjetna vprašanja, ki jih bo uporabnik zastavljal. Poznavanje vprašanj pomaga pri določanju entitet in relacij med njimi ter testiranju pravilnega delovanja aplikacije.
2. Kreiranje osnovnega modela, določanje osnovnih entitet in relacij med njimi iz relacijske ali večdimenzijske zbirke podatkov.
3. Nadgradnja osnovnega modela z entitetami in povezavami, ki zagotavljajo odgovore na zahtevnejša vprašanja.
4. Testiranje in nadgrajevanje modela do stopnje, ko uporabnik dobi pravilne odgovore na svoja vprašanja.
5. Izdelava končne aplikacije. Lahko je to spletni program ali del neke druge "strežnik-odjemalec" aplikacije.

Kot pomoč pri gradnji EQ modela je primerno uporabiti kakšno izmed strokovnih metod, kot je npr. objektno orientirana metoda TAD (Damij, 2001), kjer je v tabelarični obliki možno definirati različne elemente modela, kot so: entitete, aktivnosti in naloge.

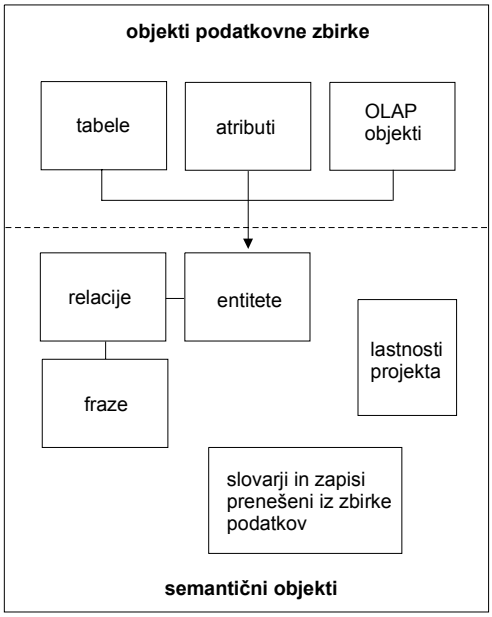
3.3.2.1. Definiranje zahtev uporabnika

Potrebno je opraviti razgovor z bodočim uporabnikom aplikacije, da se ugotovi, katere informacije želi pridobiti od sistema, na kakšen način sedaj pridobiva informacije in kaj pogreša. Spisek vprašanj omogoča lažje definiranje entitet, ki se bodo vključile v sistem, povezave med njimi in ostale elemente semantičnega modela. V primeru prodaje bo vodstvo podjetja verjetno želelo vedeti podatke o vrednostni in količinski prodaji izdelkov v določenem časovnem obdobju, o kupcih, številu naročil, ki jih je pridobil posamezen komercialist in prav gotovo tudi nekatera specifična vprašanja, kot je npr.: Koliko in katere prejemnike ima posamezen kupec, če je kupec izdelkov drug kot je prejemnik.
Natančen pregled dosedanjih načinov analiziranja poslovanja podjetja in razgovori z vodilnimi delavci so predpogoj za uspešno načrtovanje EQ modela. Potrebno se je zavedati, da ustvarjanje EQ modela predstavlja natančno analiziranje nekega segmenta realnega življenja z različnih vidikov. Ta spoznanja pa je z različnimi metodami potrebno povezati v računalniški model realnosti: od ljudi, predmetov, dokumentov, do povezav, tipa in kvalitete odnosov med njimi. Model realnosti mora biti odraz dejanskega "dihanja" podjetja in ne nek namišljen teoretičen primer iz knjig.

3.3.2.2. Kreiranje osnovnega modela

Drugi korak pri kreiranju projekta je kreiranje modela. Model je zbirka informacij o objektih iz zbirke podatkov (tabele, atributi, povezave), semantičnih objektih (entitete, relacije in vnosi v slovarje) in dodatnih lastnostih projekta.

Slika 24: English Query model:

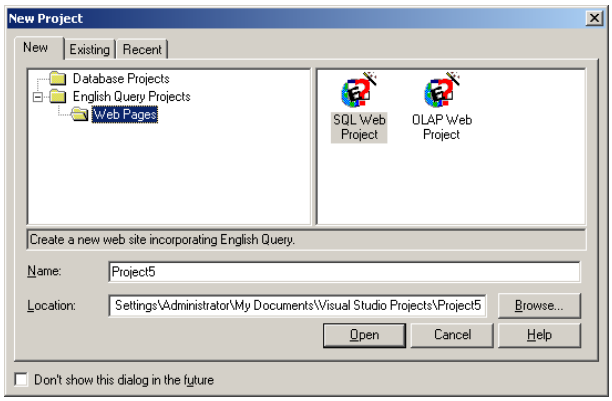


Vir: Microsoft English Query Books Online

Objekte, ki so vključeni v model, je možno shraniti v različne module. Organizacija objektov v module je zelo primerna predvsem v okolju, kjer se kreirajo razgibane aplikacije, ki imajo podobno logiko delovanja. Objekti se lahko poljubno premikajo iz enega v drug modul v projektu. Uporaba modulov omogoča, da se nek EQ model naredi le enkrat, uporabi pa se ga lahko večkrat, v odvisnosti od izbora različnih modulov, kar razvijalcem zelo poenostavi delo.

Za izdelavo novega projekta je na voljo "čarovnik" za EQ SQL projekt, če je v ozadju relacijska zbirka podatkov, kot tudi EQ OLAP projekt, ki se gradi na osnovi večdimenzijskih podatkovnih struktur. V tem primeru na podlagi ustrezne povezave računalnik sam zgradi osnovne entitete in povezave med njimi. Ob izbiri določene OLAP kocke v EQ model "čarovnik" samodejno generira osnovni semantični model. Poleg omenjenih je možno izbrati tudi projekt izgradnje SQL ali OLAP EQ internet aplikacije (slika 25)

Slika 25: Izbor izdelave SQL ali OLAP EQ internet aplikacije

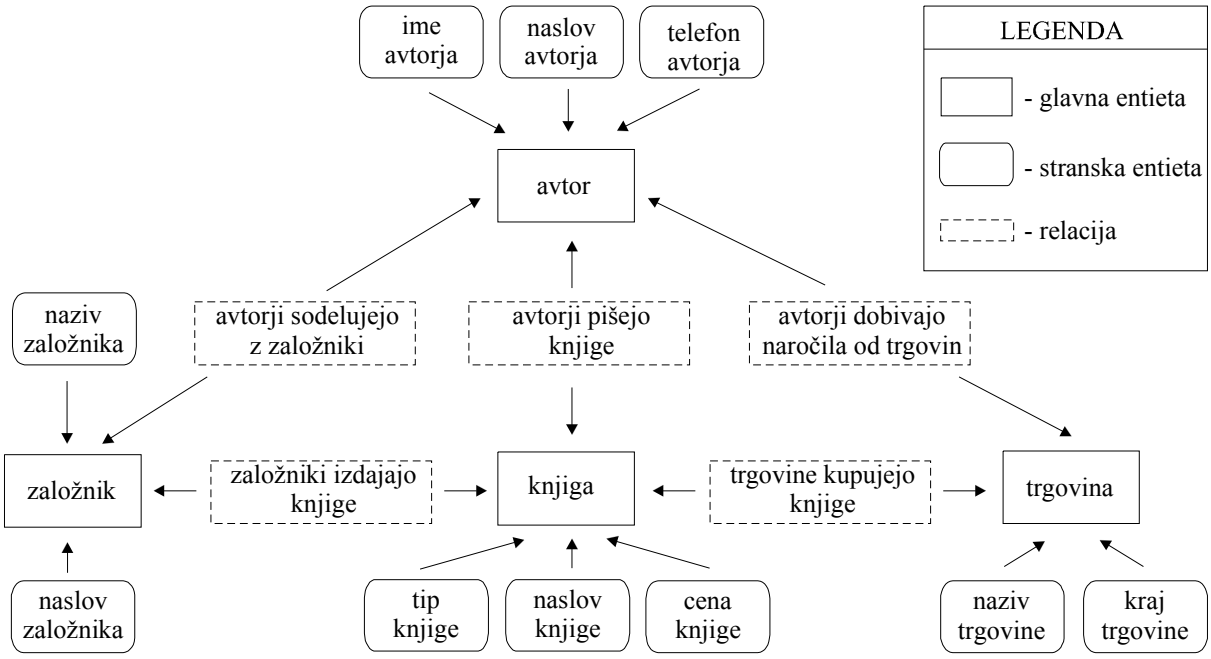


Vir: Ekranska slika, Microsoft English Query

a) Semantični model

Semantični model predstavljajo entitete, relacije, dodatni vnosi v slovarje in posebne lastnosti projekta.

Slika 26: Primer semantičnega modela



Vir: Avtor

Na sliki 26 je prikazan primer semantičnega modela za kreiranje EQ aplikacije za potrebe založništva.

1. Glavne entitete, ki "nastopajo" v takem modelu, so: *avtor*, *založnik*, *knjiga* in *trgovina*. V večdimenzijskih strukturah so to navadno dimenzije kock.
2. Stranske ali pomožne entitete predstavljajo lastnosti posamezne glavne entitete. V relacijskih tabelah so to različni atributi.
3. Relacije: Z relacijami se definirajo povezave, odnosi med entitetami v smislu *Trgovine kupujejo knjige* ali *Ime_avtorja je ime avtorja*. Teh relacij ni možno najti v zbirkah podatkov, definirajo pa določene akcije, ki jih je potrebno izvesti, da se izdela ustrezna poizvedba. Stranske entitete samodejno prevzamejo značilnosti relacij, ki so definirane med glavnimi entitetami. Za glavno entiteto *avtor* so lahko definirane različne stranske entitete, kot so: ime, naslov in telefon. Če je definirana relacija, kot je navedena v primeru *Avtorji dobivajo naročila od trgovin*, lahko tak model odgovori na vprašanje: *Kateri avtorji so dobili naročila s strani prve trgovine*, ali: *Katere so telefonske številke avtorjev, ki so dobili naročila s strani prve trgovine*.

3.3.2.3. Nadgradnja osnovnega modela

Osnovni model, kreiran s "čarovnikom" je potrebno popraviti in nadgraditi z dodatnimi semantičnimi objekti: entitetami, relacijami, slovarji. Razvijalec lahko v osnovi izbere tudi popolnoma prazen projekt, kjer ročno določi že osnovne objekte, lastnosti in povezave.

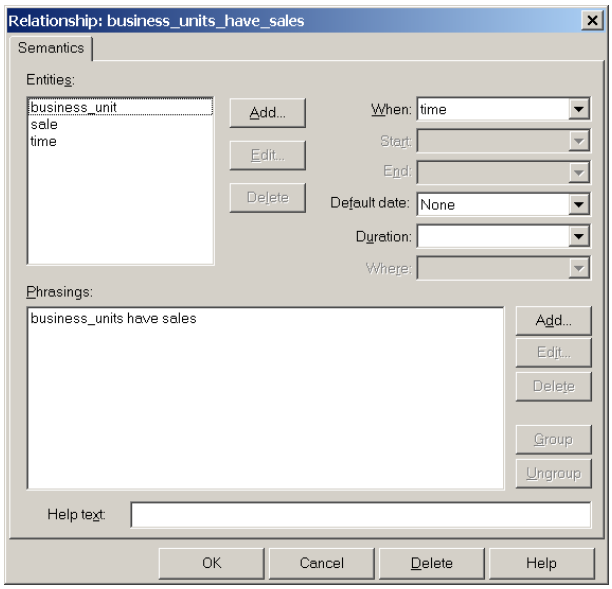
a) Določanje relacij, odnosov med entitetami:

Z relacijami se določajo odnosi med entitetami. Definira se vloga, ki jo igra določena entiteta v nekem modelu. Brez natančno izdelanih semantičnih relacij računalniški model ne bo razumel vprašanj, ki jih zastavlja uporabnik.
Glavni problem večine baz podatkov je ta, da niso ustrezno definirani odnosi med entitetami, ki vladajo v realnem svetu. V bazi podatkov so lahko zapisana imena ljudi in njihova starost, vendar računalnik ne zna odgovoriti npr. na vprašanje *Koliko je mladih ljudi.* Lahko bi rekli, da se z ustreznim definiranem relacij med entitetami poskuša ustvariti čimbolj verodostojno kopijo nekega segmenta realnega življenja.

- Definiranje časovne komponente relacije

Običajno so pri strateških analizah zanimive vrednosti, ki se nanašajo na neko časovno obdobje. Časovno vrsto v EQ modelu praviloma predstavlja svoja entiteta. S pomočjo te entitete se lahko definira začetek, konec in trajanje neke relacije.

Slika 27: Definiranje časovne komponente relacije



Vir: Ekranska slika, Microsoft English Query, Project Explorer

Primer na sliki 27 prikazuje uporabo treh entitet: *poslovna enota* (business_unit), *prodaja* (sale) in *čas* (time). Odnos med poslovno enoto in prodajo je definiran z relacijo *Business_units have sales*, pri čemer je relacija veljavna samo za časovne enote, ki so navedene v dimenziji *time*. Posebne omejitve glede začetka, konca in trajanja relacije niso definirane. S tako definirano relacijo lahko uporabnik dobi odgovore na vprašanja kot so:

1. Show the business units and their sales.
2. Did first business unit have sales?
3. Show sales in 2001.

Če bi bil določen začetni in končni datum za primer relacije *Podjetja prodajajo izdelke*, bi uporabnik lahko spraševal vprašanja, kot so:

1. Kdaj je podjetje A začelo prodajati izdelex X?
2. Kdaj je podjetje A prenehalo prodajati izdelex X?
3. Ali je podjetje A prodajalo izdelek X v letu 1999?

Omejitev trajanja relacije bi bila lahko uporabljena v primeru kreiranja modela za analizo tekaških tekem, kjer bi bila pod področjem *Duration* lahko navedena entiteta, ki vsebuje čase teka. Tako bi model znal pravilno odgovoriti na vprašanja, kot so:

1. Koliko časa je tekla Jožica?
2. Koliko časa je tekel Jože na Ljubljanskem maratonu?
3. Kdo je imel najslabši čas?

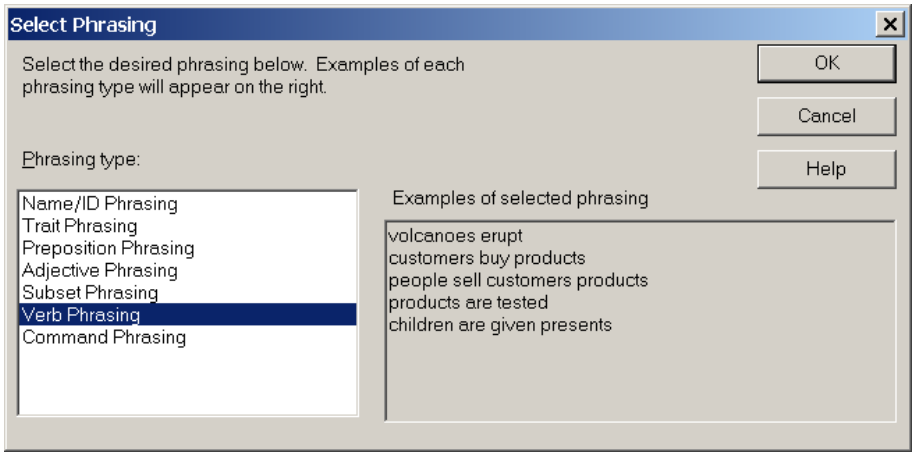
- Definiranje krajevne komponente v relaciji

Podobno kot pri času je možno definirati tudi relacije, s katerimi je možno odgovarjati na vprašanja o kraju relacije. To so vprašanja, ki se običajno začnejo z besedo *Kje*. Primer:

1. Kje je Janez kupil rože?
2. Kje je Ljubljana?
3. Kje so taborili taborniki v letu 2000?

- Tipi relacij, ki jih je možno uporabiti pri kreiranju odnosov med entitetami (slika 28):

Slika 28: Izbira tipa relacije



Vir: Ekranska slika, Microsoft English Query, Project Explorer

1. Name/ID phrasing:

Kreiranje relacije, se katero se definira ime ali identifikacijska številka, npr. *Imena_zaposlenih so imena zaposlenih* ali *Sifra_zaposleni je sifra od zaposlenih*. S tako definirano relacijo je možno odgovoriti na vprašanja, kot so: *Pokaži kupca z imenom Jerca Šmid*, *Prikaži imena zaposlenih*, *Kaj je znanega o Petru Pokori*.

2. Trait phrasing:

Kreiranje relacije, kjer ena entiteta nastopa kot atribut, lastnost, prilastek druge entitete, npr. *Ljudje imajo rojstne dneve*, *Žoge imajo barvo* ali *Pacienti imajo krvne skupine*.

3. Preposition phrasing:

Kreiranje relacije med dvema entitetama, kjer je ena subjekt in je z objektom povezana preko predloga. To so besede, kot so: po, na, med, pred, toda, za, od, čez. V primeru stavka: *Pacienti so na opazovanjih (zaradi bolezni) (za raziskavo),* so *pacienti* subjekt, *na* je predlog in *opazovanjih* objekt. Obstajajo lahko tudi ostale entitete, kot je v zgornjem primeru *zaradi bolezni* ali za *raziskavo*, ki še natančneje določajo relacijo. Model v tem primeru lahko odgovori na naslednja vprašanja: *Kdo je na raziskavi X*, *Kateri pacienti so na opazovanjih zaradi srčnega infarkta*, *Kdo je na opazovanjih za raziskavo Y*, *Kdo je na opazovanjih zaradi srčnega infarkta za raziskavo Y*.

4. Adjective phrasing:

Relacija med entitetami je določena s pridevnikom. Lahko je le ena beseda, ki opiše subjekt, pridevnik je lahko del entitete, lahko je tudi mera, npr. visok, star, hiter. Primer: *Nekaj izdelkov je starejših kot so drugi.* Ob tem se lahko določijo mejne vrednosti za pridevnik *star* in tudi pridevnik za vrednosti, ki ne ustrezajo pridevniku *star*, npr. *nov*, *mlad*. S tako definiranim modelom se dobijo pravilni odgovori na vprašanja: *Pokaži mi stare izdelke*, *Kateri izdelki so novi*.

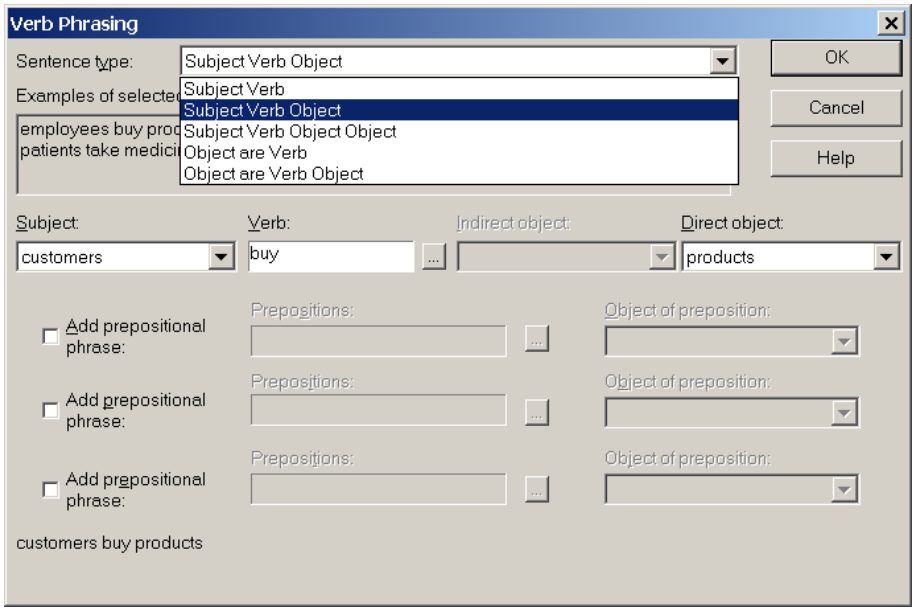
5. Subset phrasing:

Kreiranje relacije, kjer je ena entiteta del druge entitete, kjer objekt predstavlja neko podmnožico subjekta. Lahko je le ena beseda, lahko je to neko polje v drugi entiteti. Primer: *Nekaj zaposlenih je moških*, *Nekateri vrhovi so kamniti*. V tem primeru se lahko zastavljajo vprašanja: *Kateri zaposleni so moški*, *Prikaži kamnite vrhove.*

6. Verb phrasing:

Relacija je določena z glagolom, npr. *Komercialist prodaja*, lahko je definirana s subjektom in direktnim objektom, npr.: *Kupci kupujejo izdelke*, kjer je *izdelek* direktni objekt, ali z indirektnim objektom, npr.: *Komercialisti prodajajo kupcem izdelke*, kjer je *kupec* indirektni objekt (slika 29).

Slika 29: Primer uporabe glagola pri definiranju relacije



Vir: Ekranska slika, Microsoft English Query, Project Explorer

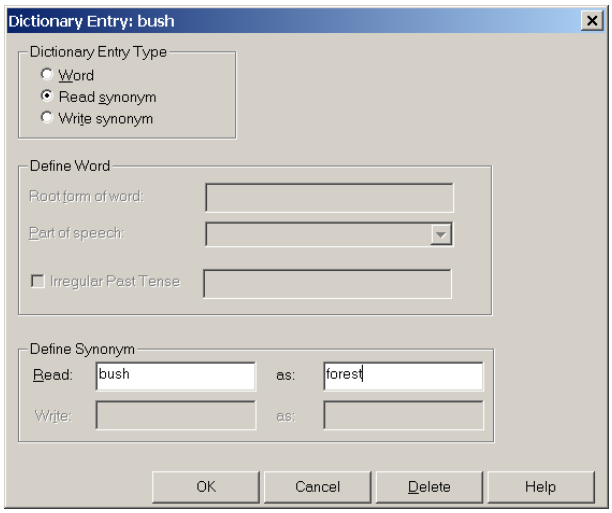
7. Command phrasing:

Relacija je definirana z ukazom, kot je npr. *Natisni datoteko*, *Izhod*, *Naroči izdelek* ali *Pošlji kupcem elektronsko pošto*.

b) EQ slovar

EQ slovar vsebuje tisoče običajnih angleških besed, ki ponavadi zadostujejo za ustrezno interakcijo med človekom in računalnikom. V primeru nekih specifičnih zahtev, kot je npr. medicinsko področje, pa je v EQ slovar možno dodati tudi nove besede, oziroma sinonime za že obstoječe besede.

Slika 30: Primer vnosa novih besed v EQ slovar



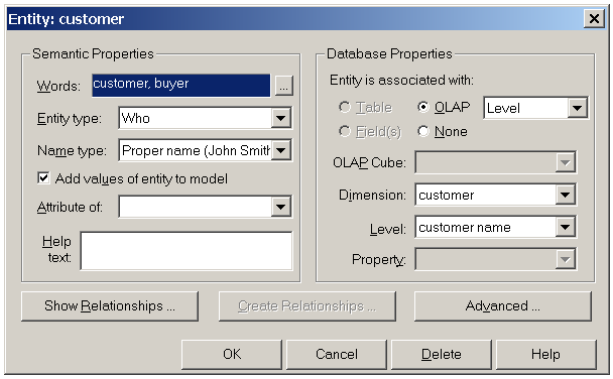
Vir: Ekranska slika, Microsoft English Query, Project Explorer

Poznani so različni načini nadgrajevanja osnovnih zapisov v slovarju. Pri razvoju EQ modula (*.eqm) je v Microsoft Visual Studio urejevalniku možno uporabiti opcijo *Semantic objects* za dodajanje novih besed v slovar. Pri novih besedah je v področju *Part of speech* možno nastaviti, ali je to glagol, samostalnik, pridevnik, prislov. Ravno tako se lahko definirajo posebnosti posameznih besed, npr. drugačna oblika samostalnika v množini ali nepravilni glagoli. Tak primer nepravilne množinske oblike je npr. beseda *person*, ki je v množini *people*.
Možno je navesti sinonim za že obstoječo besedo v slovarju. Ti sinonimi so globalni in prevladajo nad sinonimi, ki so določeni za posamezno entiteto. Zato je potrebno biti previden:

- *Read synonym* (slika 30): Tu se določi način interpretiranja neke besede pri analizi vprašanja. Če naj se beseda *ship* prebere kot *boat*, bo program vprašanje *How many*

boats sailed last week interpretiral kot *How many ships sailed last week.* V tem primeru bo odgovor pravilen. Napačno pa bi bilo v primeru vprašanja: *How many products did we ship last week*, ko bi računalnik poizkušal odgovoriti na vprašanje: *How many products did we boat last week*. V takem primeru je bolje dodati sinonim, ki velja samo za določeno entiteto (slika 31).

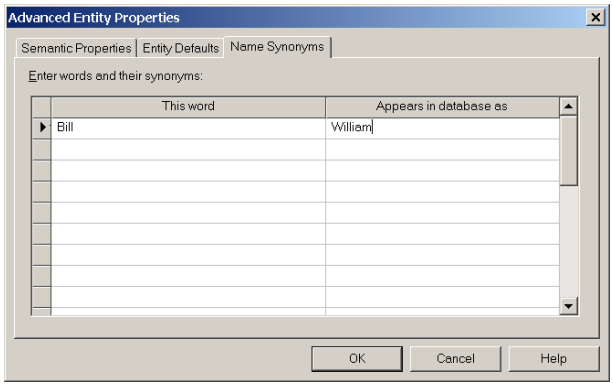
Slika 31: Definiranje sinonimov za entiteto



Vir: Ekranska slika, Microsoft English Query, Project Explorer

- *Write synonym*: Tu se določijo sinonimi, ki jih program uporabi v odgovoru na določeno vprašanje. Primer: Če je določeno *Write employees as associates*, bo računalnik besedo *employees* v vprašanju nadomestil z *associates* v odgovoru.
- Možno je določiti celo sinonime za vrednosti iz baz podatkov. Ta možnost pride v poštev v primerih, ko nekaterih vrednosti v bazi podatkov ni v taki obliki, kot jih uporabnik želi, oziroma so shranjene v obliki, ki so za uporabnika nerazumljive. Tako se npr. ime *William*, ki je zapisano v bazi podatkov, na ekranu prikaže kot *Bill*. Za uporabnika se tako definiran sinonim obnaša, kot da bi bil fizično zapisan v bazi podatkov (slika 32).

Slika 32: Sinonimi za vrednosti iz baz podatkov



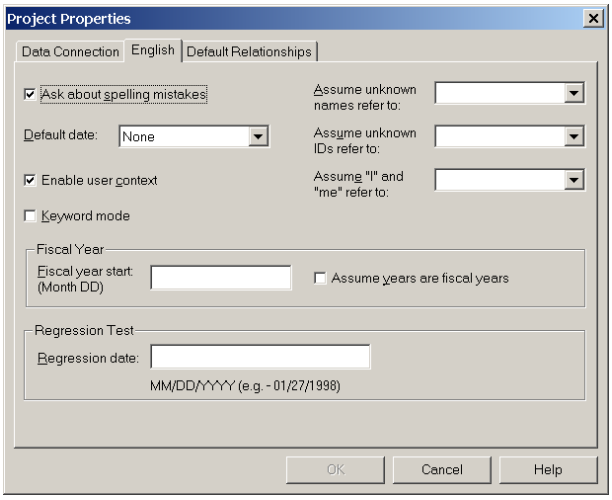
Vir: Ekranska slika, Microsoft English Query, Project Explorer

c) Dodatne lastnosti, ki veljajo za celoten projekt

Pri lastnostih, ki veljajo za celoten projekt, je možno nastaviti tudi stvari, ki omogočajo prirediti projekt različnim zahtevam:

1. Dodatno spraševanje - razjasnjevanje, če se pojavijo slovnične napake. Če ta opcija ni izbrana, bo EQ samodejno poskušal popraviti napake.
2. Privzet datum: Lahko se nastavi privzet datum (leto mesec, dan), če v vprašanju ni posebno naveden. Če je opcija izbrana, bo program vprašanje *Prikaži prodajo podjetja X* razumel kot *Prikaži prodajo podjetja X v konkretnem letu*.
3. Nanašanje na predhodno zastavljena vprašanja in odgovore: Če je najprej zastavljeno vprašanje: *Kdo je bil sprejet v službo lansko leto,* se nato lahko vpraša kar: *Kateri oddelek jih je sprejel*, ker računalnik ve, da se to nanaša na zadnji dobljen rezultat. Podobno velja za primer: *Pokaži prodajo za leto 1996* in nato le *za 1997.*
4. Upoštevanje ključnih besed: Lahko se določi pravilo, da mora program razumeti vsak del vprašanja, ali da le na podlagi besed, ki so mu v vprašanju znane, odgovori. Odgovor na vprašanje: *Pokaži mlade kupce, ki so kupili sadje* je tako v odvisnosti od izbire različen. Če opcija ni izbrana, bo sistem odgovoril, da na podlagi znanih informacij ne more pravilno odgovoriti, ker ne ve, kateri so *mladi kupci*. Ne ve, kaj je to mladost. V drugem primeru - če je opcija izbrana, pa program zanemari objekte in relacije, ki jih ne razume in odgovori na "poenostavljeno vprašanje". Vprašanje razume kot: *Pokaži kupce, ki so kupili sadje*, na katerega pa pozna odgovor.
5. Določijo se lahko privzete vrednosti za neznane objekte in šifre. Prav tako se da navesti, na koga naj se nanaša, če se v vprašanju pojavijo besede, kot so *moj* in *jaz*.
6. Nastavijo se lahko okviri za fiskalno leto, ki lahko vsebuje drugačne časovne vrednosti, kot so nastavljene v časovni entiteti.
7. Definirajo se lahko okviri za regresivna vprašanja. Odgovor na vprašanje: *Kakšna je bila prodaja včeraj* bo vsak dan drugačen, če ne bo določenega konkretnega datuma. S to opcijo se zagotovi, da se bodo testi odzivali enako, če se zastavljajo vprašanja v različnih časovnih obdobjih.
8. Zanimiva je opcija *Default relationships*, kjer je možno vnaprej razjasniti težave, ki lahko nastanejo ob vprašanjih, ki so dvoumna. Ponavadi tak problem nastane v primeru, če je za isto entiteto določeno več različnih relacij. Primer: Če sta definirani relaciji *Poslovne enote proizvajajo izdelke* in *Poslovne enote prodajajo izdelke*, računalnik ob vprašanju uporabnika *Pokaži poslovne enote in njihove izdelke* ne bo vedel, ali uporabnik želi dobiti informacijo o izdelanih ali prodanih izdelkih. Pod opcijo *Default relationships* se lahko nastavi, katero izmed relacij naj model izbere ob uporabi določene kombinacije entitet v vprašanju.

Slika 33: Nastavljanje dodatnih lastnosti za projekt



Vir: Ekranska slika, Microsoft English Query, Project Explorer

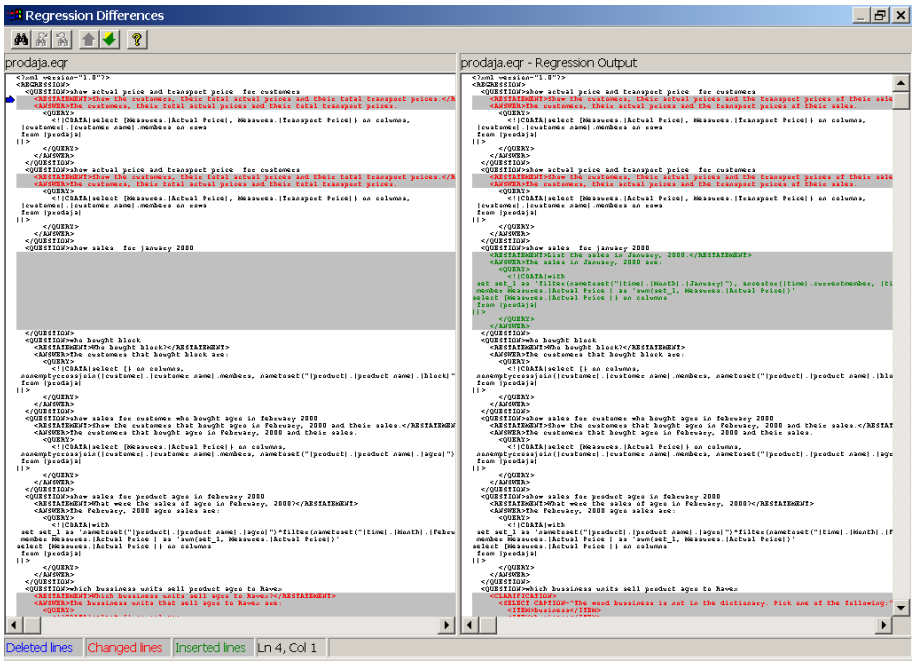
3.3.2.4. Preizkušanje in dopolnjevanje modela

Ko je osnovni model zgrajen, sledi preverjanje ustreznosti. Znotraj EQ je na voljo poseben modul *Model Test*, kjer se preverjajo odgovori na vprašanja, ki so bila definirana na podlagi predhodnih razgovorov z uporabniki. S pomočjo modula se lahko preveri tako pravilnost pretvorbe naravnega jezika v računalniško obliko (SQL ali MDX sintakso) kot pravilnost rezultatov. Za pomoč pri dopolnjevanju, nadgrajevanju in izboljševanju modela je na voljo tudi poseben "čarovnik".
Vprašanja, preoblikovana vprašanja s strani aplikacije in pretvorbo v računalniško sintakso je v XML obliki možno shraniti v regresijsko datoteko (*.eqr), ki se uporablja za kasnejše preverjanje, za uporabo v drugih programih, XML oblika pa je primerna tudi za ročno popravljanje in dopolnjevanje preko Visual Studio Editorja ali drugega urejevalnika besedila.
Kot testna datoteka je lahko pripravljena tudi navadna besedilna datoteka z zapisanimi vprašanji (po eno vprašanje na vrstico). Tudi tako pripravljeno datoteko je nato možno uvoziti kot regresijsko datoteko, program jo zna tudi samodejno pretvoriti v XML obliko.

Preizkus s pomočjo regresijske datoteke se izvede z ukazom *Run regression*. Z opcijo *View output* sistem izdela izhodno datoteko. V njej je poleg vprašanja uporabnika navedeno tudi preoblikovano uporabniško vprašanje v obliko, ki jo razume računalnik, in pretvorba v MDX sintakso. Morebitne razlike med izvorno in izhodno datoteko se primerja z ukazom *View differencees* v področju *Project Explorer*. Odpre se okno, ki z barvami pokaže razlike:

a) Modro obarvane vrstice - zbrisano
b) Zeleno obarvane vrstice - dodane vrstice
c) Rdeče obarvane vrstice - spremenjene vrstice

Slika 34: Preverjanje ustreznosti regresijske datoteke



Vir: Ekranska slika, Microsoft English Query, Project Explorer

V primeru, ki je prikazan na sliki 34, se regresijska datoteka imenuje *prodaja.eqr*. Prikazana je na levi strani okna. Na desni strani je prikazana spremenjena datoteka *prodaja.eqo - Regression Output*, ki nastane po preverjanju ustreznosti zapisov v regresijski datoteki. Ta se na trdi disk zapiše s končnico (*.eqo). Spremenjeno datoteko se kasneje lahko pretvori (promovira) v osnovno regresijsko datoteko. Tako pripravljene informacije o vprašanjih in odgovorih nanje v XML obliki je možno kasneje uporabiti v različnih programih.

Slika 35: Primer regresijske datoteke v XML obliki z enim vprašanjem

```
<?xml version="1.0"?>
<REGRESSION>
 <QUESTION>show actual price and transport price  for customers
  <RESTATEMENT>Show the customers, their total actual prices and their total
transport prices.</RESTATEMENT>
  <ANSWER>The customers, their total actual prices and their total transport prices.
   <QUERY>
    <![CDATA[select {Measures.[Actual Price], Measures.[Transport Price]} on
columns,
 [customer].[customer name].members on rows
 from [prodaja]
]]>
   </QUERY>
  </ANSWER>
 </QUESTION>
```

Vir: Microsoft English Query, regresijska datoteka

3.3.2.5. Kreiranje končne aplikacije

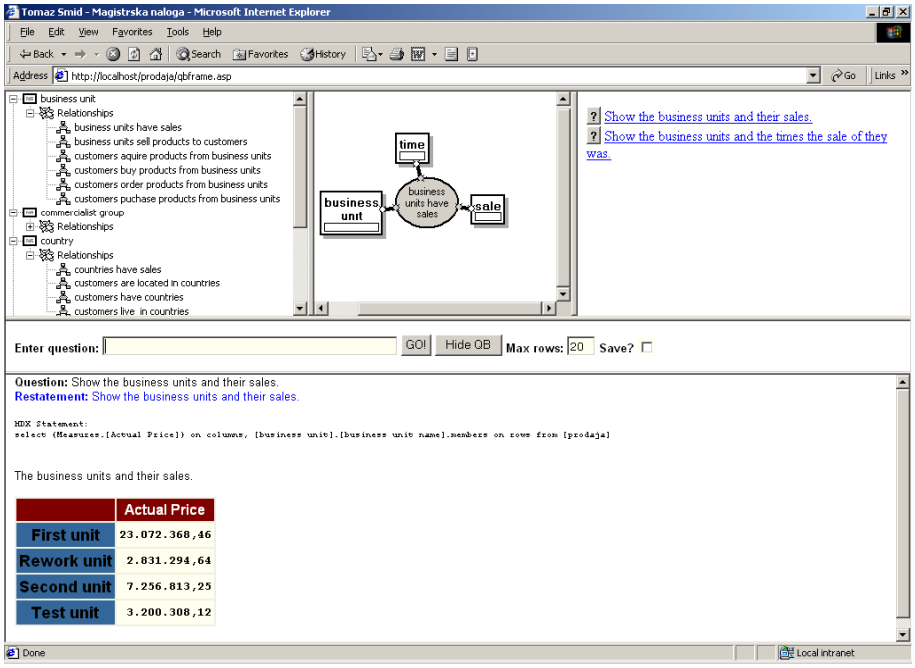
Privzeta končnica za zgrajeno aplikacijo je (*.eqd), ki predstavlja okrajšavo za English Query Domain file. Pripravljena je za delo z določeno verzijo EQ gonilnika (engine). Če je bila datoteka zgrajena s prejšnjo verzijo EQ, je potrebno ponovno povezati in zgraditi (*.eqd) datoteko iz datoteke (*.eqp).

- Graditelj vprašanj (Question Builder)

Question Builder je modul, ki ga razvijalci lahko vgradijo v različne končne aplikacije, ki podpirajo ActiveX komponente, kot so: medmrežne strani, ki uporabljajo ASP komponente, Microsoft Visual Basic aplikacije, Microsoft Visual C++ programi.

Question Builder je posebna Microsoft ActiveX komponenta, ki programerjem omogoča enostavno vgrajevanje English Query modelov v različne končne aplikacije. Uporabnikom take aplikacije pomeni pomoč pri zastavljanju vprašanj. Komponenta je namreč zgrajena tako, da uporabniku na grafičen način prikaže entitete, ki so prisotne v modelu, ter relacije med njimi. Za vsako izmed entitet v modelu lahko uporabnik ugotovi različne stvari, kot so: edninska, množinska oblika entitete, sinonimi. Z enostavnimi "povleci-spusti" (drag and drop) operacijami si uporabnik lahko sam pripravi ustrezne komponente modela, ki je predmet trenutne analize, Question Builder pa mu postreže z različnimi možnimi vzorčnimi vprašanji, ki jih je možno zastavljati.

Slika 36: Primer izgleda Question Builderja



Vir: Ekranska slika, Microsoft Internet Explorer, Question Builder

Graditelj vprašanj je sposoben prikazati dva tipa vzorčnih vprašanj:

1. Generalna vprašanja: Tako vprašanje je npr.: *Show the business units and their sales* ali *Show the business units and the times the sale of they was* (slika 36).

2. Specifična vprašanja: Uporabnik v grafičnem delu prikaza lahko vnese konkretno vrednost iz zaloge vrednosti posamezne entitete, modul pa prikaže različna možna vprašanja za konkretno entiteto. Če bi v primeru, ki je prikazan na sliki 36, za entiteto *business units* izbrali vrednost *First unit*, bi Question Builder predlagal vprašanje: *Did First unit has sales.*

Pravilno delovanje EQ aplikacij poleg datotek, ki so del posameznega EQ projekta, na uporabniških računalnikih zahteva, da je nameščenih tudi nekaj izvedbenih (Run-Time) datotek, razen v primeru, če aplikacija deluje na medmrežnem strežniku. V tem primeru je potrebno namestiti ustrezne komponente le na medmrežni strežnik.

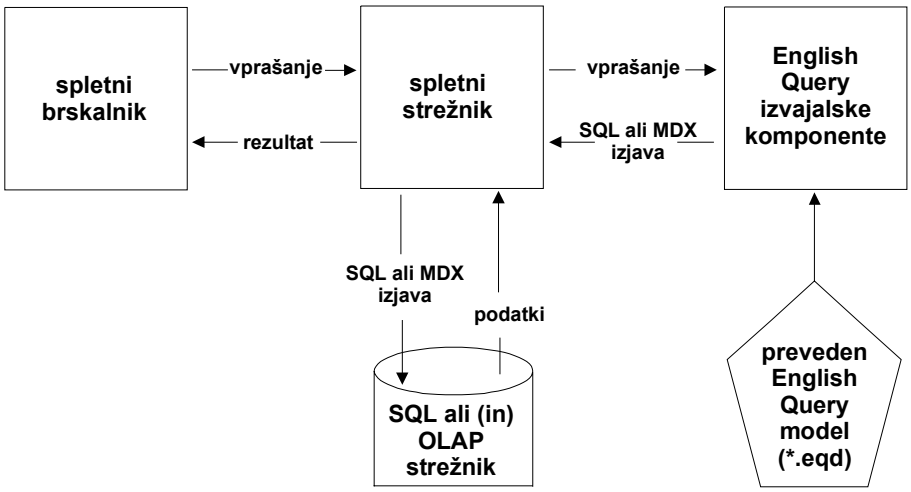
Poleg kopiranja ustreznih datotek je potrebno nekatere komponente tudi registrirati, to je s posebnim orodjem zapisati v sistemski register na odjemalskem računalniku. Če se datoteke nameščajo ročno, je potrebno z orodjem *regsvr32.exe* registrirati datoteke *mseqole.dll* in *mseqgrqb.ocx*.

Vse potrebne datoteke se ob namestitvi programa EQ zapišejo v privzeto mapo *\Program Files\Common Files\System\EQ80*.

Princip delovanja spletne EQ aplikacije:

Končni uporabnik se preko spletnega brskalnika poveže na strežnik ter vpiše ali z govorom poda vprašanje. To vprašanje nato "preda" internet strežniku (Microsoft Internet Information Services) skupaj z ustreznimi naslovi do ASP komponent, ki so sposobne izvajati Microsoft Visual Basic Scripting Edition (VBScript) programe. VBScript program posreduje vprašanje English Query-u. Ta uporabi (*.eqd) datoteko, ki na podlagi znanja iz EQ modela spremeni vprašanje v ustrezno SQL ali MDX obliko in preko Microsoft ActiveX podatkovnih objektov (ADO) naredi poizvedbo v relacijski zbirki podatkov (Microsoft SQL strežnik), ali papreko Microsoft ActiveX večdimenzijskih podatkovnih objektov (ADO MD) naredi poizvedbo v večdimenzijski zbirki podatkov (Microsoft Analysis Services).
ADO in ADO MD objekti rezultate na koncu v ustrezni obliki prikažejo na ekran računalnika.

Slika 37: Princip delovanja spletne EQ aplikacije



Vir: SQL Server Books Online

3.4. Lernout & Hauspie Voice Xpress - Program za sintezo in razpoznavanje govora

3.4.1. O podjetju

Lernout & Hauspie Speech Products je bilo do pred kratkim eno od vodilnih podjetij v svetu za izdelavo programske opreme za razpoznavanje govora. Prisotno je bilo na različnih področjih: v računalniški, avtomobilski, telekomunikacijski industriji, v zdravstvu, na področju svetovanja, prava, v izobraževalne namene.

NPD INTELECT - eno izmed vodilnih neodvisnih podjetij na področju tržnih raziskav je o podjetju zapisalo:

"L&H je v kategoriji programske opreme za razpoznavanje govora na 35 milijonov vrednem ameriškem trgu trenutno vodilno podjetje z več kot 70% tržnim deležem (v 12 mesečnem obdobju, končanem 31.3.2001). " (NPD INTELECT, 7.6.2001)

Nekateri izmed zadnjih zelo uspešnih projektov podjetja Lernout & Hauspie so:

a) Program iChart, medmrežni program za področje zdravstva, za predpisovanje receptov, prikaz informacij o pacientih. (L&H, 21.8.2000).
b) Vključitev modula za pretvorbo teksta v govor v portugalski Telecom Inovação (L&H, 30.5.2001).
c) Programi za avtomobilsko industrijo (govorna komunikacija z navigacijskim sistemom, prenosnim telefonom, dlančnikom. (L&H, 2002)
d) Velik dosežek je izdelava programa za pretvorbo besedila v govor za kitajski jezik. Program je sposoben pretvarjati v govor besedilo, ki je zapisano v angleškem in različnih oblikah kitajskega jezika. Taka "bilingvistična" sinteza govora je prva te vrste (L&H, 15.11.2001).

V letu 2000 je podjetje L&H prevzelo eno izmed vodilnih podjetij za razpoznavanje govora, to je Dragon Systems Inc., (L&H, 28.3.2000). Ker je bil to včasih resen tekmec z zelo dobrimi izdelki, so vodilni možje v podjetju napovedali, da bo v bližnji prihodnosti možno pričakovati združitev najboljših funkcij programov za razpoznavanje govora obeh podjetij v en sam program, ki naj bi dosegal še boljše rezultate na trgu kot do sedaj, kar navsezadnje potrjujejo tudi mnoge nove pogodbe z zdravstvenimi ustanovami, telekomunikacijsko in avtomobilsko industrijo, podpisali pa so tudi sodelovanje s podjetjem Yahoo.

Združeno podjetje je zaradi nepravilnosti v poslovanju zašlo v krizo, kar je izkoristilo podjetje ScanSoft, ki je konec leta 2001 prevzelo glavna področja delovanja podjetja Lernout & Hauspie (L&H, 12.12.2001). Poslovne nepravilnosti pa ne zmanjšujejo tehničnih kvalitet programov, ki so trenutno sigurno med najnaprednejšimi.

Videti je, da je področje računalniškega razpoznavanja govora zelo perspektivna dejavnost, ki pa še ni docela izoblikovana. V smislu sedanjih trendov globalizacije je zato pričakovati nadaljnje prevzeme in združitve podjetij, ki se ukvarjajo z izdelki za računalniško razpoznavanje govora.

3.4.2. O programu Voice Xpress 5.0

Povzeto po "PC Magazine, 1.8.2000":

Podjetje Lernout & Hauspie z vsako novo verzijo programa Voice Xpress izboljšuje kvaliteto razpoznavanja govora in razširja možnosti uporabe. Verzija 5 vsebuje mnogo novih dodatkov, ki omogočajo precej več kot le golo diktiranje besedila.
Podobno kot pri verziji 4, si moramo pred uporabo programa vzeti okoli 10 minut časa, da "naučimo" - personaliziramo program. Potrebno je navesti nekaj osnovnih podatkov, kot so npr. ime, spol, starost. Potrebno je izmeriti jakost šumov v okolici in nastaviti mikrofon. V paketu je pri verziji 5.0 že dodan tudi kvaliteten Plantronicsov naglavni set (mikrofon + slušalka), ki zagotavlja ustrezno kvaliteto zajemanja govora. Po osnovnih testih je potrebno prebrati vnaprej pripravljena besedila, da se program privadi na hitrost, barvo in ostale posebnosti posameznikovega govora.
Po takih nastavitvah lahko po zagotovilih proizvajalca kvaliteta razpoznavanja govora dosega okoli 96%, v določenih primerih celo do 98%, le pri zahtevnejših besedilih pade na 95%. Ker se program sproti uči novih besed, se z večkratno uporabo kvaliteta razpoznavanja le še povečuje.

Z novim modulom *Nothing But Speech* je dodana nova kvaliteta, ki zagotavlja, da program ne sprejema nezaželenih glasov, kot so *ah* ali *uhm*.
Že napisane dokumente je možno vnesti v modul *Accuracy Builder*, ki preveri vsebino in se nauči besed, ki jih še nima v slovarju.

Podobno kot IBMov ViaVoice Pro Millennium Edition, tudi Voice Xpress omogoča nadzor vseh funkcij osebnega računalnika, tudi namizje in ostale funkcije operacijskega sistema Windows: aplikacije, menije, iskalce po spletu, urejevalnike elektronske pošte.
Dodani so tudi nekateri uporabniški programi, kot so Web-centric, Now You're Talking in Voice Xpress Personal Finance.
Uporaba različnih ukazov za nadzor programa s pomočjo govora včasih vzbuja strah pri uporabnikih, da ne bi z napačnim ukazom zgubili kakšnega dokumenta ali naredili kakšno drugo neumnost, zato je vedno na voljo spisek govornih ukazov, ki jih je možno uporabiti v določenem trenutku.

Modul *Voice Xpress Café* je dobrodošel pripomoček za začetnike, saj lahko na voden način spoznavajo možnosti programa, oziroma naredijo mnogo vaj v smislu bolj učinkovitega dela s programom.

Dodelan modul *RealSpeak* daje novo dimenzijo računalniškim aplikacijam. Možno ga je uporabiti kadarkoli za pretvorbo označenega pisanega besedila v govor. Sedaj je še bolj izpopolnjen kot v preteklosti, saj je govor neprimerno bolj "človeški", za razliko od predhodnih, kje je bil predvajan govor zelo sintetičen. To dokazuje tudi podpis pogodbe z

najbolj rastočim angleškim mobilnim operaterjem Orange, ki bo tehnologijo *RealSpeak* vključil v bodoče aplikacije.

Zaradi sodelovanja podjetja Lernout & Hauspie z Microsoftom je program še posebej prilagojen za delo z Microsoft Office 97/2000.

Zanimiva je tudi opcija mobilnega nareka *Talk and Go*, ki omogoča snemanje lastnega glasu na različne mobilne snemalne naprave, kasneje pa napravo priključiti na računalnik in sprožiti postopek samodejnega pisanja v urejevalnik besedila, saj je program sposoben spremeniti vnaprej posnet govor v pisano besedilo.

L&H Voice Xpress Professional vsebuje tudi možnost uporabe dodatnih slovarjev za specifične namene uporabe, kot so: ekonomija in finance, zdravstvo in tehnologija.

3.4.3. Definiranje profila uporabnika

V smislu čimbolj uspešnega razpoznavanja govora je na uporabniškem računalniku najprej potrebno ustvariti profil uporabnika. Uporabniku lasten profil se ustvari na podlagi niza korakov, ki jih predlaga program L&H Voice Xpress Professional 5.0. Ti koraki so:

1. Identifikacija uporabnika:

   a) Določitev imena uporabnika
   b) Spol
   c) Starost: pod ali nad 16 let
   d) Tip vnosne naprave: Najbolje je uporabiti naglavni komplet, to je slušalke in mikrofon, ki je priložen programu pri nakupu programske opreme.
   e) Izbira slovarja, ki bo uporabljen skupaj s kreiranim profilom, če je na strani uporabnika nameščenih več slovarjev. Program za razpoznavanje govora se tako lahko uporablja enkrat v medicinske namene in kasneje za izdelavo pravno zapletenih poslovnih pogodb. Ker se izrazoslovje precej razlikuje, je možno uporabiti različne slovarje z besedami, ki so lastne posameznim specifičnim področjem uporabe. Pravilna izbira slovarjev lahko bistveno prispeva h kvaliteti računalniškega razpoznavanja človeškega govora.

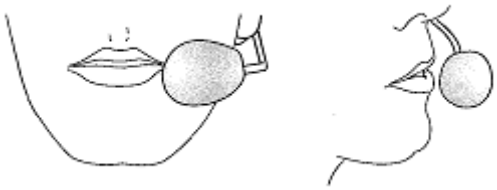
2. Priprava mikrofona:

   a) Izbira mikrofona in pravilen priklop na zvočno kartico: Najbolje je uporabiti naglavni set, ki je ob nakupu že priložen programu. Naglavni set vsebuje slušalke in mikrofon, ki se vključijo v ustrezno vtičnico na zvočni kartici računalnika. Ne priporoča se uporaba vgrajenih mikrofonov, kot jih lahko zasledimo npr. v prenosnih računalnikih ali računalniških ekranih, niti uporaba raznih podaljškov ali

razdelilcev, ki bi lahko oslabili in popačili vhodni signal do stopnje, ki onemogoča kvalitetno zaznavo in zajem govora.

b) Položaj mikrofona: Mikrofon je potrebno namestiti nekoliko ob ustih (slika 38), da je oddaljenost ust od mikrofona stalno enaka (enaka jakost govora) in da preprečimo motnje signala ob izgovorjavi določenih črk, kot sta npr. črki *p* ali *s*.

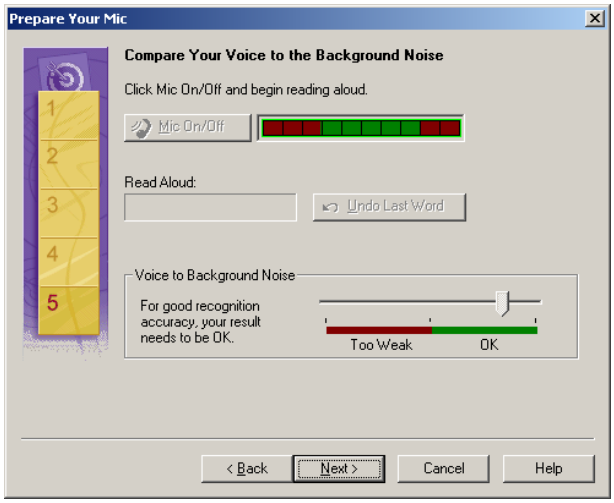
Slika 38: Pravilna namestitev naglavnega mikrofona



Vir: Skenirana slika, L&H Voice Xpress navodila

c) Nastavitev mikrofona: Potrebno je nastaviti občutljivost signala glede na jakost govora, da je signal čim boljši, vendar ne toliko močan, da bi prihajalo do popačenj. Eden izmed pomembnih dejavnikov je tudi definiranje "tišine" oziroma jakost šumov, ki so prisotni v okolici, ko uporabnik ne govori.

Slika 39: Nastavitev mikrofona in šuma okolice



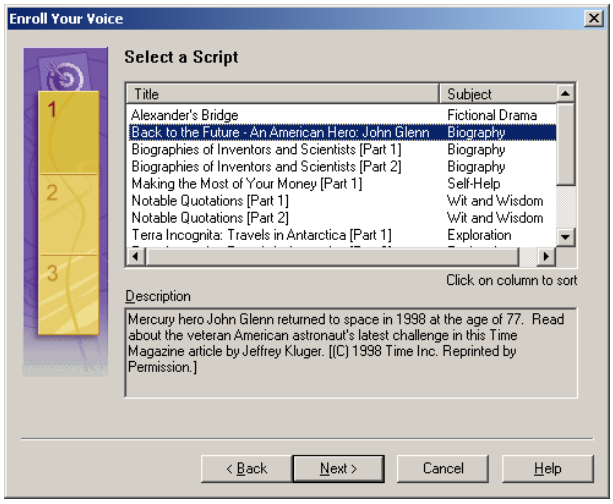
Vir: Ekranska slika, L&H Voice Xpress

3. razpoznavanje značilnosti govora:

Z nekajminutnim branjem različnih besedil program ugotavlja značilnosti govora, kot so: hitrost, presledki med besedami, poudarki in govorne napake. Uporabnik lahko izbira

naslove različnih žanrov. Zaželeno je večkratno branje različnih besedil take vsebine, ki je blizu izrazoslovju, ki ga bo uporabnik uporabljal pri svojem delu.

Slika 40: Izbor besedila za "učenje" programa za razpoznavanje govora



Vir: Ekranska slika, L&H Voice Xpress

Napačno razpoznane besede je možno kadarkoli popraviti in izpopolniti program, da jih bo kasneje pravilno razpoznal. Prav tako je možno na različne načine enostavno vstavljati nove besede v slovar. S časom profil govorca postaja vedno bolj definiran, kar vpliva na povečanje pravilnosti razpoznavanja govora.

Pred uporabo razpoznavanja govora se je potrebno naučiti tudi kodeksa govora - načina komuniciranja z računalnikom. Potrebno je poznati nabor ukazov, ki jih program uporablja pri interakciji z različnimi programi na računalniku. Z govorom je možno upravljali stvari kot so:

a) Določanje oblike besed (poudarjeno, podčrtano)
b) Določanje pozicije na ekranu (nova vrstica, nazaj en znak, na začetek strani)
c) Izbor besed (osvetlitev določenega dela, ki bo predmet urejanja)
d) Kopiranje in uporaba ostalih ukazov za urejanje besedila
e) Pisanje elektronskih sporočil
f) Uporaba funkcijskih, smernih in ostalih kontrolnih tipk na tipkovnici
g) Datumi, valute, telefonske številke

- Primer nareka enostavnega teksta v Microsoft Wordu z uporabo programa Voice Xpress 5.0 (odebeljeno so označeni ukazi za določanje ločil in pozicije delov besedila na dokumentu):

dear mister roberts ***colon new paragraph*** *how are you* ***question mark*** *I am looking forward to seeing you in new york on tuesday* ***comma*** *july twenty fourth two thousand at ten o'clock* ***period*** *please call my assistant at seven eight one four three zero two to confirm* ***period new paragraph*** *sincerely* ***comma new line*** *cris jones*

Opcije v opravilnih vrsticah se enostavno izbirajo z ukazi *Show*, *Open* ali *View*, katerim sledi opcija iz menija.

- Program omogoča štiri različne načine komuniciranja z računalniškimi programi:

    a) *Normal* (*Dictation* + *Command*): To je privzeti način dela, ki uporabniku istočasno omogoča narek besedila in upravljanje z ukazi iz menijev.
    b) *Dictation only*: Ta način se uporablja v primeru daljših narekov besedila, npr. v urejevalnikih besedila ali pri pisanju elektronske pošte. Primernejši je zaradi tega, ker ni potrebno paziti na rezervirane ukaze, ki bi v primeru načina *Normal* lahko pomenili izvedbo določene operacije in ne zapisano besedo.
    c) *Command only*: Z vklopom tega načina dela bo program govor poskušal pretvoriti v ukaze za delo s programi.
    d) *Spelling only*: Vseh imen, oznak, okrajšav, program nima zabeleženih v slovarju, zato je potrebno določen niz znakov črkovati.

V primeru slabega razpoznavanja govora, prevelikega šuma v okolici, želji po večji zasebnosti in podobno, je program Voice Xpress možno kadarkoli izklopiti, delati naprej s klasičnimi vmesniki (tipkovnica, miška, elektronska peresa) in razpoznavanje govora kasneje zopet vklopiti s pritiskom na gumb *Mic On/Off*, ukazom *Ctrl+Alt* ali besedami *stop listening* oziroma *listen to me*.

Da trud izdelave ustreznega profila govorca v primeru menjave ali odpovedi računalnika ne bi bil zaman, je celoten posameznikov profil možno shraniti, izvoziti in ga kadarkoli uvoziti v drug računalnik.

Dobro pripravljen profil uporabnika je pogoj za kvalitetno uporabo govorne komunikacije med človekom in računalnikom, zato ga je potrebno prenesti na vse tiste računalnike, kjer se bo uporabljala aplikacija z govornim vmesnikom.

4. OPIS IZDELANE APLIKACIJE

V smislu natančnejše ponazoritve koncepta rešitve, ki je opisana v dosedanjih poglavjih, sem izdelal tudi konkretno aplikacijo. Primer je namenjen vodstveni strukturi za namene kreiranja analiz prodaje podjetja. Kljub specifičnosti področja je primer možno uporabiti kot predlogo za izdelavo rešitev na sorodnih področjih.

4.1 Značilnosti

Primer je izdelan na podlagi dejanskega stanja srednje velikega slovenskega proizvodnega podjetja. Ime podjetja, nazivi kupcev, poslovnih enot, vrednosti prodaje in ostali podatki so izmišljeni. Podjetje bom v nadaljnjih opisih imenoval "Center d.d."

Podjetje Center d.d. prodaja različne vrste gradbenega materiala. Proizvodnja se odvija v štirih ločenih poslovnih enotah. Izdelke prodaja v različne države. Komercialisti so združeni v več različnih prodajnih skupin. Vodstvo podjetja zanima delež prodaje posamezne komercialne skupine in ne vsakega komercialista posebej.
Podjetje hrani transakcijske podatke v relacijski zbirki podatkov. Tabele so pripravljene v obliki, ki omogoča enostavno izdelavo skladišča podatkov, oziroma podatkovnega trga za prodajno področje.

4.2. Platforma

a) Strojna oprema:

- V prikazan primeru je bil uporabljen le en osebni računalnik, tako za potrebe namestitve strežniških komponent kot za prikaz uporabe s strani končnega uporabnika. Računalnik je imel naslednjo zgradbo:
  - procesor Intel Pentium 4 hitrosti 1,6 MHz,
  - 392 MB RAM
  - 20 GB trdi disk
- Zvočna kartica
- Naglavni set podjetja Plantronics (slušalke + mikrofon)

b) Programska oprema:

Pri izdelavi rešitve je bila uporabljena naslednja programska oprema:

- Microsoft Windows 2000 Server
- Microsoft SQL Server 2000

- Microsoft Analysis Services
- Microsoft English Query
- Microsoft Internet Information Services 5.0
- Microsoft Office 2000
- Lernout & Hauspie Voice Xpress 5.0 Professional

4.3. Skladišče podatkov

Skladišče podatkov se imenuje *Prodaja*. Velikost zbirke, ki je nameščena na Microsoft SQL strežniku, je okoli 15 MB. Zgradba skladišča podatkov je zelo enostavna: Sestavlja jo 6 tabel, od tega ena tabela dejstev in 5 podpornih dimenzijskih tabel - šifrantov.

4.3.1. Tabela dejstev

Tabela dejstev se imenuje *sales_fact* in je delno normalizirana. Vsaka vrstica v tabeli dejstev oziroma prometni tabeli predstavlja informacijo o eni prodajni transakciji, o eni poziciji na računu. Tabelo sestavlja skoraj 35.000 vrstic, kar za podjetje Center d.d. predstavlja prodajo izdelkov za celotno leto 2000 in del leta 2001. Tabela dejstev je sestavljena iz sledečih polj - atributov:

a) *Date*: Datum. Predstavlja datum potrditve prodaje določenega izdelka.
b) *Commercialists_group_ID*: Šifra prodajne skupine. Prodajna skupina je umetna tvorba, kjer so združeni komercialisti in prodajni referenti, ki prodajajo nek zaokrožen prodajni program izdelkov, oziroma prodajajo na neko zaokroženo geografsko področje. V konkretnem primeru je bil podatek o uvrstitvi določenega komercialista v posamezno komercialno skupino vnaprej določen in vse potrebne informacije o računu že shranjene v transakcijski bazi podatkov. Če tega podatka ne bi imeli, bi se moralo prirejanje določene šifre delavca ustrezni številki komercialne skupine in zapis v tabelo dejstev predhodno izvesti s pomočjo različnih transformacijskih orodij, npr. DTS orodjem, ki se namesti skupaj z SQL strežnikom.
c) *Employe_ID*: Šifra delavca. V to polje se zapisujejo matične številke komercialistov, ki so kreirali račun. Čeprav vodstvo podjetja trenutno te informacije ne potrebuje, je zaradi morebitne kasnejše nadgradnje analiz smiselno določene podatke pustiti v tabeli kot "rezervo". Dober razvijalec mora večkrat predvideti tudi dogajanja v prihodnosti, saj ustrezno pripravljen model, ki omogoča hitro prilagajanje, pomeni velik prihranek pri času in denarju. Šifra delavca je v tabelo dejstev torej vključena v smislu morebitnega kasnejšega ugotavljanja delovne uspešnosti posameznega komercialista, optimizacije njegovega dela. Pri tem se lahko pojavlja problem verodostojnosti podatkov. V posameznih področjih

prodaje se delo lahko deli med več ljudi, kar pomeni, da ni nujno, da nek račun res izdela tisti komercialist, ki je tudi dejansko pridobil kupca. V tej smeri je potrebno biti zelo pozoren, še posebno v primeru, če bi se na podlagi števila ali celo vrednosti prodaje odločalo o nagrajevanju delavcev. Neprimeren ocenjevalni model bi lahko povzročil veliko slabe volje.

d) *Country_ID*: Šifra države v katero je bila izvršena prodaja
e) *Product_ID*: Šifra komercialne klasifikacije izdelka
f) *Business_unit_ID*: Šifra poslovne enote
g) *Customer in Customer_branch*: Nazivi kupcev in podružnice, oziroma podenote kupcev. Tu so v tabelo dejstev zapisana kar imena. Ti dve polji nista normalizirani - ni šifer kupcev, ki bi bili kot tuj ključ povezani s pomožno tabelo - s šifrantom. V primeru, da se skladišče podatkov analizira s pomočjo OLAP kocke, je prednost take strukture nabor kupcev, ki se poveča šele takrat, ko bi bila za novega kupca opravljena prva prodaja. Pri pregledih je tako zagotovljeno manjše število elementov na posameznem nivoju dimenzije. S takim vodenjem kupcev se lahko lažje opazi dinamika naraščanja novih kupcev. Slabost take strukture se pokaže v primeru, če se spreminjajo imena kupcev. V tem primeru Analysis strežnik spremenjeno ime zazna kot nov niz znakov in v OLAP kocki prične z novim agregiranjem prodajnih vrstic v tabeli dejstev. Težko je reči katera opcija je boljša, to je odvisno od situacije. V nekaterih primerih želimo, da se vrednosti prodaje določenega kupca vodi ločeno, če se spremeni ime. S tem lahko ohranimo nekaj več zgodovinskih dejstev. Recimo, da se spremeni ime kupca iz *MEZIC d.d.* v *MEZIC d.o.o.*. Lahko vidimo, da se je s spremembo imena spremenila tudi vsebina - oblika gospodarske družbe, zato hočemo novo družbo beležiti kot novega kupca. Če bi pri določenih pregledih želeli oba obravnavati kot enega, bi to lahko storili samodejno s transformacijskimi orodji, ali pa bi jih uporabnik po potrebi ročno združeval v svojih pregledih. V primeru uporabe normalizirane oblike tabele dejstev, kjer bi v tabelo vpisovali le šifre kupcev in podružnic, bi zaradi tega, da tabela kupcev ne bi vsebovala več tisoč imen, morali dodati še en atribut, kamor bi se v primeru prodaje novemu kupcu zapisal status aktivnosti. V kocko bi potem vključili le aktivne. Rešitev tega problema je torej možna na več načinov.

V tabeli dejstev se vodijo kupci v dveh ločenih poljih; kot *kupec* in kot *podružnica* zaradi več razlogov:

- V polje *Customer* se zapisujejo le imena tistih podjetij, ki so v smislu prodaje izdelkov strateškega pomena. V realnosti so to ponavadi podjetja, katerim se prodaja veliko izdelkov. Za taka podjetja se izdelajo (lahko tudi v drugi OLAP kocki) planirane vrednosti prodaje za bodoča časovna obdobja, popusti in ostale stvari, ki definirajo prodajno politiko podjetja. Z vidika analize tako postane zanimiva možnost kreiranja navidezne OLAP kocke, ki bi omogočala primerjanje planirane in dejanske vrednosti prodaje na podlagi različnih OLAP kock.

- "Nestrateške" kupce se zabeleži tako, da se kot vrednost polja *Customer* zapiše besedo *Other companies,* dejanska imena podjetij pa kot vrednost polja *Customer_branch*. S tem je pri pregledu OLAP kocke zagotovljeno, da se na prvem nivoju pregleda kupcev pojavi poleg imen strateških kupcev tudi kupec *Other companies*, ki združuje vrednosti vseh - lahko tudi več tisoč malih kupcev, katerih prodajne vrednosti so pri strateških analizah redkokdaj zanimive. Če vseeno obstaja tovrstna potreba po analiziranju, je še vedno možno "vrtanje v globino", kjer se na drugem nivoju prikaza kupcev vidi analitičen prikaz tudi vseh nestrateških kupcev.
- Dvonivojsko beleženje kupcev je pomembno tudi v primeru prodaje v različna podjetja ali podružnice, ki jih uporabnik želi na prvem nivoju obravnavati pod enotnim imenom. Če bi šlo npr. za prodajo nekega prehrambnega izdelka, bi lahko na prvem nivoju določili ime *Mercator skupaj*, z vrtanjem v globino pa prišli do prodajnih vrednosti za posamezne podružnice, npr *Mercator Sora*, *Mercator Šiška*, *Mercator Škofja Loka*. Pravila za združevanje posameznih podjetij v neko novo tvorbo bi bilo potrebno predhodno opraviti že pri transformaciji podatkov iz transakcijske baze podatkov.

h) Mere: V tabeli dejstev so zapisani tudi količinski in vrednostni podatki za vsako prodajno vrstico na računu. Beležijo se naslednje mere:

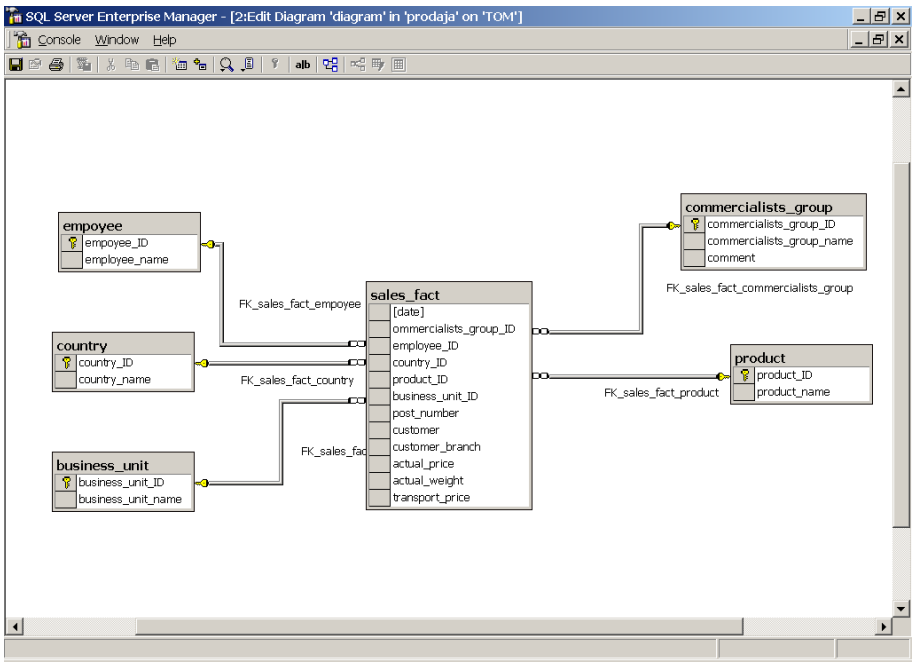
- *Actual_price*: Prodajna cena izdelka, izražena v evrih. To je tako imenovana neto cena, kjer je prikazana prodajna vrednost izdelka brez vključene cene prevoza.
- *Actual_weight*: Teža prodanega izdelka v tonah.
- *Transport_price*: Vrednost prevoza. Zaradi ločene vrednosti prodanega izdelka in prevoza se lahko določi delež prevoznih stroškov pri prodaji. Celotna vrednost prodaje je enaka *Actual_price + Transport_price*. Zanimivo pri tem primeru je dejstvo, da je pri analizah prodaje za zadnje dni neto vrednost prodaje enaka bruto vrednosti. To je posledica do 30 dnevnega zamika plačila transportnih stroškov avtoprevoznikom. Pri analizi prodaje zadnjih prodanih izdelkov delež transportnih stroškov v celotni prodaji tako še ni znan. Pravilna razmerja so vidna za dogodke izpred enega meseca ali starejše.

4.3.2. Podporne tabele - šifranti

Šifranti so tabele, ki se ne spreminjajo pogosto. Te tabele so večkrat zelo enostavne, z malo atributi in vrsticami. Skladišče podatkov v opisanem primeru vsebuje 5 podpornih tabel in sicer:

a) *Country*: Tabela držav v katere podjetje prodaja svoje izdelke. Ker nas zanima prodaja le v strateške države, v tabeli niso navedene vse države. Države, kjer je prodaja zelo majhna, so združene pod imenom *Other*. Pri gradnji skladišča podatkov je potrebno poskrbeti, da se pri zapisih v tabeli dejstev izvede pretvorba določenih šifer držav iz transakcijskih baz podatkov v umetno šifro, ki prvotno ime države spremeni v *Other*. S tem se zagotovi večja preglednost in manjše število vrednosti na posameznem nivoju dimenzije v OLAP kockah.
b) *Employee*: Seznam komercialistov in komercialnih referentov, ki so izvršili prodajo, oziroma tistih, ki so kreirali račun. Lahko je to kar tabela zaposlenih, povzeta iz kadrovske baze podatkov.
c) *Commercialists_group*: Tabela s komercialnimi - prodajnimi skupinami. Prodajne skupine so sestavljene iz enega ali več komercialistov ali komercialnih referentov. Namesto posameznikov je torej govora o "teamu" komercialistov in referentov, ki delajo za skupni cilj. Tekmovanje glede čim boljše prodaje je z nivoja posameznika preneseno na nivo prodajnih skupin.
d) *Product*: Tabela vrst izdelkov. Tu so navedene vse komercialne klasifikacije izdelkov, ki jih podjetje prodaja.
e) *Business_unit*: Tabela poslovnih enot. Nek izdelek lahko prodaja ena ali več poslovnih enot. Veliko podjetij je v današnjem času globalizacije organizirano kot skupek geografsko oddaljenih poslovnih enot, ki v mnogih primerih proizvajajo ali prodajajo enake izdelke. Dobra analiza prodaje, stroškovna in druge analize med različnimi poslovnimi enotami so zato velikega pomena pri strateških analizah glede optimizacije poslovanja ali npr. širitve podjetja.

Slika 41: Diagram tabel in povezav v skladišču podatkov



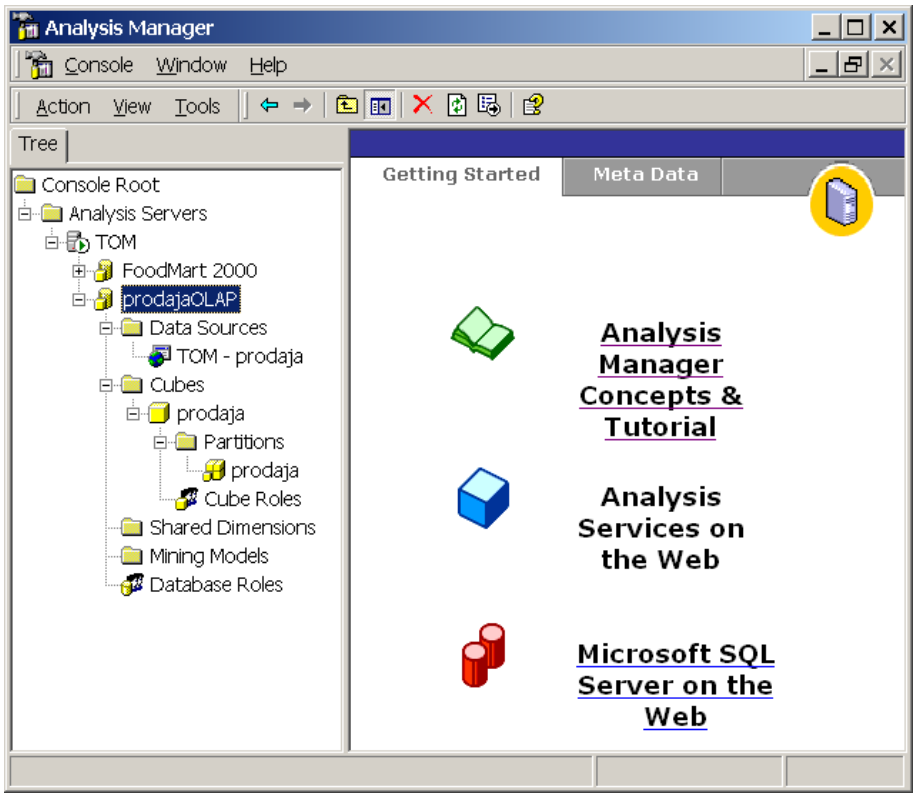
Vir: Ekranska slika, Microsoft SQL Enterprise Manager

4.4. Kreiranje OLAP kocke

V Microsoft Analysis Serverju sem kreiral večdimenzijsko zbirko z imenom *prodajaOLAP*. Taki zbirki je potrebno določiti povezavo do izvornih podatkov. V konkretnem primeru je s pomočjo gonilnika "Microsoft OLE DB provider for SQL server" definirana povezava do relacijske zbirke, to je skladišča podatkov z nazivom *Prodaja* na Microsoft SQL strežniku z imenom *Tom*. Ker je Microsoft Analysis Server samostojen strežnik, ni odvisen od uporabe izključno Microsoftovih relacijskih baz. Skladišče podatkov bi bilo možno implementirati tudi s produkti drugih proizvajalcev, npr. z Oraclom.
Znotraj večdimenzijske zbirke *prodajaOLAP* je možno izdelati več OLAP kock. Za izdelavo rešitve je bilo dovolj kreiranje le ene kocke. Poimenoval sem jo *Prodaja*.

Uporabljena je bila le ena particija. Ob zahtevnejših strukturah bi bilo smiselno izdelati kocko, ki bi bila zgrajena iz več delov - particij. Posamezne particije bi vsebovale večje dimenzije, kar bi omogočalo ločeno kreiranje agregatov (lahko tudi na različnih računalnikih), parcialno procesiranje, osveževanje podatkov le za določeno particijo, ki je vključena v kocko, in podobno. Pravilno izbrana struktura OLAP kocke je izrednega pomena glede performančnih značilnosti poizvedb.

Slika 42: Izdelava OLAP kocke *Prodaja* v Analysis Managerju



Vir: Ekranska slika, Microsoft Analysis Manager

4.4.1. Izdelava dimenzij kocke

Kocka *Prodaja* je sestavljena iz šestih dimenzij:

1. *Business unit*: Dimenzija poslovnih enot. Izdelana je na podlagi polja *Business_unit_name* iz tabele *Business unit*. Dimenzija omogoča spremljanje prodaje podjetja Center d.d. ločeno po poslovnih enotah. V konkretnem primeru so to štiri poslovne enote.
2. *Commercialist group*: Dimenzija prodajnih skupin. Izdelana je na podlagi polja *Commercialist_group_name* iz tabele *Commercialist_group*. Ugotavlja se uspešnost prodaje različnih prodajnih skupin in ne posameznega komercialista ali prodajnega referenta.
3. *Country*: Dimenzija držav: Dimenzija je ravno tako kot prejšnji dve enostavna, enonivojska. Zgrajena je na podlagi polja *Country_name* iz tabele *Country*. Pod imenom *Other* je zabeležena prodaja v države, ki za trenutno analizo niso zanimive. Lahko so to le vzorci izdelkov v kakšne "eksotične" države. Ker je dimenzija država enonivojska, se država *Other* ne deli več na nižje hierarhične nivoje, kot je to definirano pri kupcih. Natančnejša analiza držav, ki so združeni v ime *Other,* torej ni možna.
4. *Customer*: Dimenzija kupcev: Dimenzija je sestavljena dvonivojsko. Na prvi hierarhični stopnji se nahajajo imena podjetij, ki so obravnavana kot strateški kupci, umetno kreirana imena, kjer je združenih več različnih podjetij pod isto ime, in kupec *Other companies*, ki predstavlja vsoto prodajnih vrednosti vseh nestrateških kupcev. Ta del je zgrajen na podlagi polja *Customer* iz tabele *Sales_fact*. Na drugem, podrejenem nivoju so shranjene podružnice podjetij, nestrateška podjetja in fizične osebe, ki so izbrani na podlagi zapisov v polju *Customer_branch* iz tabele *Sales_fact*.

   Taka zgradba je primerna zato, ker le nekaj kupcev predstavlja večino prodaje, oziroma veliko malih kupcev pomeni le majhen delež v prodaji. Kocka mora biti zgrajena pregledno, da se pri analizah uporabnik lahko osredotoča na najpomembnejše podatke.
5. *Product*: Dimenzija izdelkov. Dimenzija je izdelana na podlagi polja *Product_name* iz tabele *Product*. Člane dimenzije sestavljajo komercialna imena izdelkov, torej imena, pod katerimi se ti izdelki prodajajo. V podjetjih obstajajo tudi proizvodne klasifikacije izdelkov, ki so namenjene interni obravnavi znotraj podjetja.
6. *Time*: Časovna dimenzija. Dimenzija je zgrajena na podlagi polja *Date* iz tabele *Sales_fact*. Posebnost dimenzije je, da jo sestavljajo le datumi, ko je bila izvršena vsaj ena prodaja. Časovna vrsta, če bi jo procesirali na eno izmed osi, torej ni enaka datumom koledarskega leta, kjer bi bili navedeni tudi datumi, kjer ni bilo nobene prodaje (recimo okoli novega leta), temveč je enaka datumom, ko je bila opravljena vsaj ena prodajna transakcija. To v konkretnem primeru zadošča.

Pri izdelavi kocke se velikokrat uporablja časovna dimenzija, ki je kreirana na podlagi polja v posebni časovni tabeli. Ta tabela je lahko zgrajena tako, kot kaže slika 43:

Slika 43: Možna zgradba tabele za kreiranje časovne dimenzije

ime polja	podatkovni tip
cas_kljuc	integer
datum	datetime
dan_v_tednu	varchar
teden_v_letu	int
praznik	int

Vir: Avtor

V tabelo dejstev bi lahko zapisovali poseben časovni ključ, ki bi bil povezan s časovno tabelo. To bi omogočalo lažje indeksiranje tabel in bolj prožne možnosti za kreiranje poizvedb. Na podlagi zgornje strukture bi lahko izdelali kocko, kjer bi bilo možno zastavljati vprašanja kot so:

- V katerem tednu v letu je bila realizirana največja prodajna vrednost ?
- Ali smo kdaj prodajali na praznik ?
- Pokaži prodajo za vse četrtke v letu ?

V prikazanem primeru je zadoščala kar izbira datumov iz tabele dejstev, ki pa je nato razdeljena na 4 hierarhične nivoje: *leto*, *kvartal*, *mesec*, *dan*. Komponente kocke *Prodaja* in hierarhične delitve dimenzij so vidne na sliki 45, na str. 77.

4.4.2. Izdelava osnovnih mer kocke

V kocko *Prodaja* so vključene tri osnovne mere:

1. *Actual price*: Neto prodajna cena izdelka brez prevoznih stroškov: Mera je definirana na podlagi polja *Actual_price* iz tabele dejstev *Sales_fact*. Beseda *Actual* je uporabljena zato, ker je predvidena možnost kasnejšega združevanja kock. Več kock bi se namreč dalo povezati v eno navidezno, virtualno kocko. Tako bi se lahko povezala kocka, kjer se beležijo prodajne vrednosti s kocko, kjer so zabeležene planirane vrednosti prodaje. Tak model bi omogočal analizo odnosa med dejansko prodajo *Actual price* in planirano vrednostjo prodaje *Planned price*.
2. *Actual weight*: Teža prodanega izdelka. Tako kot prodajna vrednost je določena iz tabele dejstev na podlagi polja *Actual_weight*.

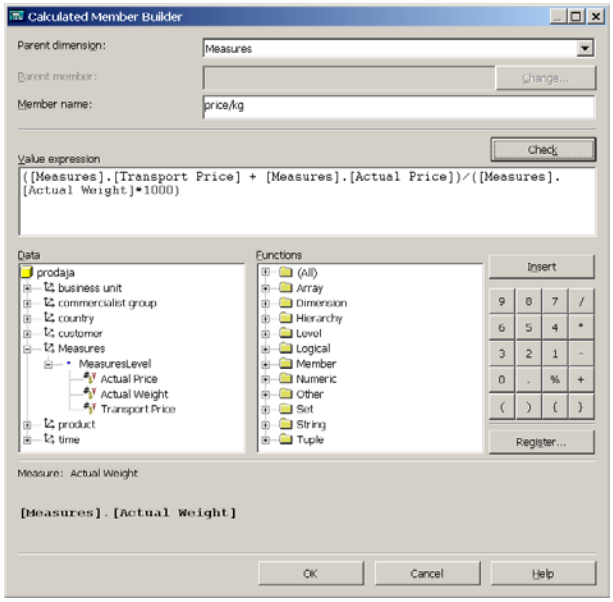
3. *Transport price*: Vrednost prevoza izdelka. Določena je na podlagi polja *Transport_price* iz tabele dejstev *Sales_fact*.

4.4.3. Izdelava izvedene mere kocke

Poleg osnovnih mer je na podlagi zahteve uporabnikov definirana tudi ena izračunana, izvedena mera, to je bruto cena izdelka na kilogram (*Price/kg*), ki se izračuna po enačbi:

*Price/kg = (Transport price + Actual price)/ (Actual weight * 1000)*

Slika 44: Definiranje izvedene mere *Price/kg* s programom Calculated Member Builder

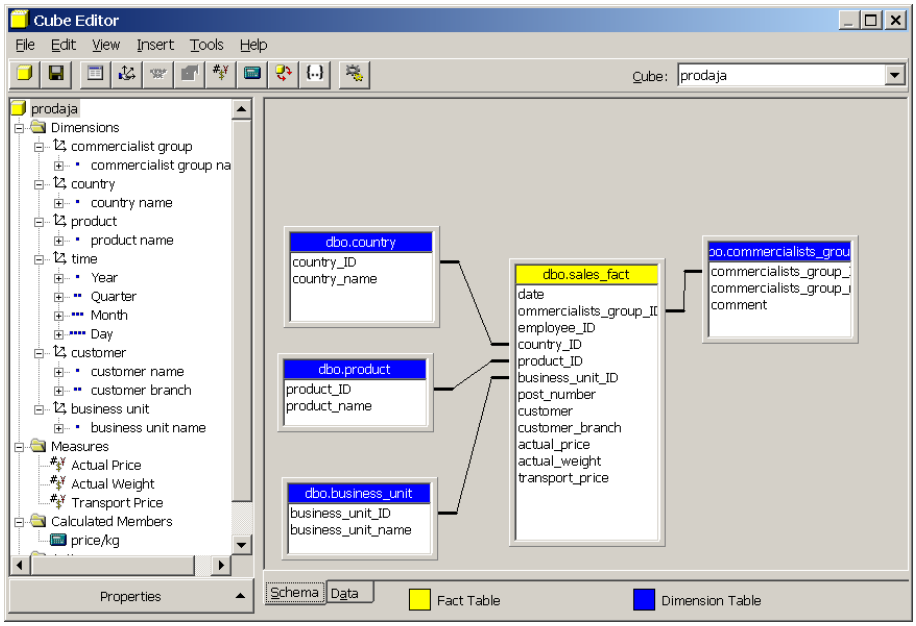


Vir: Ekranska slika, Microsoft Analysis Manager, Calculated Member Builder

4.4.4. Definiranje povezav med dimenzijami in tabelo dejstev

Povezave med dimenzijskimi tabelami in tabelo dejstev *Sales_fact* je možno ustvariti tudi v programu Cube Editor. Predhodno izdelane povezave med tabelami v relacijski zbirki podatkov - skladišču podatkov torej niso predpogoj za pravilno izdelavo OLAP kock.
Dimenzije so privatne, ker se bodo uporabljale le v tej kocki. Če bi bila kreirana še kakšna druga kocka, npr. za nabavo, proizvodnjo ali kadrovsko službo, bi lahko dimenzije, ki so skupne vsem področjem, kot so npr. delavci ali poslovne enote, po kreiranju dali na voljo (*Shared Dimensions*), da bi jih lahko uporabljali v vseh kockah v sklopu določene zbirke večdimenzijskih struktur.

Slika 45: Grafičen prikaz zgradbe kocke *Prodaja* v programu Cube Editor



Vir: Ekranska slika, Microsoft Analysis Manager, Cube Editor

4.4.5. Izdelava agregatov

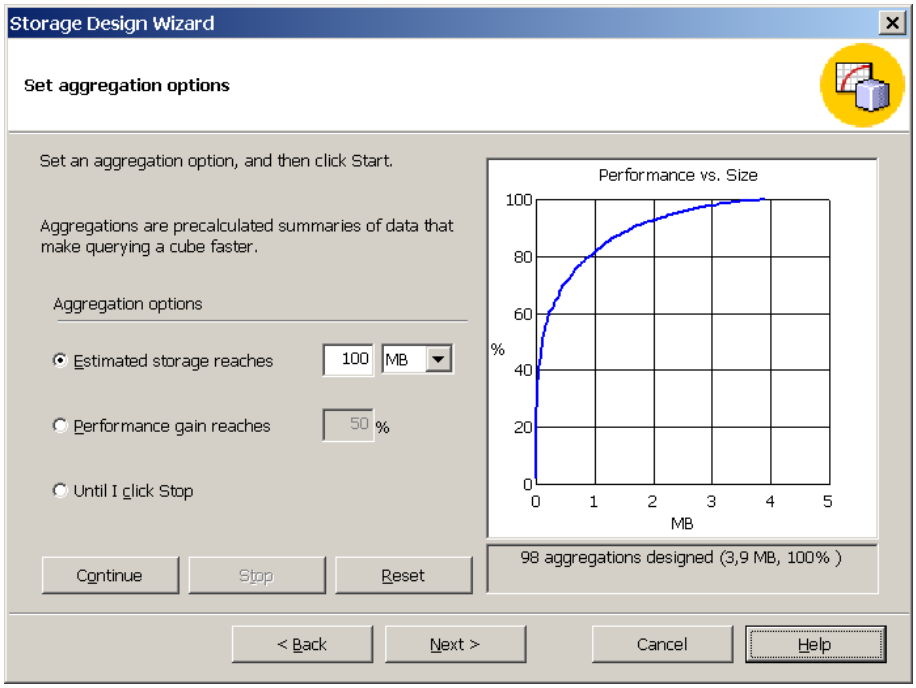
Ko sem zgradil strukturo, sem shranil kocko v MOLAP načinu, ki bo v večdimenzijski strukturi hranila tako izvorne kot agregirane podatke.

Agregati se shranjujejo skupaj z izvornimi podatki v kocki. Če kocko shranimo lokalno, zasedejo izvorni podatki skupaj z agregati in informacijami o strukturni zgradbi le 539 KB prostora na trdem disku. Velikost relacijske zbirke, skladišča podatkov *Prodaja,* pa brez tabel z agregiranimi vrednostmi znaša preko 5 MB. Razlika je res velika. Napredni algoritmi za kreiranje agregatov, stiskanje podatkov in shranjevanje le celic s podatki v večdimenzijski strukturi so zelo učinkoviti. Bojazen glede eksplozije podatkov pri uporabi veliko in velikih dimenzij je zato odveč.
Ker se tako shranjeno kocko s podatki o celoletni prodaji lahko shrani v datoteko velikosti 539 KB, je podatke (celo na eni disketi) možno enostavno prenesti na uporabniški računalnik, kjer uporabniki lahko izvajajo analize na podlagi podatkov iz OLAP kocke *Prodaja,* brez povezave na Analysis Services strežnik.

4.4.5.1. Izdelava agregatov s programom Storage Design Wizard

S Storage Design Wizardom sem določil optimalno število agregatov, ki so potrebni za čim boljši odziv sistema. Na sliki 46 je vidno, da mora sistem zgraditi 98 agregatov, če želimo, da bo pospešitev sistema 100%. Za tako opravilo obstaja ocena o porabljenih 3,9 MB prostora na trdem disku, kar je mnogo manj, kot je bilo določeno kot zgornja meja (100 MB).

Slika 46: Kreiranje optimalnega števila agregatov s programom Storage Design Wizard



Vir: Ekranska slika, Microsoft Analysis Manager, Storage Design Wizard

4.4.5.2. Izračun povprečnega časa poizvedb

Optimiranje delovanja sistema je vidno iz enačbe:

*pospesitev = 100 * (maxcas - zeljencas) / (maxcas - mincas)*

pospesitev = željen odstotek pospešitve sistema
maxcas = maksimalni poizvedovalni čas
mincas = minimalni poizvedovalni čas
zeljencas = željen čas odziva sistema

Pri pogoju, da je *zeljencas* = *mincas,*

je enačba enaka:

*pospesitev = 100 * (maxcas - mincas) / (maxcas - mincas)*

kar pomeni, da bomo pri zahtevi po najboljšem možnem odzivnem času sistema (*zeljencas=mincas*) morali zgraditi 98 agregatov, da bo program lahko zagotavljal 100% pospešitev sistema glede na sistem, kjer ne bi bilo vnaprej izdelanih nobenih sumarnih agregatov. Te dodatne strukture naj bi po oceni programa zavzele 3,9 MB prostora na trdem disku.
Minimalni poizvedovalni čas je v veliki meri odvisen od zmogljivosti strojne opreme, ki se uporablja pri izdelavi rešitve. V konkretnem primeru se je večina poizvedb izvedla v delčku sekunde.

4.4.5.3. Procesiranje OLAP kocke

Analysis Services poznajo različne načine procesiranja:

- *Inkrementalni način*: Pomeni način, pri katerem se dodajajo novi podatki k obstoječi kocki ali delu kocke - particiji in se ažurirajo agregati. Ker dodatni zapisi ne spreminjajo obstoječih podatkov in struktur, je tak način gradnje precej hitrejši, kot če bi kocko vedno gradili od začetka. Tak način ne vpliva na izvajanje trenutnih analiz s strani uporabnikov. Med procesiranjem kocke lahko uporabniki nemoteno izvajajo poizvedbe. Ko je procesiranje končano, se samodejno vidijo spremembe, brez potrebe po ponovni povezavi na strežnik. Tako procesiranje ni primerno, če se je spremenila struktura kocke, oziroma če so se spremenili izvorni podatki, iz katerih je bila kocka zgrajena.
- *Osvežitev kocke*: Pri tem načinu se podatki v kocki zbrišejo in osvežijo z novimi. Prav tako se ponovno preračunajo agregati. Izbor te opcije je primeren za primer, ko se podatki v skladišču podatkov spremenijo, ne spremeni pa se struktura kocke. Podobno kot pri inkrementalnem načinu uporabniki tudi tu lahko nemoteno izvajajo analize med samim izvajanjem procesiranja kocke. Po osvežitvi kocke se podatki v poizvedbah samodejno spremenijo brez potrebe po ponovni vzpostavitvi povezave z Analysis Services strežnikom.
- *Polno procesiranje*: Pri tem načinu strežnik zgradi kocko od začetka. Tak način je najpočasnejši, uporablja se v primeru, če se je spremenila struktura kocke, npr. če je bila dodana nova dimenzija, hierarhični nivo dimenzije ali mera. Po takem procesiranju se mora uporabnik ponovno prijaviti na strežnik, da lahko izvaja analize na podlagi nove kocke.

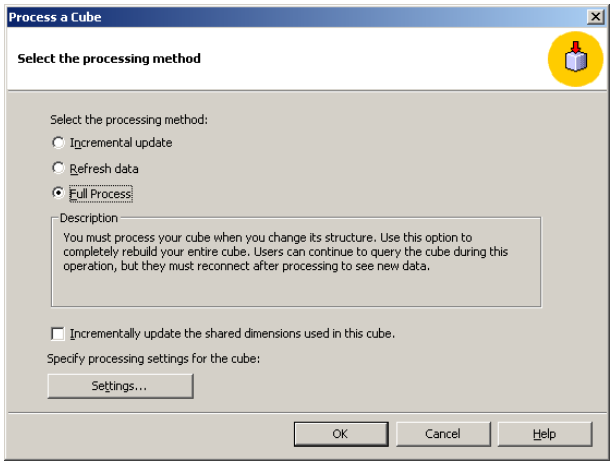
Možno je nastaviti ločeno procesiranje posameznih privatnih ali javnih dimenzij, ki so vključene v kocko in ostale stvari, kot so:

- Kdaj naj bodo podatki na voljo uporabnikom za analizo: ali takoj, ko so bili vneseni novi podatki, ali šele takrat, ko so preračunani tudi vsi agregati. V prvem primeru

se bodo novi podatki sicer hitreje prikazali, odziv sistema pri kreiranju poizvedb pa bo precej počasnejši.

- Po koliko najdenih napakah naj se procesiranje ustavi.
- Kje naj se shranjuje dnevnik s podatki o poteku procesiranja kocke.

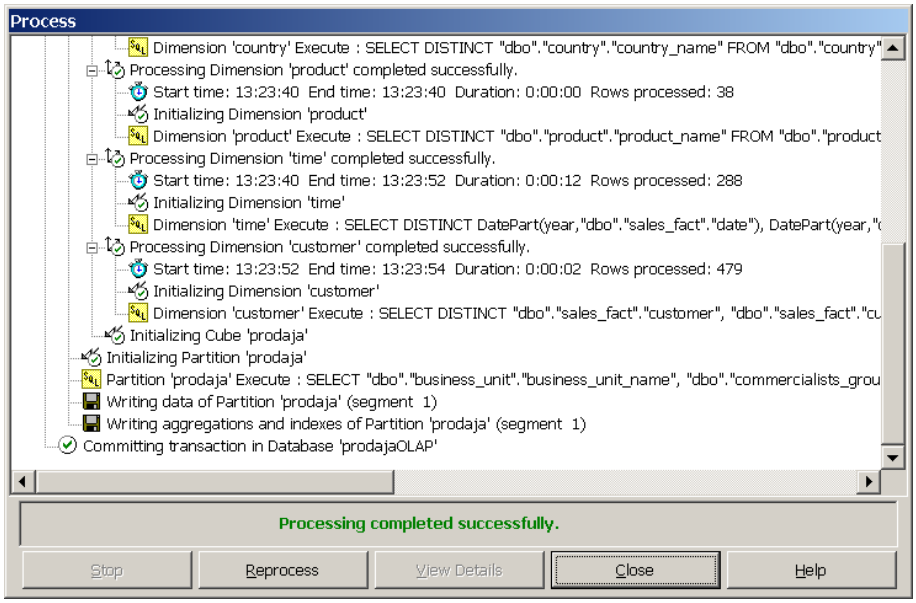
Slika 47: Izbira načina procesiranja kocke



Vir: Ekranska slika, Microsoft Analysis Manager

Celotno procesiranje OLAP kocke *Prodaja*, ki pomeni prepis izvornih podatkov v večdimenzijsko obliko in izdelava 98 agregatov, traja v konkretnem primeru manj kot minuto, zato sem uporabil kar opcijo *Polno procesiranje*.

Slika 48: Procesiranje kocke

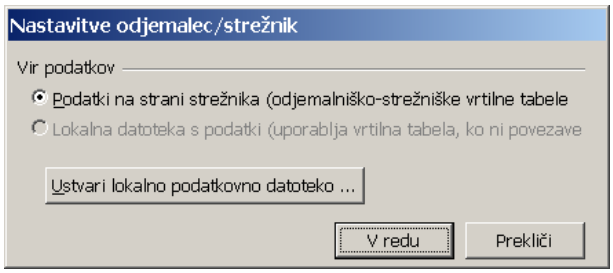


Vir: Ekranska slika, Microsoft Analysis Manager

4.4.6. Kreiranje lokalne kocke

Lokalno kocko se lahko ustvari preko Microsoft Excela 2000. Po tem, ko je bila v Excelu narejena povezava z Analysis strežnikom in ustvarjena večdimenzijska vrtilna tabela za pregled podatkov iz kocke, se lahko iz orodne vrstice *Vrtilna tabela* izbere opcija *Nastavitve odjemalec/strežnik*,

Slika 49: Ustvarjanje lokalne kocke iz programa Microsoft Excel 2000:

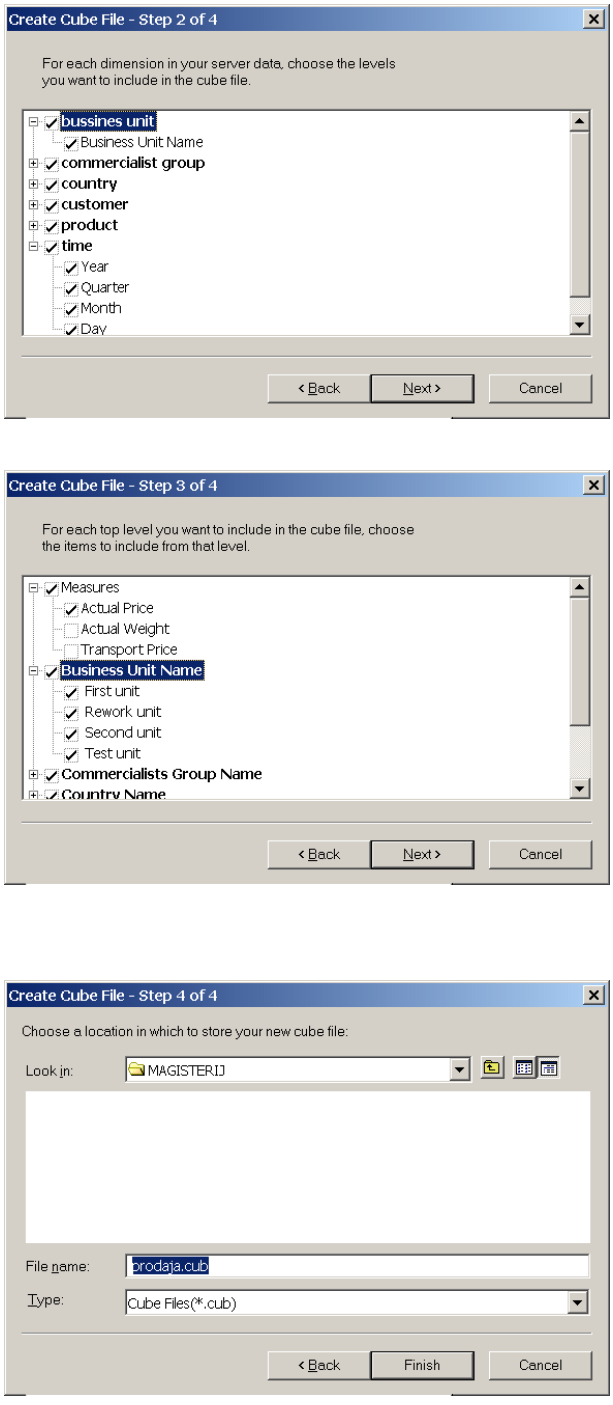


Vir: Ekranska slika, Microsoft Excel 2000

kjer se s pomočjo opcije *Ustvari lokalno podatkovno datoteko* sproži postopek za kreiranje lokalne kocke. Lokalne kocke so zelo primerne za poizvedovanje po podatkih, ki se spreminjajo na daljša obdobja. Za izdelavo mesečnih poročil se lahko izdela lokalno kocko le enkrat na mesec. Prednost takega načina je, da uporabniki pregledujejo podatke iz kocke brez potrebe po mrežni povezavi z Analysis strežnikom.
Ustvarjanje lokalne kocke poteka v štirih korakih. Tudi tu je v pomoč "čarovnik" z imenom Create Cube Wizard:

1. korak: Prikaz navodila za pravilno izdelavo lokalne kocke
2. korak: Izbira dimenzij in nivojev posamezne dimenzije, ki bodo vključene v lokalno kocko.
3. korak: Izbira članov posamezne dimenzije in izbira mer, ki bodo vključene v lokalno kocko.
4. korak: Določitev imena kocke (*.cub) in področja, kamor bo shranjena.

Slika 50: Grafični prikaz korakov pri kreiranju lokalne kocke



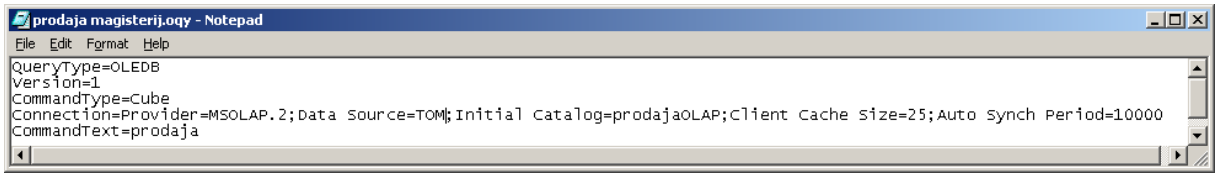
Vir: Ekranska slika, Microsft Excel 2000, Create Cube File

4.4.7. Pregledovanje kocke preko Microsoft Excela

Iz Microsoft Excela 2000 se z OLAP virom podatkov da povezati enako kot s katerim koli drugim zunanjim virom podatkov. Excel zna delati tudi z OLAP izdelki

neodvisnih proizvajalcev, če le ti priskrbijo gonilnik za vir podatkov, ki je združljiv z OLE-DB za OLAP.
Excel lahko podatke, ki jih pridobi iz OLAP vira podatkov, prikaže le v poročilu vrtilne tabele ali grafikona. Nastavitve za OLAP poročila vrtilne tabele in grafikona se lahko shranijo v predloge in ustvarijo poizvedbene datoteke (*.oqy). Ko se (*.oqy) datoteka odpre, se v Excelu prikaže prazno poročilo vrtilne tabele pripravljeno za obdelavo s strani uporabnika.

Slika 51: Vsebina (*.oqy) poizvedbene datoteke



Vir: Ekranska slika, datoteka "magisterij.oqy"

Napreden način pregledovanja OLAP podatkovnih zbirk omogoča Excelu, da lahko analizira zelo veliko količino izvornih podatkov.

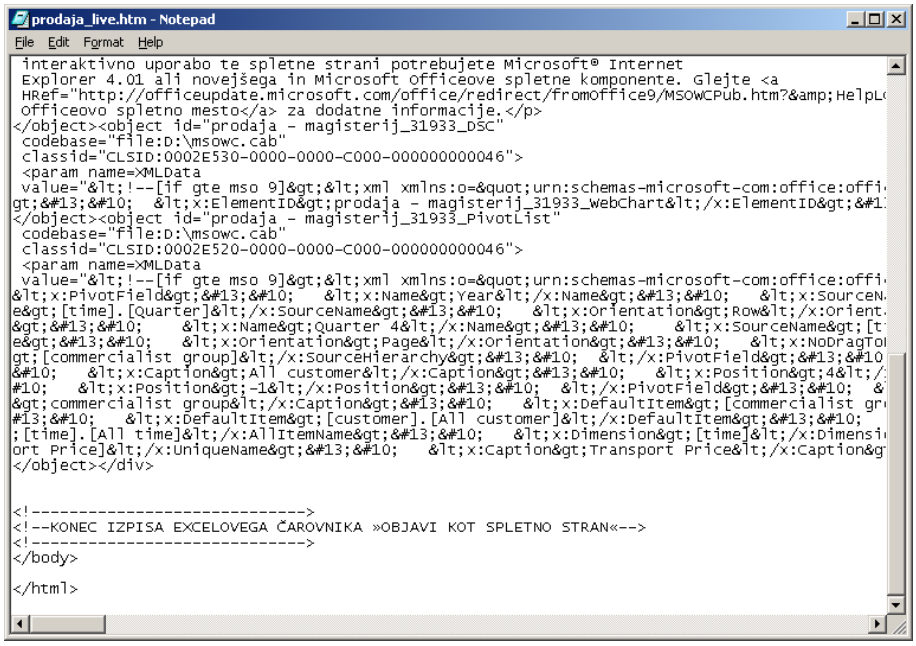
OLAP podatkovne zbirke so organizirane tako, da olajšajo pridobivanje in analiziranje velike količine podatkov. Preden Microsoft Excel prikaže povzetke v poročilu vrtilne tabele ali grafikona, izvede OLAP strežnik večino izračunov, da povzame podatke. Tak način manj obremenje računalniško omrežje in uporabniške računalnike.

OLAP strežnik pošlje Excelu nove podatke vedno, ko uporabnik spremeni pogled, postavitev poročila vrtilne tabele ali grafikona.

4.4.8. Kreiranje interaktivne spletne aplikacije za pregledovanje kocke

Microsoft Excel 2000 omogoča samodejno izdelavo interaktivne spletne strani na podlagi definirane večdimenzijske vrtilne tabele. Če bi tako aplikacijo shranili na ustreznem intranet ali internet strežniku, bi uporabnikom omogočili analiziranje podatkov iz OLAP kock kar preko brskalnika Microsoft Internet Explorer.
Tak način omogoča prikaz ne statične, temveč interaktivne spletne strani, katere izgled grafikonov in preglednic definira vsak uporabnik posebej. Določene funkcionalnosti pregledov, ki jih sicer omogoča Microsoft Excel 2000, se s tem zgubijo, je pa tak spletni pregled za marsikoga povsem zadovoljiv. Programsko spreminjati parametre aplikacije, ki je bila izdelana samodejno s pomočjo "čarovnika", je nemogoče, kot je razvidno iz pregleda kode izdelane datoteke *Prodaja_live.htm.*

Slika 52: Del programske kode samodejno generirane interaktivne OLAP spletne strani

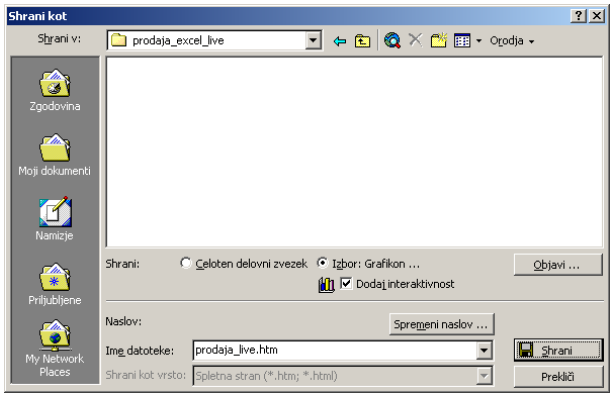


Vir: Ekranska slika, Microsoft Notepad

Tako spletno stran je možno izdelati v dveh korakih ob predpostavki, da je v Excelu že izdelana večdimenzijska vrtilna tabela, ki prikazuje podatke z OLAP kocke (iz Analysis Services strežnika ali iz lokalne datoteke):

1. V Excelu je potrebno izbrati opcijo *Datoteka* in nato *Shrani kot spletno stran*
2. Izbere se *Izbor: Grafikon...,* odkljuka možnost *Dodaj interaktivnost* in določi ime, v konkretnem primeru *Prodaja_live .htm* (slika 53)

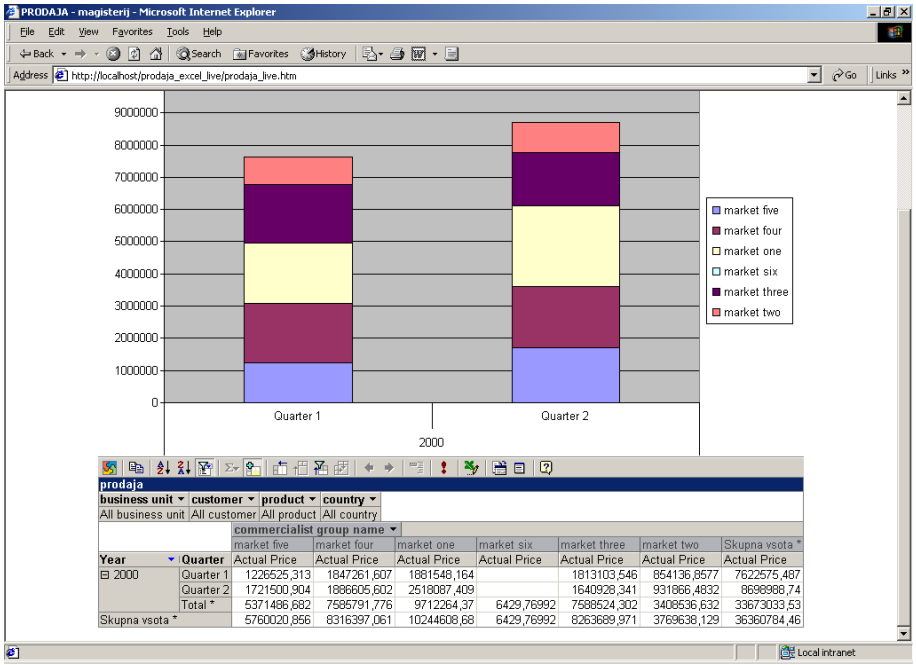
Slika 53: Izdelava interaktivne spletne večdimenzijske vrtilne tabele



Vir: Ekranska slika, Microsoft Excel 2000

Tako izdelano aplikacijo je že možno pregledovati s programom Microsoft Internet Explorer. Če bi to naredili na računalniku, kjer je nameščen tudi Internet Information Services, bi tudi drugi uporabniki lahko dostopali do OLAP kocke kar preko spletnega brskalnika (slika 54).

Slika 54: Primer interaktivne spletne analize na podlagi OLAP kocke



Vir: Ekranska slika, Microsoft Internet Explorer

4.5. English Query aplikacija

Izdelavo English Query aplikacije sem izvedel po korakih, ki so navedeni v poglavju 3.3.2., na str.46:

4.5.1. Pomembna vprašanja, na katera mora aplikacija znati odgovoriti

Zbrana in pregledana so bila poročila, ki jih uprava in posamezne službe že uporabljajo. Večina pregledov je bila narejena s pomočjo centralnih računalniških programov, preko katerih se tudi izvajajajo vse potrebna opravila za obvladovanje poslovnega procesa. To so v večini primerov dvodimenzionalni sumarni pregledi transakcijskih procesov, npr.:

- Seštevek prodaje določenega izdelka v nekem obdobju.
- Skupni znesek prodaje v posamezni poslovni enoti.
- Pregled prodaje določenega izdelka v odvisnosti od komercialista.

Po razgovoru z vodstvom podjetja je bilo ugotovljeno, da ni toliko problem dobiti rezultat za določen kriterij, največji problem so ravno stvari, ki jih določa FASMI test:

1. Hitrost: Vsak sumarni pregled je ponavadi pomenil nekajminutno čakanje na rezultat. Program za izdelavo pregledov je bil namreč napisan tako, da je pri kakršnikoli poizvedbi vedno prečesaval izvorne podatke. Vnaprej ni bilo izdelanih nobenih sumarnih tabel, ki bi lahko precej pohitrili pridobitev rezultata. Tako čakanje na vsak rezultat je bilo za vodstvene delavce popolnoma neprimerno. Vse take analize so ponavadi pripravljali delavci v posebni službi za analize lahko tudi cel teden. Ko je tak povzetek končno prišel do vodstva podjetja, se je lahko nek dogodek, pred katerim bi morali biti oboroženi z raznimi kvalitetnimi informacijami, že zgodil, izdelana so bila poročila brez prave vrednosti v smislu: "Po toči zvoniti je prepozno !"
2. Analiza: Uporabniki so morali biti za izdelavo pregledov dobro usposobljeni, saj so morali znati nastaviti različne filtre, poznati pravilno zaporedje filtrov, razvozlati razne šifre gibanj izdelkov. Podatki so bili na koncu prikazani v razpredelnici samo v numerični obliki in razčlenjeni po raznih postavkah, zato je bilo potrebno dobljene rezultate večkrat prenesti v druge programe in jih tam dodatno preračunati, izdelati grafe in podobno. Vsi so pogrešali različne možnosti za enostavno izdelavo analiz, kot so možnost "povleci - spusti" za pripravo nastavitev, grafični prikazi podatkov, različni tipi sortiranja, vpis opisnega vprašanja, na podlagi katerega bo program izdelal analizo, vplivanje na določene postavke v rezultatu v smislu izdelave "kaj-če" analiz in podobno.
3. Večdimenzionalnost: Dosedanji programi so imeli veliko omejitev glede količine podatkov, ki so jo bili sposobni obdelati in predvsem število dimenzij - kriterijev, ki so bili vključeni v preglede.

Začetni nabor vprašanj, na katere mora aplikacija sposobna pravilno odgovoriti, je bil opredeljen tudi na podlagi nastavitev pregledov OLAP kocke, ki so jih uporabniki ustvarjali v Microsoft Excelu 2000 v prvi fazi izdelave rešitve.

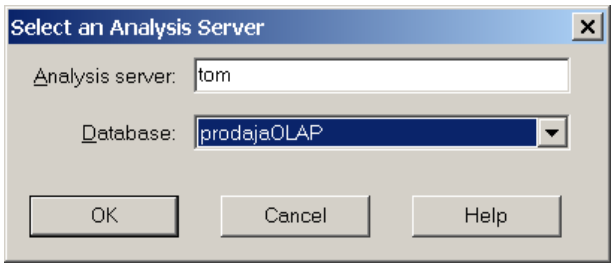
4.5.2. Kreiranje osnovnega modela

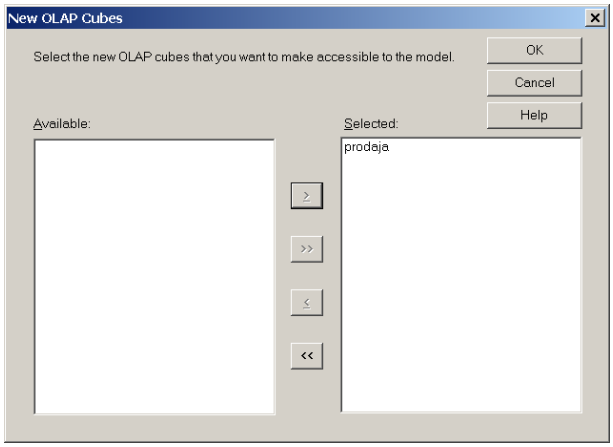
S pomočjo "čarovnika" za izdelavo EQ projekta so bile v programu Microsoft English Query določene začetne nastavitve (slika 55):

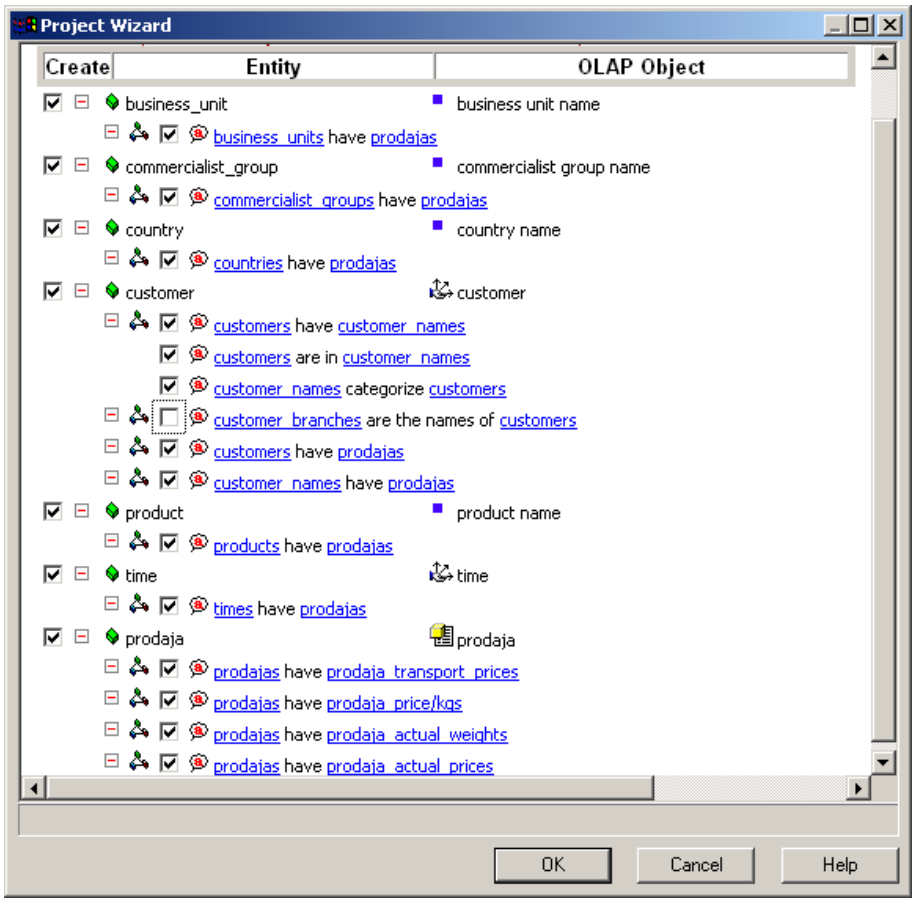
1. Definiranje povezave z Analysis Serverjem in izbor večdimenzijske baze podatkov, kjer se nahaja OLAP kocka, na podlagi katere se bo gradil konkreten EQ model.
2. Izbira OLAP kocke za uporabo v EQ modelu: V konkretnem primeru je bila zgrajena le ena kocka. EQ model je možno graditi tudi na podlagi več OLAP kock.
3. Izbira entitet: Izbrane so bile vse entitete, ki jih je ponudil "čarovnik", to so: *business unit*, *commercialist group*, *country*, *customer*, *product* in *time*. Izbrane so bile tudi vse mere: *actual price*, *transport price*, *actual weight* in izvedena

dimenzija *price/kg*. Na podlagi izbranih entitet je "čarovnik" sam zgradil osnovne povezave med entitetami, ki jih je sestavil na podlagi definiranja hierarhične strukture OLAP kocke *Prodaja*. Samodejno je zgradil povezave, kot je npr.: *business units have prodajas*, kar pomeni, da poslovne enote vsebujejo različne mere, to je različne vrednostne in količinske prodajne vrednosti. Samodejno zgrajene povezave so ostale nespremenjene, v tej fazi je bila opuščena le povezava *customer brancehes are the names of customers*, ki določa, da člani hierarhičnega nivoja *customer branch* dimenzije *customer* predstavljajo imena kupcev, kar je napačno.

Slika 55: Nastavitev osnovnih parametrov EQ aplikacije:







Vir: Ekranska slika, Microsoft English Query

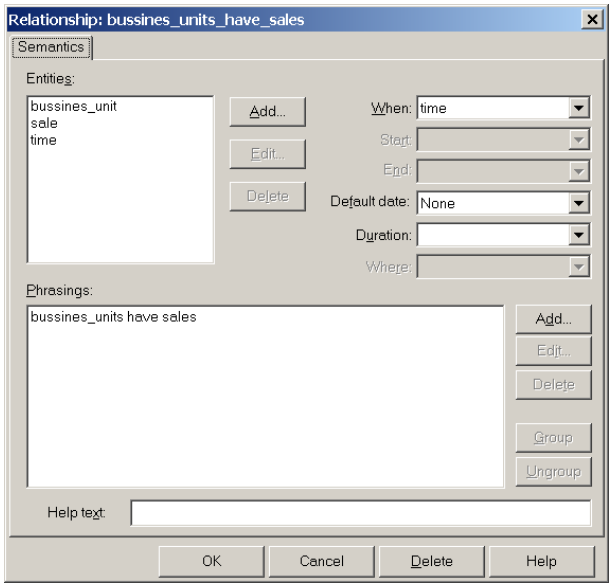
4.5.3. Nadgradnja osnovnega modela z entitetami, povezavami in vnosi v slovar

Ker so nekatere stvari v osnovnem modelu napačne ali nepopolne, je bilo potrebno nadgrajevati osnovni model. Model je bil dopolnjen z naslednjimi relacijami:

- Dodana je bila entiteta *customer branch* in določena kot vrednost nivoja *customer branch* dimenzije *customer*.
- Spremenjena je bila entiteta *customer*, da je definirana s hierarhičnim nivojem *customer name* dimenzije *customer*.
- Dodana je bila relacija *customers have customer branches*, ki pravilno določa odnos med kupci in podružnicami.
- Spremenjeno je bilo ime entitete *prodaja*, ki jo je "čarovnik" vzel iz imena OLAP kocke v *sale*, v smislu gradnje celotnega modela v angleškem jeziku.
- Dodana je bila relacija *customers branches have sales*. Na podlagi te relacije model zna odgovoriti na vsa vprašanja, ki se tičejo količinske in vrednostne prodaje za posamezno kupčevo podružnico.
- Natančneje je bil opredeljen odnos določenih entitet in entitete *time*. V relacije *customers have sales*, *customers branches have sales*, *commercialist groups have*

sales, *bussiness units have sales*, *sales have sales actual price*, *sales have sales actual weight*, *sales have sales transport price* in *sales have sales price/kg* je bila dodana entiteta *time* kot opredelitev *When*. S tem delujejo odgovori o prodaji, ki zajemajo tudi časovno komponento, npr. *Show actual price for customer Mezic for january 2000* (slika 56).

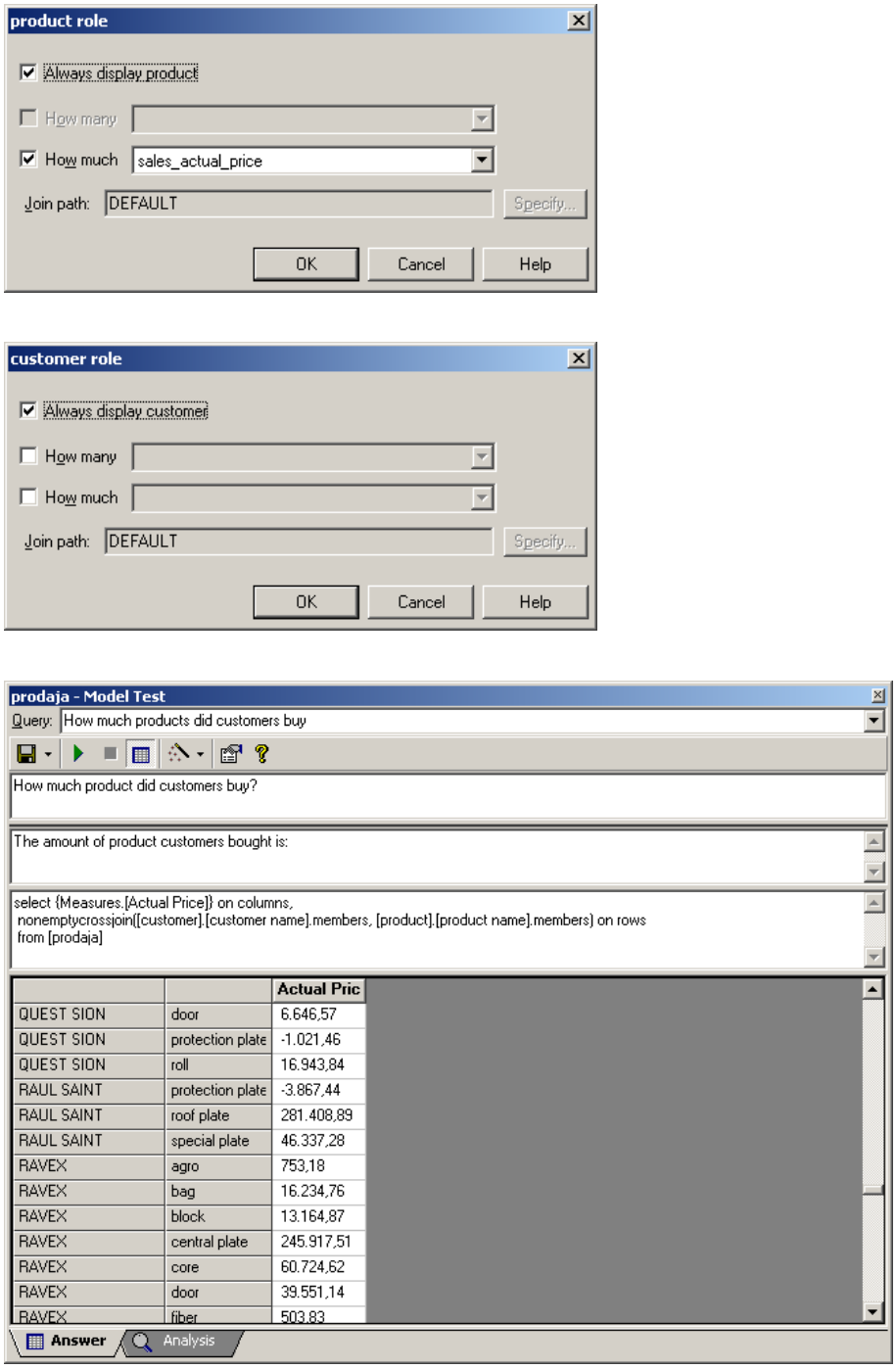
Slika 56: Entiteta *time* kot časovna opredelitev relacije *business units have sales*



Vir: Ekranska slika, Microsoft English Query

- Dodana je bila *Verb Phrasing* relacija v načinu *Subject Verb Object Object* z opredelitvijo *comercialist groups sell customers products*. Pri tem je bil še dodatno opredeljen direktni objekt *product* v primeru vprašanja *How much products did we sell to Mezic* z vrednostjo *sales actual price* kot vrednostni (amount) odgovor. V primeru vprašanja *How many products ...* pa bo model uporabil ukaz *COUNT* v MDX sintaksi, kar pomeni, kolikokrat je bil prodan izdelek določenemu kupcu.
- Ker je bil že v predhodnih relacijah definiran odnos med kupci in podružnicami, bo EQ model znal odgovoriti tudi na vprašanje v stilu *How much products did we sell to Gorec.*
- Podobno je bila dodana tudi relacija *business units sell customers products*.
- Nakup je bil opredeljen z relacijo *customers buy products.*
- Pri relaciji *customers buy products* je natančneje definirano obnašanje sistema ob določenih vprašanjih. Če uporabnik npr. vpraša *How much products did customers buy* se želi, da program ne bo odgovoril zgolj s številkami, ampak tudi z imeni kupcev in vrstami izdelkov za posameznega kupca. V ta je bila pri tej relaciji za entiteto customer izbrana opcija *Always display customer*, entiteto product pa vrednostno *How much* definirana z mero *sales actual price* in prav tako določena z *Always display product*, da se v odgovoru vedno izpiše tudi izdelek (slika 57).

Slika 57: Natančnejša opredelitev relacije *customers buy products* in odgovor

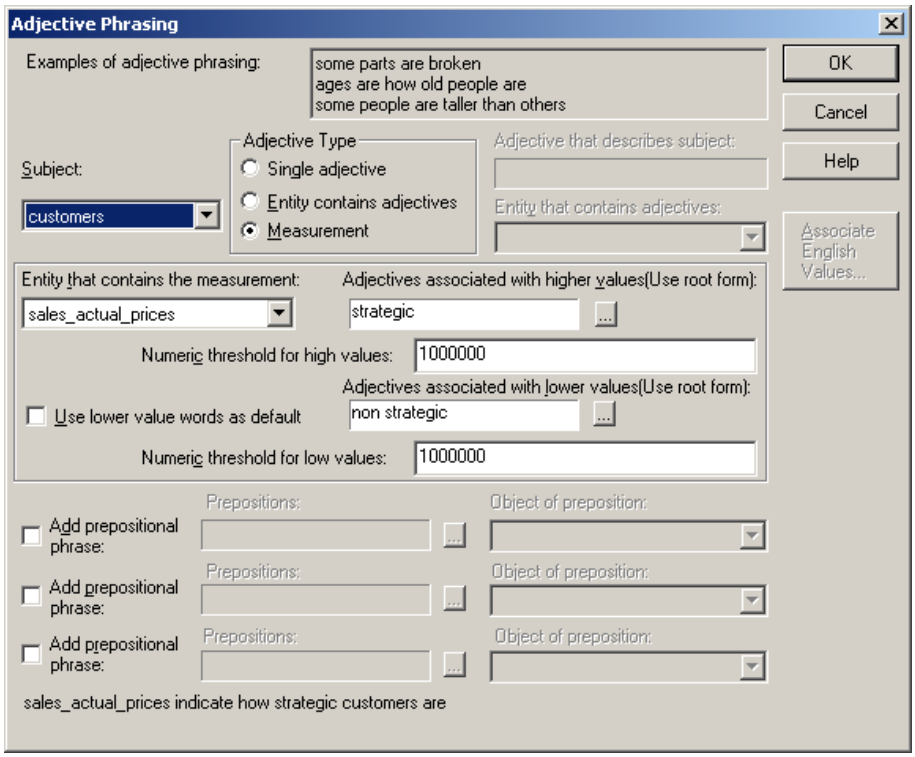


Vir: Ekranska slika, Microsoft English Query

- Za pravilen odgovor na vprašanje o lokaciji posameznega kupca, torej v kateri državi se nahaja posamezen kupec, je bila definirana entiteta *customers are in countries*.
- Dodana je bila *Adjective Phrasing* relacija *sales actual price indicates how strategic customers are*. Potrebno je bilo določiti mero in tisto vrednost, ki določa pojem "strateški". V tem primeru je mera *actual price* in pogoj, da je nek kupec

strateški: *actual price > 1000000*. Tako bo delovalo vprašanje v smislu *Show strategic customers*. Prav tako je bilo potrebno določiti pridevnik, ki ne zadošča temu pogoju. Določen je bil pridevnik *non strategic*.

Slika 58: Definiranje relacije, ki definira strateške kupce



Vir: Ekranska slika, Microsoft English Query

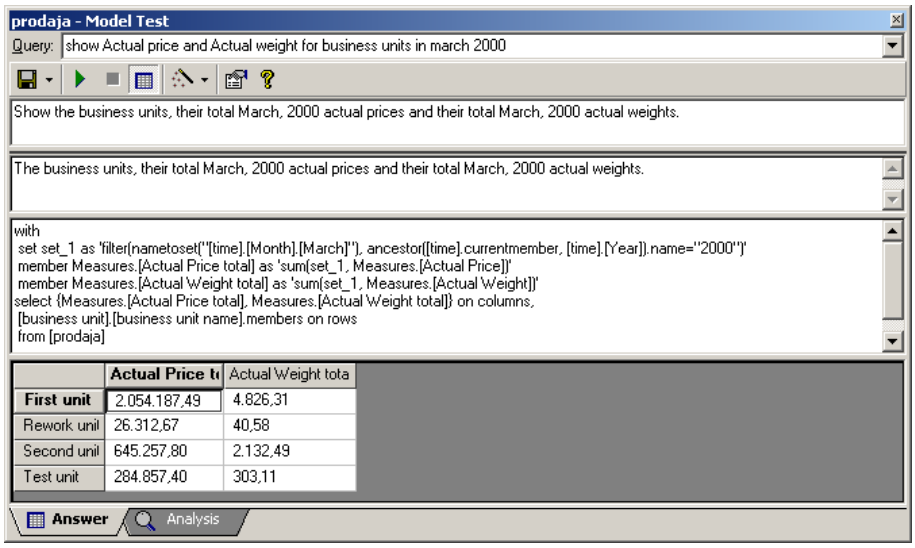
- Dodani so bili sinonimi za določene entitete. Tako so bili entiteti *country* dodani tudi sinonimi *fatherland* in *countryside*, entiteti customer *buyer*, entiteti product *artefact*, entiteti time *epoch* in *age.*
- V entiteti *customers buy products* so bili določeni sinonimi za glagol *buy*, to je *get* in *purchase*.
- Določeni so bili tipi entitet. Za vprašanja, ki se začnejo z *Who* so veljavne entitete *business unit, commercialist group, customer, customer branch*, na vprašanje *Where* bo odgovorila entiteta *country*, pri *When* je pravilna entiteta *Time* in mere (*Measures*) za entiteto *Sale*.

4.5.4. Preverjanje ustreznosti modela

Microsoft English Query urejevalnik vsebuje tudi modul Model Test, ki že razhrošči in zgradi model do take mere, da se lahko preizkusi na konkretni bazi podatkov s pravimi vprašanji. V modelu lahko razvijalec vidi vse pomembne informacije (slika 59):

- Preoblikovanje vnosnega vprašanja v obliko, ki jo sistem razume v smislu ustrezne pretvorbe v MDX sintakso.
- Opisni del odgovora na zastavljeno vprašanje v smislu *Prodaja poslovnih enot v letu 2000 je znašala:*
- Pravilnost samodejno zgrajene MDX računalniške sintakse, na podlagi katerega računalnik zna poiskati odgovor v OLAP kockah.
- Konkretni rezultat v tabelarični obliki.
- V primeru nejasnosti vprašanja je vključen tudi pomočnik Suggestion Wizard, ki predlaga gradnjo ustreznih relacij, ki bi bile potrebne za pridobitev pravilnega odgovora.
- Testirana vprašanja je možno shraniti v regresijsko datoteko v XML obliki za kasnejšo analizo, v konkretnem primeru je to datoteka *prodaja.eqr*.

Slika 59: Testiranje ustreznosti modela :



Vir: Ekranska slika, Microsoft English Query

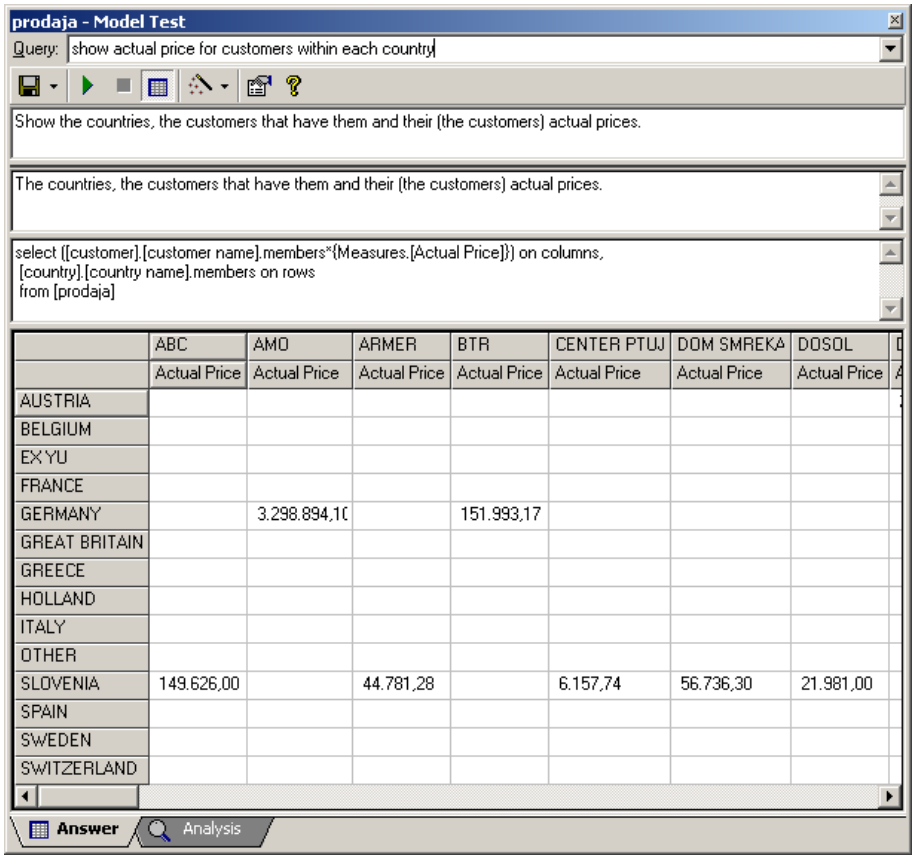
4.5.5. Spisek tipov vprašanj, ki dajo ustrezen odgovor

- Izpis seznama članov dimenzij, kot so vprašanja*: List the countries*, *Show countries*, *countries*, *Who do we sell to*. Pri vprašanju *What are the countries*, program preuredi vprašanje v *What is the definition of country* in odgovori s strukturnimi lastnostmi entitete. Podobno velja za entitete *customer, business unit, commercialist group, sale, product* in *customer branch.*
- *Show the customers and their customer branches* oziroma krajše *Show customers and their branches.*
- Izpis posameznih članov dimenzij v smislu: *Show branches of Novit* ali *What are the customer branches of Novit.* V primeru vprašanja *Show Novit*, bo sistem

preuredil vprašanje v *What do you know about Novit* in bo zaenkrat odgovoril le z imenom podjetja. Če bi bila entiteta customer definirana z več atributi, bi bilo možno zvedeti več značilnosti podjetja, npr. naslov, telefon, kontaktna oseba. Primer vprašanja, ki bi ga v tem primeru lahko zastavil uporabnik: *List customers that have company phones = "00386 4 5134-556"* ali *List the customers in the "4220" postal area.*

- Delujejo tudi vprašanja, ki se začnejo z bededo *Find*, npr.: *Find the customers who ordered block.*
- Vprašanja o vrednostnih in količinskih podatkih - mer: *Show sales actual price for Austria, Show actual weight for business units,* lahko zajemajo tudi več mer, npr.: *Show actual price and actual weight and transport price for countries.* Deluje prikaz količinske prodaje posamezne komercialne skupine, npr.: *Show actual weight for Market six.*
- Vprašanja, ki se tičejo določenega časovnega obdobja, npr.: *Show sales actual price for Austria in 2000, Show actual price and actual weight and transport price for countries in 10 january 2000, Show transport price for business units in march 2000, Show sales between May 1, 2000 and January 15, 2001*
- Ker je bila definirana relacija, s pomočjo katere je možno ugotoviti, kateri kupci so strateški (slika 58, na str. 52), je možno vprašati naslednja vprašanja: *Show strategic customers.* Vprašanja delujejo tudi v odvisnosti od časa: *Show strategic customers in 2000.*
- Lahko se uporabijo tudi bolj zapletena vprašanja, npr. vprašanje o količinah in vrednostih kupcev, prikazano analitično po njihovih podružnicah: *Show actual price and actual weight and transport price for customers by branch.*
- V primeru, da uporabnik želi dobiti informacijo o številu kupcev, katerim je bila prodana določena vrsta izdelka, ločeno po državah, je možno vprašanje: *How many customers in each country have ordered bag*. Program tu uporabi funkcijo *COUNT* v MDX sintaksi. Podobno se model obnaša, če se želi izvedeti imena kupcev, katerim je podjetje prodalo več kot 3 izdelke: *Show customers that have more than 3 products.*
- Možno je spraševati tudi v smislu negiranih vprašanj, recimo: *Which customers have not ordered agro.*
- Vprašanja, ki imajo za posledico zelo obsežne analitične odgovore, npr.: *Show actual price for customer within each country,* ki ga model priredi v *Show the countries, the customers that have them and their (the customers) actual prices* (slika 60).

Slika 60: Primer prikaza prodajnih vrednosti kupcev, ločeno po državah



Vir: Ekranska slika, Microsoft English Query

- Na vprašanje *Show Time* bo program odgovoril s trenutnim sistemskim časom na računalniku.

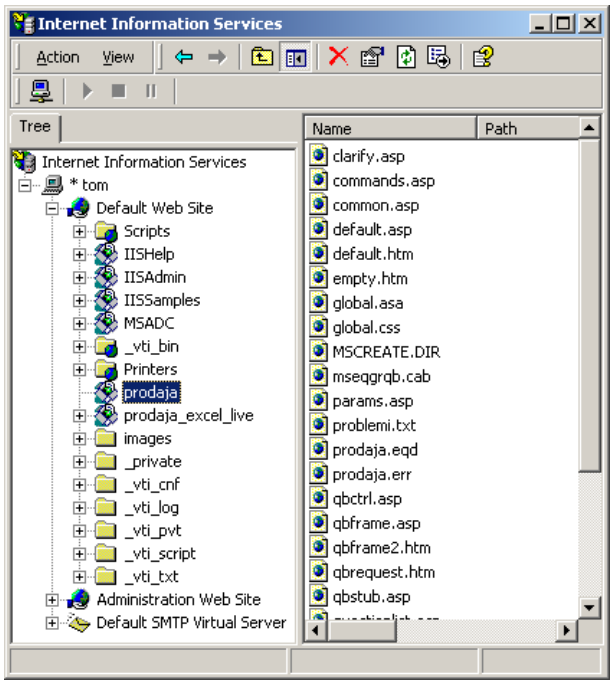
4.6. Izdelava internetne aplikacije

Za osnovo pri izdelavi spletne aplikacije sem uporabil primer ASP aplikacije, ki se namesti na lokalni disk po namestitvi programa Microsoft English Query. Nahaja se na področju *\Program Files\Microsoft English Query\Samples\Applications\ASP*. ASP tehnologija namreč zagotavlja učinkovit način kreiranja interaktivnih spletnih aplikacij (Esposito, 13.2.2002).

1. Na računalniku z Windows 2000 Server operacijskim sistemom je bilo najprej potrebno namestiti Internet Information Services 5.0, da je strežnik omogočal izvajanje spletnih storitev.
2. V nastavitveni datoteki *setup.cmd* je bilo potrebno nastaviti vse potrebne parametre za izdelavo mape in ustreznih datotek za spletno English Query aplikacijo:

- Ime English Query domain datoteke: *Prodaja.eqd*
- Ime enote trdega tiska, kamor so se fizično namestile datoteke za spletno aplikacijo: *C:*
- Mapa, kjer so se nahajale predhodno zgrajene EQ datoteke: *D:\Documents and Settings\Administrator.TOM\My Documents\Visual Studio Projects\Prodaja*
- Ime Analysis Services OLAP strežnika: *Tom*
- Naziv OLAP zbirke - kocke: *ProdajaOLAP*

Slika 61: Izdelava English Query spletne strani



Vir: Ekranska slika, Internet Information Services

3. Program je izdelal mapo *C:\prodaja* in nanjo prepisal potrebne datoteke za delovanje aplikacije. Nekaj izmed pomembnih datotek je:

   - *default.htm*: Osnovna internet stran, ki preusmeri izvajanje na datoteko *default.asp*.
   - *default.asp:* V njej se določi naslov aplikacije in osnovne razdelitve ekrana računalnika - okvirji (frames), znotraj katerih se bo izvajala aplikacija. V osnovi je aplikacija ločena na dva dela: na področje, kjer se zastavlja vprašanje in področje za prikaz odgovora.
   - *params.asp*: V tej datoteki je bilo potrebno nastaviti osnovne parametre za pravilno delovanje aplikacije. Nekaj stvari se je nastavilo samodejno po vnosu podatkov pri izvajanju datoteke *setup.cmd*, ostale nastavitve, kot so uporabnik in geslo, pa je bilo potrebno preveriti in dodati (slika 62):

Slika 62: Vsebina datoteke *params.asp*

```
<!-----------------------------------------------------------------------
Magistrska naloga
Microsoft English Query internet aplikacija

Update: Tomaz Smid, 2002

------------------------------------------------------------------------>
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub SetParams
' Set to True to display debugging messages
Application("Debugging")                    = False
'
' English Query knowledge parameters
'
' Name of domain file
Application("DomainFile")                   = "prodaja.eqd"
' Set to True to ask for clarification of spelling errors
Application("ClarifySpellingErrors")        = False
'
' Database parameters
'
'System DSN for connection
Application("DSN")                                  = ""
'
'Login name to use to access database
Application("Login")                                = "sa"
'... and password
Application("PWD")                                  = ""

' OLAP Parameters
'
Application("OLAPServer")                           ="tom"
Application("OLAPDatabase")                         ="prodajaOLAP"
'
'Default ODBC connection time out (sec)
Application("QueryTimeOut")                         = 60
'
' Site parameters
'
' set to determine if we are being used in high simultaneous usage (Internet)
' or low simultaneous usage (Intranet) scenario. default is Intranet
Application("Deployment")                           = "Intranet"
Application("EMail")                                = "tomaz.smid@termo.si"
'
' Request page
'
Application("ReqWinTitle")          = "Tomaz Smid - Magistrska naloga"
Application("ReqPageTitle")         = "Tomaz Smid - Magistrska naloga"
Application("SampleQuestionsFile")  = Server.MapPath("vzorec.txt")
Application("ProblemQuestionsFile") = Server.MapPath("problemi.txt")

' Response page

Application("RespWinTitle")                 = "English Query Odgovor"
Application("RespPageTitle")                = "English Query Odgovor"

FixupConnectionString
End Sub

Sub FixupConnectionString
'
' If username is specified, remove "IntegratedSecurity" from the connection string.
'
        Dim strConnect
        Dim i, j

        strConnect = Application("SQLConnectString")
        If IsEmpty(strConnect) Then Exit Sub
        If IsEmpty(Application("Login")) Or Application("Login") = "" Then Exit Sub

        i = Instr(strConnect, "Integrated Security")
        If i =  0 Then Exit Sub
        j = i + Instr(Mid(strConnect, i), ";")
        Application("SQLConnectString") = Left(strConnect, i - 1) & Mid(strConnect, j)
End Sub
</SCRIPT>
```

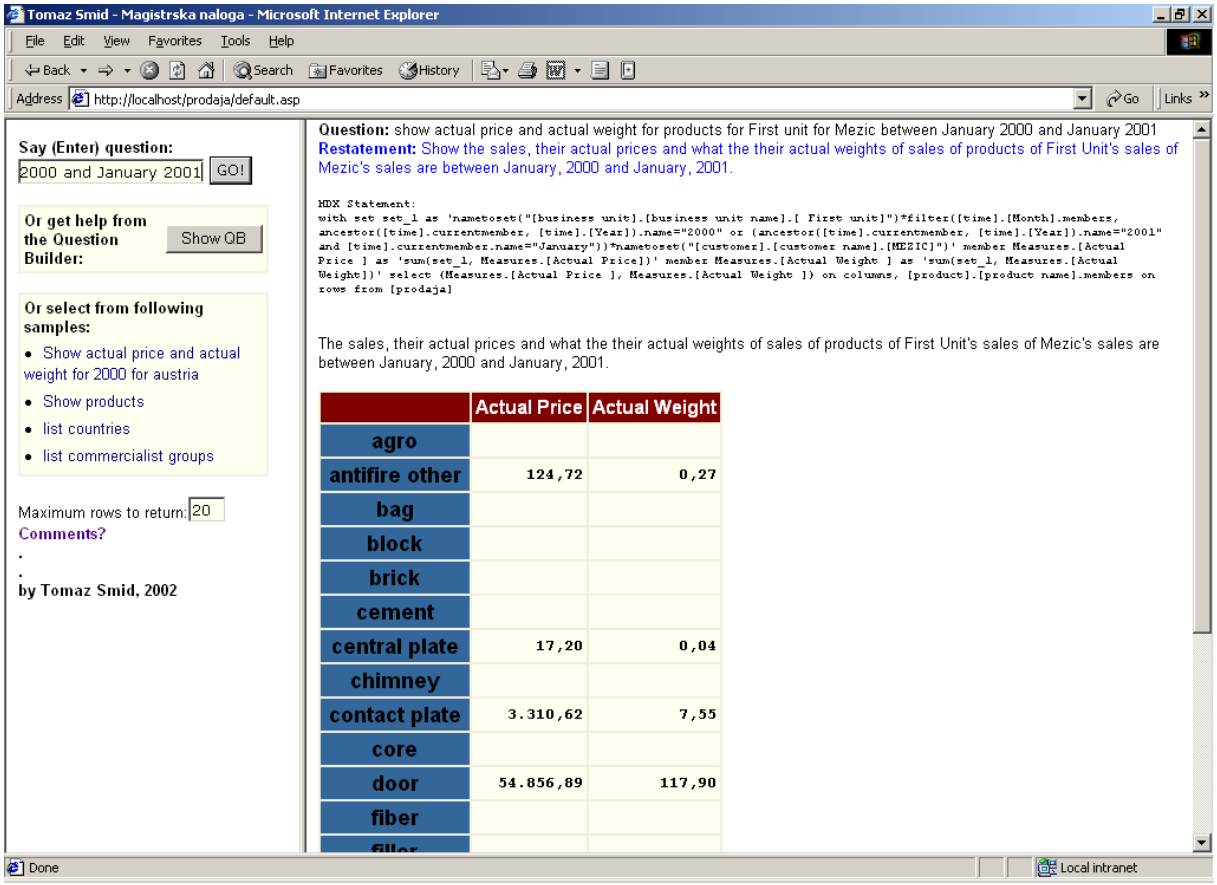
Vir: Datoteka *params.asp*

V datoteki params.asp sta bili nastavljeni dve pomembni stvari in sicer ime datoteke z vzorčnimi - vnaprej pripravljenimi vprašanji z ukazom *Application("SampleQuestionsFile")= Server.MapPath("vzorec.txt")* in ciljna datoteka za zapis vprašanj, ki ne dajo pravilnega odgovora: *Application("ProblemQuestionsFile")= Server.MapPath("problemi.txt").*

- *global.asa:* V tej datoteki so zapisane različne globalne nastavitve, ki veljajo za celotno aplikacijo.
- *request.asp* in *response.asp:* Tu so definirane nastavitve vnosnih polj in način prikaza rezultata.
- *clarify.asp:* Datoteka, ki omogoča mehanizme za razjasnjevanje vprašanja. Program npr. predlaga zamenjavo besede z neko drugo, če je vprašanje dvoumno.
- *qb*.asp:* Različne datoteke, ki so potrebne za pravilno delovanje programa *Question Builder*.

4. Aplikacijo je na koncu možno uporabljati znotraj spletnega brskalnika Microsoft Internet Explorer na naslovu *http://localhost/prodaja/default.asp*. Področje *localhost* je v naslovu uporabljeno zato, ker je aplikacija delovala kar na spletnem strežniku.
5. Končen izgled uporabniškega vmesnika je prikazan na sliki 63. Za vprašanje je uporabljeno kar problemsko vprašanje, prikazano na sliki 1, na str. 1, ki je bilo navedeno kot izhodišče raziskave, torej: *Show actual price and actual weight for products for First unit for Mezic between January 2000 and January 2001.*

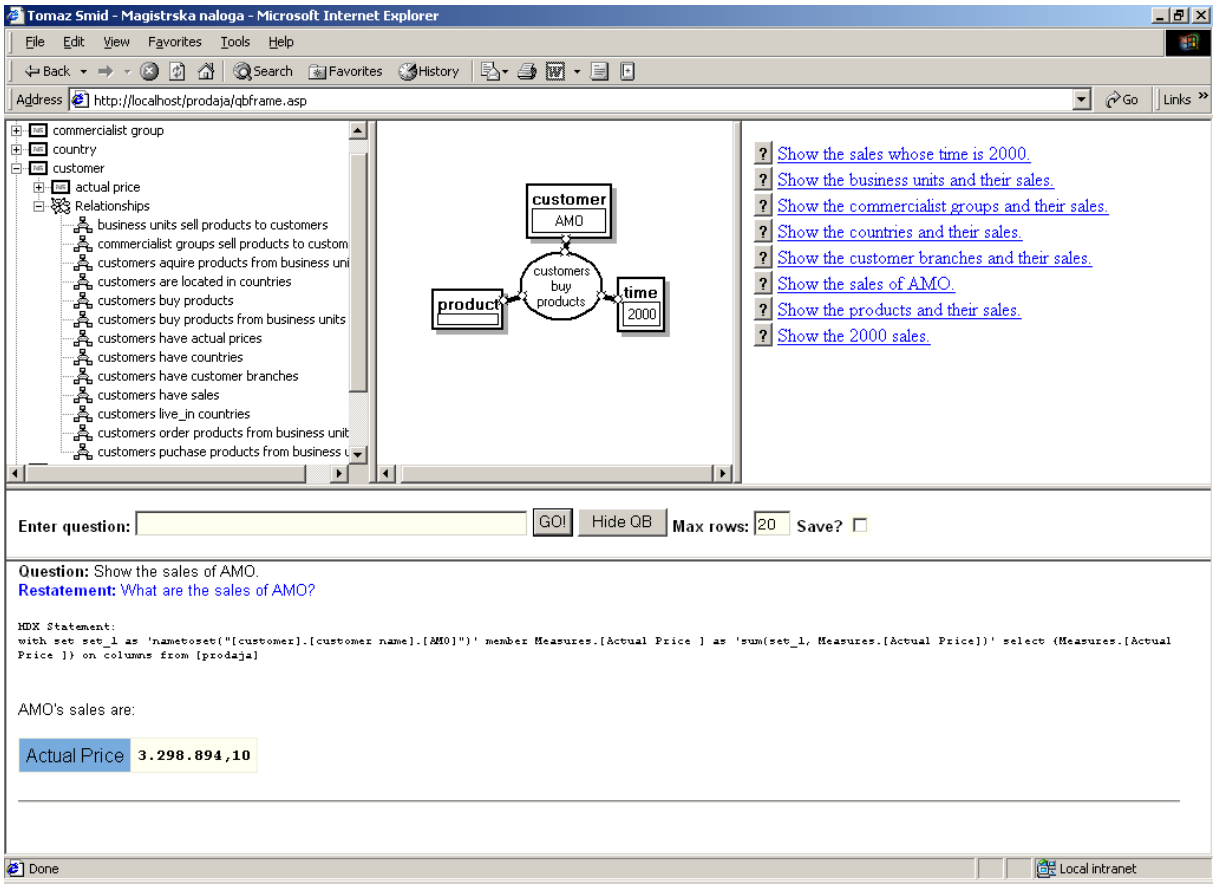
Slika 63: Primer izgleda končne aplikacije



Vir: Ekranska slika, Microsoft Internet Explorer

6. Pomoč pri izdelavi poizvedb z uporabo Question Builderja - gumb *Show QB*:
Uporablja se zato, da uporabnik sploh ve, katere so entitete, ki nastopajo v modelu, kakšne so povezave med entitetami in katera so vprašanja, ki jih uporabnik lahko zastavlja pri svojih poizvedbah. Uporaba programa, ki je napisan v JavaScript programskem jeziku, grafično orodje, ki uporabniku s pomočjo "povleci-spusti" funkcije, lahko zelo olajša kreiranje poizvedb, predvsem v prvi fazi, ko še ni vešč pravilne gradnje poizvedb.

Slika 64: Uporaba Question Builderja pri kreiranju poizvedb



Vir: Ekranska slika, Microsoft Internet Explorer

4.7. Izpopolnjevanje programa Voice Xpress

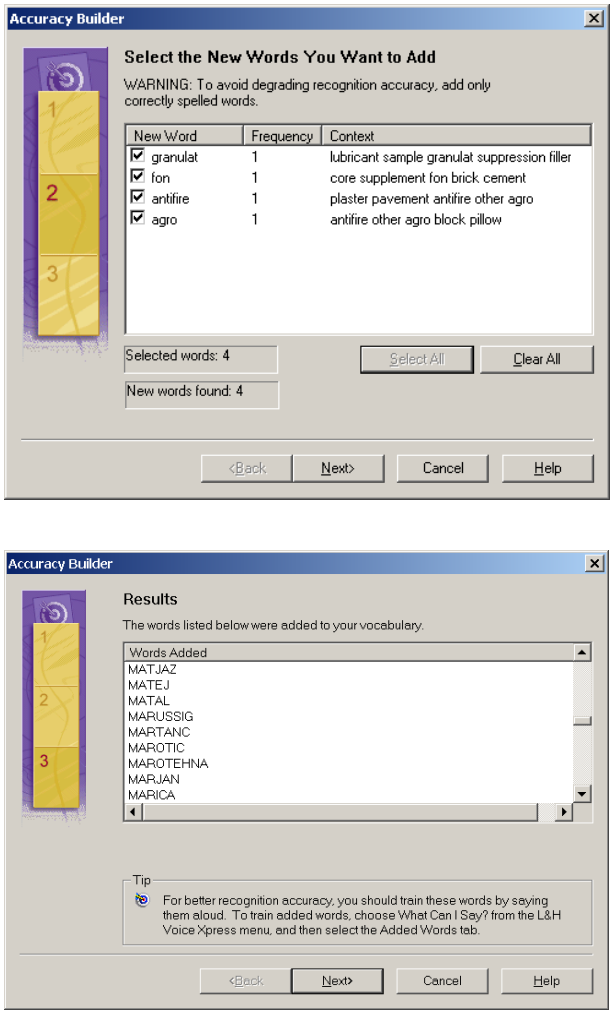
Na koncu sem funkcionalnost spletne aplikacije nadgradil še z možnostjo govornega vnosa poizvedbenega vprašanja. Program Voice Xpress je v tem trenutku sigurno eden izmed najboljših programov za razpoznavanje in sintezo človeškega govora. Posebno v primeru, če se pred uporabo upošteva vsa priporočila proizvajalca.

Upoštevana so bila vsa priporočila proizvajalca glede potrebnih opravil, ki povečajo kvaliteto razpoznavanja govora. Izkazalo se je, da je kvaliteta oziroma pravilnost

razpoznanega govora zadovoljiva, vseeno pa program še ni dovolj robusten, da bi ga uporabnik pri svojem delu lahko uporabljal vedno in povsod. Zagotoviti okoli 98% pravilnost razpoznavanja govora, ki ga omenja proizvajalec, bi bilo zelo težko realizirati, še posebno pri kreiranju bolj zapletenih vprašanj v English Query aplikacijah. Poleg precej velikega šuma, ki je ponavadi prisoten na delovnem mestu, in uporabe angleškega jezika, ki predstavlja za Slovence problem (izgovorjava, naglas), težave izvirajo predvsem iz narave English Query aplikacij, kjer se praviloma pojavlja velik nabor imen, od kupcev, imen komercialistov, krajev, držav in podobno. Ta imena se ob osveževanju OLAP kock tudi dnevno spreminjajo. Predvsem imena podjetij in izdelkov so mnogokrat umetne skovanke in okrajšave. V smislu zagotavljanja čim boljšega razpoznavanja obstajajo tri možnosti, ki pa so lahko precej zamudne:

a) Prvi način je uporaba preklopa na tako imenovani *Spelling Only* način komuniciranja z računalnikom, ki bi ga uporabnik uporabil, če bi v vprašanju želel uporabiti neko ime, ki ga ni v slovarju. Tako bi bilo npr. podjetje MEZIC GRADNJA možno črkovati. Besed, ki vsebujejo šumnike, pa z obstoječim programom zaenkrat še ni možno uporabiti.
b) Drugi način je bolj eleganten in uporablja možnost, ki sem jo od vseh znanih programov zasledil le pri programu Voice Xpress. To je uporaba takoimenovane SMF (Semantic Modeling Format) datoteke, ki jo je možno izdelati s programom Microsoft English Query kot možnost izvoza English Query projekta (*.eqp). SMF je posebna XML datoteka, kjer so zabeležene komponente semantičnega EQ modela, informacije o OLAP kockah, ki so vključene v projekt: dimenzije, hierarhični nivoji dimenzij, dodatni vnosi v slovar, prav tako je v SMF datoteki zapisana tudi vsebina regresijske datoteke, če je le ta vključena v projekt, torej tudi vprašanja uporabnika, preoblikovana vprašanja s strani programa ter MDX sintaksa. V prilogi je navedena celotna vsebina datoteke *prodaja.smf*.
c) Nove besede, ki se pojavljajo v skladišču podatkov kot osnova za gradnjo večdimenzijskih struktur, je možno izvoziti v besedilno ali XML datoteko, ki jo je s pomočjo opcije *Accuracy Builder* možno uvoziti v Voice Xpress slovar. S tem se ob vsaki spremembi nabora besed v skladišču podatkov bogati tudi nabor besed, ki jih pozna program za razpoznavanje govora (slika 65).

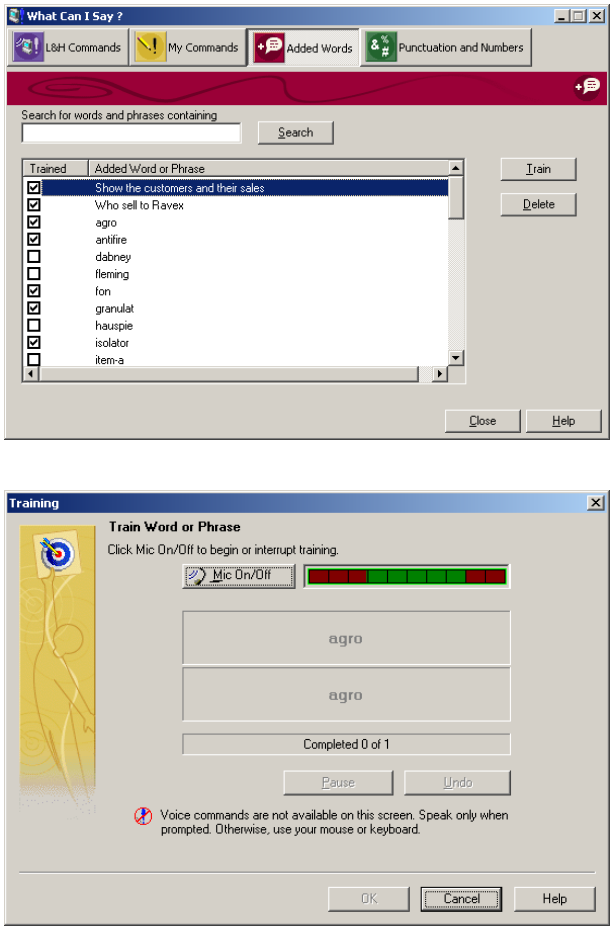
Slika 65: Dodajanje novih besed v slovar programa Voice Xpress



Vir: Ekranska slika, L&H Voice Xpress

Ni dovolj le vnos v slovar, nove besede je potrebno tudi "natrenirati" s pomočjo modula *What Can I Say* iz L&H Voice Xpress menija. Možno je dodati tudi cele fraze, stavke. Razveseljujoče je dejstvo, da z vsakim vnosom novih besed in načinom izgovorjave, sistem razpoznavanja govora postaja boljši, robustnejši.

Slika 66: Učenje pravilne izgovorjave novih besed in fraz v programu Voice Xpress



Vir: Ekranska slika, L&H Voice Xpress

5. ZAKLJUČEK

Problemska tema magistrske naloge je izbrana iz poslovnega področja, konkretno iz zagotavljanja računalniške podpore upravi podjetja pri sprejemanju strateških odločitev s poudarkom na prodajnem področju. Cilj raziskave je bil postavljen v smislu:

"Zagotoviti uporabniku, da bo lahko v naravnem jeziku vprašal računalnik za odgovore, ki so pomembni pri sprejemanju odločitev, računalnik pa mu bo odgovoril na najustreznejši možen način, ne glede na količino izvornih podatkov ali zapletenost problemskega vprašanja."

V magistrski nalogi je prikazan model ene izmed možnih rešitev, ki zahteva gradnjo več med seboj prepletenih sistemov. Prednost take zasnove je možnost preverjanja ustreznosti rešitve na različnih vmesnih stopnjah gradnje. Neodvisnost gradnikov omogoča tudi uspešen predhoden zaključek projekta na posamezni vmesni stopnji z možnostjo kasnejše nadgradnje. To je še posebno pomembno zaradi dejstva, da so nekateri deli rešitve, kot je npr. program za razpoznavanje človeškega govora, še premalo kvalitetni, da bi jih bilo možno uporabljati vedno in povsod.

Velik del magistrske naloge zavzema preverjanje ustreznosti koncepta z izdelavo konkretnega primera na osnovi realnih podatkov iz poslovnega okolja. Uspešno dokončana rešitev je potrdila ustreznost koncepta.

Magistrsko delo je pokazalo ali vsaj nakazalo, da je s pomočjo sodobnih računalniških sistemov in integriranih znanj našega poslovnega ter akademskega okolja možno izdelati uporabne sisteme za interakcijo med človekom in računalnikom na način, ki se približuje naravni komunikaciji med ljudmi.

6. POJASNILA KRATIC IN SLOVAR IZRAZOV

- **ActiveX Data Objects (ADO)** - Microsoftov podatkovni objekt
- **Active Server Pages (ASP)** - Aktivne strežniške strani
- **American National Standards Institute (ANSI)** - Ameriški nacionalni inštitut za standardizacijo
- **Application Programming Interface (API)** - Vmesnik uporabniškega programa
- **Bitmap indexing-** Način indeksiranja podatkov, ki ga uporablja Microsoft Analysis Server za indeksiranje podatkov v OLAP kockah. Zagotavlja zelo hitro pridobivanje informacij pri poizvedbah. Bitmap indeksiranje ni primerno za operacije pisanja in spreminjanja podatkov, je pa zelo primeren za branje podatkov.
- **Common Gateway Interface (CGI)** - Splošni vmesnik prehoda
- **Common Object Model (COM)** - Objektni model komponent
- **Cube** - Kocka. Del podatkov, ponavadi vzet iz skladišča podatkov, ki je organiziran in agregiran v večdimenzijsko strukturo, definirano z nizom dimenzij in mer.
- **Dimension -** Dimenzija. Del zgradbe OLAP kock, v kateri so podatki organizirani v ravneh, npr. *Država/Področje/Mesto* v dimenziji *Zemljepis*. V poročilu vrtilne tabele ali vrtilnega grafikona postane vsaka dimenzija niz polj, v katerih je mogoče razširiti ali strniti podrobnosti.
- **Distributed Component Object Model (DCOM)** - Objektni model porazdeljenih komponent
- **Entity** - Entiteta. Realen objekt, definiran s samostalnikom. Lahko je oseba, kraj, predmet, stvar ali ideja. V programu Microsoft English Query se entitete uporabljajo za kreiranje modela realnosti. Med entitetami lahko kreiramo in definiramo različne povezave.
- **Extensible Markup Language (XML) -** Oblika zapisa za dostavo bogatih, strukturiranih podatkov iz aplikacije na standarden, skladen način. XML opisuje vsebino spletnega dokumenta, medtem ko oznake HTML določajo njegov videz.)
- **Extensible Style Language (XSL)** - datoteka s slogi XML: vsebuje pravila oblikovanja, uporabljena v datoteki XML, v kateri je sklic na datoteko s slogi.
- **Hypertext Markup Language (HTML)** - Standardni označevalni jezik za dokumente v svetovnem spletu. Za določanje videza elementov strani v brskalniku, npr. besedila in grafik, in za določanje odziva na uporabnikova dejanja uporabljene oznake HTML.
- **Hyper Text Transfer Protocol (HTTP)** - Protokol za prenos hiperteksta
- **International Organization For Standardization (ISO)** - Mednarodna organizacija za standardizacijo
- **Internet Engineering Task Force (IETF)** - Internetna tehnična skupina
- **Inernet Information Server (IIS)** - Microsoftov spletni strežnik
- **Internet Server API (ISAPI)** - Vmesnik uporabniškega programa za spletni strežnik
- **Java Script (JScript)** - Microsoftov skriptni računalniški jezik Java

- **Multidimensional Expressions (MDX)** - Večdimenzijske izjave. Sintaksa, ki se uporablja za poizvedbo po večdimenzijskih strukturah podatkov (OLAP kockah). Prav tako se uporablja za kreiranje preračunanih vrednosti, za kreiranje lokalnih kock in poizvedbo po kockah z uporabo vrtilnih tabel z OLE DB ali ADO MD objekti.
- **Multidimensional Structure -** Večdimenzijska struktura. Podatkovna paradigma zbirke podatkov, ki ne obravnava podatke kot relacijske tabele, ampak kot informacijske kocke, ki vsebujejo tako izvorne podatke, informacije o dimenzijah, kot tudi sumarne podatke (agregate). Vsaka vrednost je določena s setom koordinat, ki določajo položaj neke vrednosti v strukturi. Npr. celica v koordinati {prodaja, 2000, Ljubljana, programska oprema} bo vsebovala seštevek prodaje programske opreme v Ljubljani v letu 2000.
- **Object Linking and Embeding (OLE)** - Povezovanje in vlaganje objektov
- **Online Analytical Processing (OLAP)** - Tehnologija zbirk podatkov, optimizirana za poizvedovanje in poročanje, manj pa za obdelovanje transakcij. Podatki OLAP so hierarhično organizirani in shranjeni v kockah namesto v tabelah. Uporabniki lahko vrtajo v podatke (drill down, drill up) in s tem odkrivajo značilnosti posameznih agregiranih ali detajlnih presekov dimenzij.
- **Online Transaction Processing (OLTP) -** Sistem procesiranja podatkov, ki je namenjen zajemanju in shranjevanju vseh poslovnih transakcij neke organizacije v trenutku, ko so te nastale. OLTP sistem je sposoben upravljati hkrati z več uporabniki, ki aktivno dodajajo in spreminjajo podatke.
- **Open Database Connectivity (ODBC)** - Gonilnik podatkovnega vira. Vsaka podatkovna zbirka, kot je npr. Microsoft Access, SQL server, dBASE, Paradox, Oracle itd. potrebuje svoj gonilnik po standardu Open Database Connectivity, da so lahko različni programi, kot je npr. Microsoft Query, prilagodljivi za dostop do različnih podatkovnih virov.
- **Pivot Table -** Vrtilna tabela. Interaktivno, navzkrižno poročilo v programu Microsoft Excel, v katerem so povzeti in analizirani podatki iz raznih virov, tudi zunanjih virov podatkov.
- **Relational Database Management System (RDBMS) -** Relacijska zbirka podatkov**.** Skup podatkov, ki so organizirani kot dvodimenzionalne tabele. Ena tabela je sestavljena iz večih atributov (stolpci) in dejanskih vrednosti teh atributov (vrstice). Več tabel je med seboj povezanih z relacijami. Operacije, ki jih izvaja uporabnik, so največkrat usmerjene v manipuliranje s temi relacijami v smislu kreiranja novih tabel (views, queries), da se ugotovijo nove zakonitosti take zbirke.
- **Remote Procedure Call (RPC)** - Klic procedure na daljavo
- **Rich Text Format (RTF)** - Format z obogatenim besedilom
- **Semantic Modeling Format (SMF)** - XML jezik, ki je namenjen za shranjevanje semantičnih informacij, npr. način shranjevanja Microsoft English Query projekta (entitete, relacije, kocke, vnosi v slovar, ...)

- **Structured Query Language (SQL)** - Strukturiran programski jezik za kreiranje poizvedb iz relacijskih zbirk podatkov. ANSI in ISO so definirali standard za SQL. Zadnji je bil leta 1992, zato govorimo tudi o ANSI SQL-92 jeziku.
- **Uniform Resource Identifier (URI)** - Enolični identifikator vira
- **Uniform Resource Locator (URL)** - Enolični lokator vira
- **Visual Basic Script (VBScript)** - Microsoftov skriptni računalniški jezik Visual Basic
- **Vax Management System Record Management Services (VMS RMS)** - Starejši sistem shranjevanja podatkov na računalnikih podjetja DEC.
- **World Wide Web Consortium (W3C)** - Konzorcij za svetovni splet

7. LITERATURA

1. Alwang Greg: Speek Easier with L&H5: PC Magazine reviews L&H Voice Xpress Professional Version 5. [URL:http://www.pcmag.com/article/0,2997,s%253D1671%2526a%253D4907,00.asp], PC Magazine, 1.8.2000
2. Chaffin Mark, Knight Brian, Robinson Todd: Professional SQL Server 2000 DTS (Data Transformation Service). Wrox Press Inc, oktober 2000. 855 str.
3. Codd E.F., Codd S.B. and Salley C.T.: Providing OLAP (On-Line Analytical Processing) to User-Analysts: An IT Mandate. Codd & Date, Inc, 1993
4. Dewson Robin: Beginning SQL Server 2000 Programming. Wrox Press Inc, junij 2000. 800 str.
5. Damij Talib: Tabular Application Aevelopment for Information Systems, An Object-Oriented Methodology. New York: Springer Verlag, 2001. 256 str.
6. Dobler Stefan: Speech recognition technology for mobile phones. [URL: http://www.ericsson.com/about/publications/review/2000_03/article113.shtml], Ericsson Review No. 03, 2000
7. Esposito Dino: Building Web Solutions with ASP .NET and ADO .NET. Microsoft Press, 13.2.2002. 400 str.
8. Gams Matjaž, Šef Tomaž: A Speech Module in an Agent System, Engineering Intelligent Systems, Vol. 8, No. 4, CLR Publishing, pp. 225-231, 2000
9. Grad Janez, A.Jenkins Milton: Decision Support Systems: tools, expectations and realites. Informatica, št.3. Ljubljana, november 1987. str. 18-24
10. Graves Chris et al.: Professional SQL Server 2000 Data Warehousing with Analysis Services. Wrox Press Inc, oktober 2001. 800 str.
11. Groves Simon: Business Intelligence - Description of Categories. [URL: http://www.sgroves.demon.co.uk/index.html], 26.1.2000
12. Guerrero Fernando G., Rojas Carlos Eduardo, Rojas Carlos: SQL Server 2000 Programming by Example. Que, 16.4.2001. 792 str.

13. Henderson Ken, Soukup Ron: The Guru's Guide to SQL Server Stored Procedures, XML, and HTML (With CD-ROM). Addison Wesley Professional, 21.12.2001. 576 str.
14. Henderson Kenneth W, Henderson Ken: The Guru's Guide to Transact-SQL. Addison-Wesley Pub Co, 18.2. 2000. 592 str.
15. Hunter David et al.: Beginning XML. Wrox Press Inc, november 2001. 900 str.
16. Inmon, W.H. Building Data Warehouse, New York: John Wiley&Sons, 1993. 365 str.
17. Jacobson R.: OLAP Train: Microsoft SQL Server 2000 Analysis Services Step by Step. Microsoft Press, 2000. 379 str.
18. Knight Brian: Admin911: SQL Server 2000. McGraw-Hill Professional Publishing, 21.5.2001. 369 str.
19. Lee, J.D. et al.: Speech-based Interaction with In-vehicle Computers: The Effect of Speech-based E-mail on Drivers' Attention to the Roadway. Cognitive Systems Laboratory, University of Iowa, Department of Industrial Engineering, Iowa City, Iowa, 31.5.2000
20. McEwan Noel: INT30DB Database - Introduction to Relational Database. [URL:http://ironbark.bendigo.latrobe.edu.au/secure/BITDBA/lectures/l021reldb.html], La Trobe University, Bendigo, 20.7.2001
21. McLaughlin Brett: Java & XML, 2nd Edition: Solutions to Real-World Problems. september 2001. 550 str.
22. Microsoft Corporation: Microsoft® SQL Server™ 7.0 System Administration Training Kit. Microsoft Press, 21.4.1999. 672 str.
23. Microsoft Corporation: Microsoft Sql Server 7 Data Warehousing Training Kit : McSe Training for Exam 70-019 (Training Kit). Microsoft Press, december 1999
24. Microsoft Corporation: Microsoft SQL Server 2000 Resource Kit (With CD-ROM). Microsoft Press, 28.3.2001. 1200 str.
25. Microsoft Corporation: Microsoft(r) SQL Server(tm) 7.0 Data Warehousing Online Training Kit. Microsoft Press, 15.7.2001. 450 str.
26. Minasi Mark: Mark Minasi's Windows 2000 Resource Kit. Sybex, 5.4.2000.
27. Mukherjee Joyjit: MCDBA Administering SQL Server 2000 Study Guide (Exam 70-228). McGraw-Hill Professional Publishing, 5.7.2001. 736 str.
28. Niederst Jennifer: Web Design in Nutshell. O'Reilly&Associates, 15.10.2001. 640 str.
29. Pahor, David et al.: Leksikon računalništva in informatike. Ljubljana: Pasadena, 2002
30. Pendse N., Creeth R.: The Olap Report. [URL: http://www.olapreport.com], 2001
31. Rankins Ray: Microsoft SQL Server 2000 Unleashed. Sams, 12.12. 2001. 1512 str.
32. Richard David: Effective Visual Studio .Net. Wrox Press Inc, april 2002. 650 str.
33. Richter Jeffrey: Applied Microsoft .NET Framework Programming. Microsoft Press, 23.1.2002. 500 str.
34. Seidman Claude: Data Mining with Microsoft SQL Server 2000 Technical Reference. Microsoft Press, 9.5.2001. 384 str.
35. Skinner Julian et al.: Professional ADO.NET. Wrox Press Inc, november 2001. 729 str.
36. Spofford G: MDX Solutions: With Microsoft SQL Server Analysis Services. John Wiley & Sons, 2001. 416 str.

8. VIRI

1. Business Intelligence - Business Focused-Reports, Conferences, and Exhibitions, (URL: http://www.business-intelligence.co.uk), Business Intelligence, 2001
2. Data Warehouse forum, Using the Data Warehouse for Decision Support, (URL: http://www-act.ucsd.edu/dw/forum9806/index.htm), 16.6.1998
3. IBM Corporation, IBM Via Voice Systems:News (URL: http://www-4.ibm.com/software/speech/news/20011004ems.html), 4.10.2001
4. L&H Press release (20010607): L&H signs north american publishing agreement with the Learning company, (URL:http://www.lhsl.com/news/releases/20010607_learningco.asp), L&H, 7.6.2001
5. Lernout & Hauspie Speech Products, L&H Automotive, (URL:http://www.lhsl.com/automotive/), L&H, 2002
6. Lernout & Hauspie Speech Products, L&H Press release (20000328): Lernout & Hauspie signs definitive agreement to acquire Dragon Systems, (URL:http://www.lhsl.com/sg/news/releases/20000328_dragon.asp), L&H, 28.3.2000
7. Lernout & Hauspie Speech Products, L&H Press release (20000821): Lernout & Hauspie launches iCHART internet-based services for healthcare documentation, (URL:http://www.lhsl.com/news/2000/20000821_ichart.asp), L&H, 21.8.2000
8. Lernout & Hauspie Speech Products, L&H Press release (20010530): Portugal Telecom Inovação signs licence agreement with Lernout & Hauspie, (URL:http://www.lhsl.com/news/releases/20010530_pttelecom.asp), L&H, 30.5.2001
9. Lernout & Hauspie Speech Products, L&H Press Release (20011113): Lernout & Hauspie announces L&H™ PDsay™ to voice enable pocket PC-s, (URL:http://www.lhsl.com/news/releases/20011113_pdsay.asp), L&H, 13.11.2001
10. Lernout & Hauspie Speech Products, L&H Press release (20011115): Lernout & Hauspie speech products announces breakthrough multilingual L&H™ REALSPEAK™ L&H™ versions for the Chinese market, (URL:http://www.lhsl.com/news/releases/20011115_multilanguagerealspeak.asp), L&H, 15.11.2001
11. Lernout & Hauspie Speech Products, L&H Press release (20011212): Scansoft closes acquisition of Lernout & Hauspie speech and language assest, (URL: http://www.lhsl.com/news/releases/20011212_acquisition.asp), L&H, 12.12.2001
12. Microsoft Corporation, English Query Tutorials, 2000a
13. Microsoft Corporation, Excel 2000 Help, 2000b
14. Microsoft Corporation, SQL Sever Books Online, 2000c
15. Microsoft Corporation, T3 Project Technical Overview, 2000d
16. Motor Trend News: Speech Recognition Technologies To Drive Our Cars, (URL: http://www.motortrend.com/news/stories/010607sr.html), Primedia, Inc, 2001
17. OLAP Council, The OLAP Council, (URL: http://www.olapcouncil.org/index.html), 1999

18. OLAP Council, Typical OLAP Applications, (URL:http://www.olapcouncil.org/research/typapply.htm), 1997
19. OLAP Train: Specializing in curricula for Microsoft Business Intelligence, (URL: http://www.olaptrain.com/olap.htm), OLAP Train/Aspirity, 2001
20. ProClarity - home page, (URL: http://www.proclarity.com/), ProClarity Corporation, 2001
21. Products: Seagate Info - Crystal Decisions, (URL: http://www.crystaldecisions.com/products/seagateinfo/), Crystal Decisions, 2001
22. Silsoe Research Institute: Science Groups - Mathematics and Decision Systems, (URL: http://www.sri.bbsrc.ac.uk/scigrps/sg9.htm), 2001
23. Speech Technology: Dictation Software Products_Dragon, L&H, IBM, (URL: http://www.speechtechnology.com/products/dictationsoftware/medical.html), New World Creations, 2001
24. World Wide Web Consortium (W3C). (URL: http://www.w3.org/), W3C, 1994-2002

9. KAZALO SLIK

10. PRILOGA

Celotna struktura SMF datoteke *prodaja.smf*:

```
<?xml version="1.0"?>
<MODEL NAME="prodaja" VERSION="1.2">
  <MODULE ID="MODULE:prodaja">
    <SEMANTICS>
      <ENTITY ID="ENTITY:business_unit">
        <ENTITYTYPE TYPE="PERSON"/>
        <WORD>business unit</WORD>
        <WORD>business unit name</WORD>
        <DBOBJECT LEVEL="LEVEL:business..20unit.business..20unit..20name"/>
        <NAMETYPE TYPE="PROPERNOUN"/>
        <MEMORIZENAMES/>
      </ENTITY>
      <ENTITY ID="ENTITY:commercialist_group">
        <ENTITYTYPE TYPE="PERSON"/>
        <WORD>commercialist group</WORD>
        <WORD>commercialist group name</WORD>
        <DBOBJECT LEVEL="LEVEL:commercialist..20group.commercialist..20group..20name"/>
        <NAMETYPE TYPE="PROPERNOUN"/>
        <MEMORIZENAMES/>
        <STANDSALONE/>
      </ENTITY>
      <ENTITY ID="ENTITY:country">
        <ENTITYTYPE TYPE="GEOGRAPHICAL"/>
        <WORD>country</WORD>
        <WORD>country name</WORD>
        <WORD>countryside</WORD>
        <WORD>fatherland</WORD>
        <DBOBJECT LEVEL="LEVEL:country.country..20name"/>
        <NAMETYPE TYPE="PROPERNOUN"/>
        <MEMORIZENAMES/>
        <STANDSALONE/>
      </ENTITY>
      <ENTITY ID="ENTITY:customer">
        <ENTITYTYPE TYPE="PERSON"/>
        <WORD>customer</WORD>
        <WORD>buyer</WORD>
        <DBOBJECT LEVEL="LEVEL:customer.customer..20name"/>
        <NAMETYPE TYPE="PROPERNOUN"/>
        <MEMORIZENAMES/>
        <STANDSALONE/>
      </ENTITY>
      <ENTITY ID="ENTITY:customer_branch">
        <ENTITYTYPE TYPE="PERSON"/>
        <WORD>customer branch</WORD>
        <WORD>branch</WORD>
        <WORD>bujer branch</WORD>
        <DBOBJECT LEVEL="LEVEL:customer.customer..20branch"/>
        <NAMETYPE TYPE="PROPERNOUN"/>
        <MEMORIZENAMES/>
      </ENTITY>
      <ENTITY ID="ENTITY:product">
        <WORD>product</WORD>
        <WORD>product name</WORD>
        <WORD>artifact</WORD>
        <WORD>make</WORD>
        <DBOBJECT LEVEL="LEVEL:product.product..20name"/>
        <NAMETYPE TYPE="COMMONNOUN"/>
        <MEMORIZENAMES/>
        <STANDSALONE/>
      </ENTITY>
      <ENTITY ID="ENTITY:sale">
        <ENTITYTYPE TYPE="MEASURE"/>
        <WORD>sale</WORD>
        <WORD>retail</WORD>
        <WORD>wholesale</WORD>
        <DBOBJECT CUBE="CUBE:prodaja"/>
      </ENTITY>
```

```
<ENTITY ID="ENTITY:sales_actual_price">
      <ENTITYTYPE TYPE="MEASURE"/>
      <WORD>actual price</WORD>
      <WORD>sales actual price</WORD>
      <DBOBJECT MEASURE="MEASURE:prodaja.Actual..20Price"/>
      <ATTRIBUTEOF HREF="ENTITY:sale"/>
    </ENTITY>
    <ENTITY ID="ENTITY:sales_actual_weight">
      <ENTITYTYPE TYPE="MEASURE"/>
      <WORD>actual weight</WORD>
      <WORD>sales actual weight</WORD>
      <DBOBJECT MEASURE="MEASURE:prodaja.Actual..20Weight"/>
      <ATTRIBUTEOF HREF="ENTITY:sale"/>
    </ENTITY>
    <ENTITY ID="ENTITY:sales_transport_price">
      <ENTITYTYPE TYPE="MEASURE"/>
      <WORD>transport price</WORD>
      <WORD>sales transport price</WORD>
      <DBOBJECT MEASURE="MEASURE:prodaja.Transport..20Price"/>
      <ATTRIBUTEOF HREF="ENTITY:sale"/>
    </ENTITY>
    <ENTITY ID="ENTITY:time">
      <ENTITYTYPE TYPE="DATEORTIME"/>
      <WORD>time</WORD>
      <WORD>age</WORD>
      <WORD>epoch</WORD>
      <DBOBJECT DIMENSION="DIMENSION:time"/>
    </ENTITY>
    <RELATIONSHIP ID="RELATIONSHIP:business_units_have_sales">
      <JOINTABLE/>
      <ROLE ID="ROLE:business_units_have_sales.business_unit" HREF="ENTITY:business_unit"/>
      <ROLE ID="ROLE:business_units_have_sales.sale" HREF="ENTITY:sale"/>
      <ROLE ID="ROLE:business_units_have_sales.time" HREF="ENTITY:time"/>
      <PHRASINGS>
        <TRAITPHRASING ID="PHRASING:business_units_have_sales.business_units..20have..20sales">
          <SUBJECT ROLEREF="ROLE:business_units_have_sales.business_unit"/>
          <OBJECT ROLEREF="ROLE:business_units_have_sales.sale"/>
        </TRAITPHRASING>
      </PHRASINGS>
      <WHEN ROLEREF="ROLE:business_units_have_sales.time"/>
    </RELATIONSHIP>
    <RELATIONSHIP ID="RELATIONSHIP:business_units_sell_customers_products">
      <JOINTABLE/>
      <ROLE ID="ROLE:business_units_sell_customers_products.business_unit" HREF="ENTITY:business_unit"/>
      <ROLE ID="ROLE:business_units_sell_customers_products.customer" HREF="ENTITY:customer"/>
      <ROLE ID="ROLE:business_units_sell_customers_products.product" HREF="ENTITY:product"/>
      <ROLE ID="ROLE:business_units_sell_customers_products.time" HREF="ENTITY:time"/>
      <PHRASINGS>
        <VERBPHRASING
ID="PHRASING:business_units_sell_customers_products.business_units..20sell..20customers..20products">
          <SUBJECT ROLEREF="ROLE:business_units_sell_customers_products.business_unit"/>
          <VERB>sell</VERB>
          <OBJECT ROLEREF="ROLE:business_units_sell_customers_products.customer"/>
          <OBJECT ROLEREF="ROLE:business_units_sell_customers_products.product"/>
        </VERBPHRASING>
        <VERBPHRASING
ID="PHRASING:business_units_sell_customers_products.customers..20puchase..20products..20from..20business_units">
          <SUBJECT ROLEREF="ROLE:business_units_sell_customers_products.customer"/>
          <VERB>puchase</VERB>
          <VERB>buy</VERB>
          <VERB>aquire</VERB>
          <VERB>order</VERB>
          <OBJECT ROLEREF="ROLE:business_units_sell_customers_products.product"/>
          <PREPPHRASE>
            <PREP>from</PREP>
            <OBJECT ROLEREF="ROLE:business_units_sell_customers_products.business_unit"/>
          </PREPPHRASE>
        </VERBPHRASING>
      </PHRASINGS>
      <WHEN ROLEREF="ROLE:business_units_sell_customers_products.time"/>
```

```
    </RELATIONSHIP>
    <RELATIONSHIP ID="RELATIONSHIP:commercialist_groups_have_sales">
     <JOINTABLE/>
     <ROLE ID="ROLE:commercialist_groups_have_sales.commercialist_group" HREF="ENTITY:commercialist_group"/>
     <ROLE ID="ROLE:commercialist_groups_have_sales.sale" HREF="ENTITY:sale"/>
     <ROLE ID="ROLE:commercialist_groups_have_sales.time" HREF="ENTITY:time"/>
     <PHRASINGS>
      <TRAITPHRASING ID="PHRASING:commercialist_groups_have_sales.commercialist_groups..20have..20sales">
       <SUBJECT ROLEREF="ROLE:commercialist_groups_have_sales.commercialist_group"/>
       <OBJECT ROLEREF="ROLE:commercialist_groups_have_sales.sale"/>
      </TRAITPHRASING>
     </PHRASINGS>
     <WHEN ROLEREF="ROLE:commercialist_groups_have_sales.time"/>
    </RELATIONSHIP>
    <RELATIONSHIP ID="RELATIONSHIP:commercialist_groups_sell_customers_products">
     <JOINTABLE/>
     <ROLE ID="ROLE:commercialist_groups_sell_customers_products.commercialist_group"
HREF="ENTITY:commercialist_group"/>
     <ROLE ID="ROLE:commercialist_groups_sell_customers_products.customer" HREF="ENTITY:customer"/>
     <ROLE ID="ROLE:commercialist_groups_sell_customers_products.product" HREF="ENTITY:product">
      <AMOUNT TYPE="ENTITY" HREF="ENTITY:sales_actual_price"/>
     </ROLE>
     <ROLE ID="ROLE:commercialist_groups_sell_customers_products.time" HREF="ENTITY:time"/>
     <PHRASINGS>
      <VERBPHRASING
ID="PHRASING:commercialist_groups_sell_customers_products.commercialist_groups..20sell..20customers..20products">
       <SUBJECT ROLEREF="ROLE:commercialist_groups_sell_customers_products.commercialist_group"/>
       <VERB>sell</VERB>
       <OBJECT ROLEREF="ROLE:commercialist_groups_sell_customers_products.customer"/>
       <OBJECT ROLEREF="ROLE:commercialist_groups_sell_customers_products.product"/>
      </VERBPHRASING>
     </PHRASINGS>
     <WHEN ROLEREF="ROLE:commercialist_groups_sell_customers_products.time"/>
    </RELATIONSHIP>
    <RELATIONSHIP ID="RELATIONSHIP:countries_have_sales">
     <JOINTABLE/>
     <ROLE ID="ROLE:countries_have_sales.country" HREF="ENTITY:country"/>
     <ROLE ID="ROLE:countries_have_sales.sale" HREF="ENTITY:sale"/>
     <ROLE ID="ROLE:countries_have_sales.time" HREF="ENTITY:time"/>
     <PHRASINGS>
      <TRAITPHRASING ID="PHRASING:countries_have_sales.countries..20have..20sales">
       <SUBJECT ROLEREF="ROLE:countries_have_sales.country"/>
       <OBJECT ROLEREF="ROLE:countries_have_sales.sale"/>
      </TRAITPHRASING>
     </PHRASINGS>
     <WHEN ROLEREF="ROLE:countries_have_sales.time"/>
    </RELATIONSHIP>
    <RELATIONSHIP ID="RELATIONSHIP:customer_branches_have_sales">
     <JOINTABLE/>
     <ROLE ID="ROLE:customer_branches_have_sales.customer_branch" HREF="ENTITY:customer_branch"/>
     <ROLE ID="ROLE:customer_branches_have_sales.sale" HREF="ENTITY:sale"/>
     <ROLE ID="ROLE:customer_branches_have_sales.time" HREF="ENTITY:time"/>
     <PHRASINGS>
      <TRAITPHRASING ID="PHRASING:customer_branches_have_sales.customer_branches..20have..20sales">
       <SUBJECT ROLEREF="ROLE:customer_branches_have_sales.customer_branch"/>
       <OBJECT ROLEREF="ROLE:customer_branches_have_sales.sale"/>
      </TRAITPHRASING>
     </PHRASINGS>
     <WHEN ROLEREF="ROLE:customer_branches_have_sales.time"/>
    </RELATIONSHIP>
    <RELATIONSHIP ID="RELATIONSHIP:customers_are_in_countries">
     <JOINTABLE/>
     <ROLE ID="ROLE:customers_are_in_countries.customer" HREF="ENTITY:customer"/>
     <ROLE ID="ROLE:customers_are_in_countries.country" HREF="ENTITY:country"/>
```

```
  <PHRASINGS>
   <PREPPHRASING ID="PHRASING:customers_are_in_countries.customers..20are..20in..20countries">
    <SUBJECT ROLEREF="ROLE:customers_are_in_countries.customer"/>
    <PREP>in</PREP>
    <OBJECT ROLEREF="ROLE:customers_are_in_countries.country"/>
   </PREPPHRASING>
   <TRAITPHRASING ID="PHRASING:customers_are_in_countries.customers..20have..20countries">
    <SUBJECT ROLEREF="ROLE:customers_are_in_countries.customer"/>
    <OBJECT ROLEREF="ROLE:customers_are_in_countries.country"/>
   </TRAITPHRASING>
   <VERBPHRASING ID="PHRASING:customers_are_in_countries.customers..20live..20in..20countries">
    <SUBJECT ROLEREF="ROLE:customers_are_in_countries.customer"/>
    <VERB>live in</VERB>
    <OBJECT ROLEREF="ROLE:customers_are_in_countries.country"/>
   </VERBPHRASING>
  </PHRASINGS>
 </RELATIONSHIP>
 <RELATIONSHIP ID="RELATIONSHIP:customers_buy_products">
  <JOINTABLE/>
  <ROLE ID="ROLE:customers_buy_products.customer" HREF="ENTITY:customer">
   <ALWAYSSHOW/>
  </ROLE>
  <ROLE ID="ROLE:customers_buy_products.product" HREF="ENTITY:product">
   <ALWAYSSHOW/>
   <AMOUNT TYPE="ENTITY" HREF="ENTITY:sales_actual_price"/>
  </ROLE>
  <ROLE ID="ROLE:customers_buy_products.time" HREF="ENTITY:time"/>
  <PHRASINGS>
   <VERBPHRASING ID="PHRASING:customers_buy_products.customers..20buy..20products">
    <SUBJECT ROLEREF="ROLE:customers_buy_products.customer"/>
    <VERB>buy</VERB>
    <VERB>get</VERB>
    <VERB>purchase</VERB>
    <OBJECT ROLEREF="ROLE:customers_buy_products.product"/>
   </VERBPHRASING>
  </PHRASINGS>
  <WHEN ROLEREF="ROLE:customers_buy_products.time"/>
 </RELATIONSHIP>
 <RELATIONSHIP ID="RELATIONSHIP:customers_have_customer_branches">
  <JOINTABLE/>
  <ROLE ID="ROLE:customers_have_customer_branches.customer" HREF="ENTITY:customer">
   <ALWAYSSHOW/>
  </ROLE>
  <ROLE ID="ROLE:customers_have_customer_branches.customer_branch" HREF="ENTITY:customer_branch"/>
  <PHRASINGS>
   <TRAITPHRASING ID="PHRASING:customers_have_customer_branches.customers..20have..20customer_branches">
    <SUBJECT ROLEREF="ROLE:customers_have_customer_branches.customer"/>
    <OBJECT ROLEREF="ROLE:customers_have_customer_branches.customer_branch"/>
   </TRAITPHRASING>
  </PHRASINGS>
 </RELATIONSHIP>
 <RELATIONSHIP ID="RELATIONSHIP:customers_have_sales">
  <JOINTABLE/>
  <ROLE ID="ROLE:customers_have_sales.customer" HREF="ENTITY:customer"/>
  <ROLE ID="ROLE:customers_have_sales.sale" HREF="ENTITY:sale"/>
  <ROLE ID="ROLE:customers_have_sales.time" HREF="ENTITY:time"/>
  <PHRASINGS>
   <TRAITPHRASING ID="PHRASING:customers_have_sales.customers..20have..20sales">
    <SUBJECT ROLEREF="ROLE:customers_have_sales.customer"/>
    <OBJECT ROLEREF="ROLE:customers_have_sales.sale"/>
   </TRAITPHRASING>
  </PHRASINGS>
  <WHEN ROLEREF="ROLE:customers_have_sales.time"/>
 </RELATIONSHIP>
 <RELATIONSHIP ID="RELATIONSHIP:products_have_sales">
  <JOINTABLE/>
  <ROLE ID="ROLE:products_have_sales.product" HREF="ENTITY:product"/>
  <ROLE ID="ROLE:products_have_sales.sale" HREF="ENTITY:sale"/>
  <ROLE ID="ROLE:products_have_sales.time" HREF="ENTITY:time"/>
```

```
      <PHRASINGS>
       <TRAITPHRASING ID="PHRASING:products_have_sales.products..20have..20sales">
        <SUBJECT ROLEREF="ROLE:products_have_sales.product"/>
        <OBJECT ROLEREF="ROLE:products_have_sales.sale"/>
       </TRAITPHRASING>
      </PHRASINGS>
      <WHEN ROLEREF="ROLE:products_have_sales.time"/>
     </RELATIONSHIP>
     <RELATIONSHIP ID="RELATIONSHIP:sales_actual_prices_indicate_how_strategic_customers_are">
      <JOINTABLE/>
      <ROLE ID="ROLE:sales_actual_prices_indicate_how_strategic_customers_are.customer" HREF="ENTITY:customer"/>
      <ROLE ID="ROLE:sales_actual_prices_indicate_how_strategic_customers_are.sales_actual_price"
HREF="ENTITY:sales_actual_price"/>
      <PHRASINGS>
       <ADJECTIVEPHRASING
ID="PHRASING:sales_actual_prices_indicate_how_strategic_customers_are.sales_actual_prices..20indicate..20how..20strategic..20cu
stomers..20are">
        <SUBJECT ROLEREF="ROLE:sales_actual_prices_indicate_how_strategic_customers_are.customer"/>
        <OBJECT ROLEREF="ROLE:sales_actual_prices_indicate_how_strategic_customers_are.sales_actual_price"/>
        <MEASUREMENT>
         <PLUSWORDS>
          <WORD>strategic</WORD>
          <THRESHOLD>1000000</THRESHOLD>
         </PLUSWORDS>
         <MINUSWORDS>
          <WORD>non strategic</WORD>
          <THRESHOLD>1000000</THRESHOLD>
         </MINUSWORDS>
        </MEASUREMENT>
       </ADJECTIVEPHRASING>
      </PHRASINGS>
     </RELATIONSHIP>
     <RELATIONSHIP ID="RELATIONSHIP:sales_have_sales_actual_prices">
      <JOINTABLE/>
      <ROLE ID="ROLE:sales_have_sales_actual_prices.sale" HREF="ENTITY:sale">
       <AMOUNT TYPE="ENTITY" HREF="ENTITY:sales_actual_price"/>
      </ROLE>
      <ROLE ID="ROLE:sales_have_sales_actual_prices.sales_actual_price" HREF="ENTITY:sales_actual_price"/>
      <ROLE ID="ROLE:sales_have_sales_actual_prices.time" HREF="ENTITY:time"/>
      <PHRASINGS>
       <TRAITPHRASING ID="PHRASING:sales_have_sales_actual_prices.sales..20have..20sales_actual_prices">
        <SUBJECT ROLEREF="ROLE:sales_have_sales_actual_prices.sale"/>
        <OBJECT ROLEREF="ROLE:sales_have_sales_actual_prices.sales_actual_price"/>
       </TRAITPHRASING>
      </PHRASINGS>
      <WHEN ROLEREF="ROLE:sales_have_sales_actual_prices.time"/>
     </RELATIONSHIP>
     <RELATIONSHIP ID="RELATIONSHIP:sales_have_sales_actual_weights">
      <JOINTABLE/>
      <ROLE ID="ROLE:sales_have_sales_actual_weights.sale" HREF="ENTITY:sale">
       <QUANTITY TYPE="ENTITY" HREF="ENTITY:sales_actual_weight"/>
      </ROLE>
      <ROLE ID="ROLE:sales_have_sales_actual_weights.sales_actual_weight" HREF="ENTITY:sales_actual_weight"/>
      <ROLE ID="ROLE:sales_have_sales_actual_weights.time" HREF="ENTITY:time"/>
      <PHRASINGS>
       <TRAITPHRASING ID="PHRASING:sales_have_sales_actual_weights.sales..20have..20sales_actual_weights">
        <SUBJECT ROLEREF="ROLE:sales_have_sales_actual_weights.sale"/>
        <OBJECT ROLEREF="ROLE:sales_have_sales_actual_weights.sales_actual_weight"/>
       </TRAITPHRASING>
      </PHRASINGS>
      <WHEN ROLEREF="ROLE:sales_have_sales_actual_weights.time"/>
     </RELATIONSHIP>
     <RELATIONSHIP ID="RELATIONSHIP:sales_have_sales_transport_prices">
      <JOINTABLE/>
      <ROLE ID="ROLE:sales_have_sales_transport_prices.sale" HREF="ENTITY:sale">
       <AMOUNT TYPE="ENTITY" HREF="ENTITY:sales_transport_price"/>
      </ROLE>
      <ROLE ID="ROLE:sales_have_sales_transport_prices.sales_transport_price" HREF="ENTITY:sales_transport_price"/>
      <ROLE ID="ROLE:sales_have_sales_transport_prices.time" HREF="ENTITY:time"/>
```

```
      <PHRASINGS>
       <TRAITPHRASING ID="PHRASING:sales_have_sales_transport_prices.sales..20have..20sales_transport_prices">
        <SUBJECT ROLEREF="ROLE:sales_have_sales_transport_prices.sale"/>
        <OBJECT ROLEREF="ROLE:sales_have_sales_transport_prices.sales_transport_price"/>
       </TRAITPHRASING>
      </PHRASINGS>
      <WHEN ROLEREF="ROLE:sales_have_sales_transport_prices.time"/>
     </RELATIONSHIP>
     <RELATIONSHIP ID="RELATIONSHIP:times_have_sales">
      <JOINTABLE/>
      <ROLE ID="ROLE:times_have_sales.time" HREF="ENTITY:time"/>
      <ROLE ID="ROLE:times_have_sales.sale" HREF="ENTITY:sale"/>
      <PHRASINGS>
       <TRAITPHRASING ID="PHRASING:times_have_sales.times..20have..20sales">
        <SUBJECT ROLEREF="ROLE:times_have_sales.time"/>
        <OBJECT ROLEREF="ROLE:times_have_sales.sale"/>
       </TRAITPHRASING>
      </PHRASINGS>
      <WHEN ROLEREF="ROLE:times_have_sales.time"/>
     </RELATIONSHIP>
    </SEMANTICS>
    <CUBES>
     <CUBE ID="CUBE:prodaja">
      <DIMENSIONREF HREF="DIMENSION:business..20unit"/>
      <DIMENSIONREF HREF="DIMENSION:commercialist..20group"/>
      <DIMENSIONREF HREF="DIMENSION:country"/>
      <DIMENSIONREF HREF="DIMENSION:customer"/>
      <DIMENSIONREF HREF="DIMENSION:product"/>
      <DIMENSIONREF HREF="DIMENSION:time"/>
      <MEASURE ID="MEASURE:prodaja.Actual..20Price" AGGREGATETYPE="SUM">
       <DEFAULT/>
      </MEASURE>
      <MEASURE ID="MEASURE:prodaja.Actual..20Weight" AGGREGATETYPE="SUM"/>
      <MEASURE ID="MEASURE:prodaja.Transport..20Price" AGGREGATETYPE="SUM"/>
     </CUBE>
     <DIMENSION ID="DIMENSION:business..20unit" TYPE="REGULAR">
      <LEVEL ID="LEVEL:business..20unit.business..20unit..20name" CAPITALIZATION="FIRSTLETTER"/>
     </DIMENSION>
     <DIMENSION ID="DIMENSION:commercialist..20group" TYPE="REGULAR">
      <LEVEL ID="LEVEL:commercialist..20group.commercialist..20group..20name" CAPITALIZATION="LOWER"/>
     </DIMENSION>
     <DIMENSION ID="DIMENSION:country" TYPE="REGULAR">
      <LEVEL ID="LEVEL:country.country..20name" CAPITALIZATION="UPPER"/>
     </DIMENSION>
     <DIMENSION ID="DIMENSION:customer" TYPE="REGULAR">
      <LEVEL ID="LEVEL:customer.customer..20name" CAPITALIZATION="UPPER"/>
      <LEVEL ID="LEVEL:customer.customer..20branch" CAPITALIZATION="UPPER"/>
     </DIMENSION>
     <DIMENSION ID="DIMENSION:product" TYPE="REGULAR">
      <LEVEL ID="LEVEL:product.product..20name" CAPITALIZATION="LOWER"/>
     </DIMENSION>
     <DIMENSION ID="DIMENSION:time" TYPE="TIME">
      <LEVEL ID="LEVEL:time.Year" DATETYPE="INTYEAR"/>
      <LEVEL ID="LEVEL:time.Quarter" CAPITALIZATION="FIRSTLETTER" DATETYPE="QUARTER"/>
      <LEVEL ID="LEVEL:time.Month" CAPITALIZATION="FIRSTLETTER" DATETYPE="MONTHNAME"/>
      <LEVEL ID="LEVEL:time.Day" DATETYPE="DAY"/>
     </DIMENSION>
    </CUBES>
    <DICTIONARY>
     <READSYNONYM READWORD="bush" ASWORD="forest"/>
    </DICTIONARY>
   </MODULE>
   <PROJECT>
    <DATABASE>
     <DSN>Provider=MSOLAP;Persist Security Info=False;Data Source=TOM;Connect Timeout=15;Initial
Catalog=prodajaOLAP;Client Cache Size=25;Auto Synch Period=10000</DSN>
     <DBMS TYPE="MSOLAP" VERSION="8.00.534"/>
     <LOADWORDS AMOUNT="500"/>
     <DBMAXROWS>100</DBMAXROWS>
     <DBTIMEOUT>120</DBTIMEOUT>
    </DATABASE>
```

```
    <DEFAULTS>
      <SPELLING/>
      <USERCONTEXT/>
    </DEFAULTS>
    <REGRESSION>
      <QUESTION>show actual price and transport price  for customers<RESTATEMENT>Show the customers, their total actual
prices and their total transport prices.</RESTATEMENT>
       <ANSWER>The customers, their total actual prices and their total transport prices.<QUERY>
        <![CDATA[select {Measures.[Actual Price], Measures.[Transport Price]} on columns,

 [customer].[customer name].members on rows

 from [prodaja]

]]>
      </QUERY>
    </ANSWER>
  </QUESTION>
  <QUESTION>show actual price and transport price  for customers<RESTATEMENT>Show the customers, their total actual prices
and their total transport prices.</RESTATEMENT>
  <ANSWER>The customers, their total actual prices and their total transport prices.<QUERY>
    <![CDATA[select {Measures.[Actual Price], Measures.[Transport Price]} on columns,

 [customer].[customer name].members on rows

 from [prodaja]

]]>
  </QUERY>
</ANSWER>
</QUESTION>
<QUESTION>show sales  for january 2000</QUESTION>
<QUESTION>who bought block<RESTATEMENT>Who bought block?</RESTATEMENT>
<ANSWER>The customers that bought block are:<QUERY>
<![CDATA[select {} on columns,

 nonemptycrossjoin([customer].[customer name].members, nametoset("[product].[product name].[block]"), 1) on rows

 from [prodaja]

]]>
</QUERY>
</ANSWER>
</QUESTION>
<QUESTION>show sales for customer who bought agro in february 2000<RESTATEMENT>Show the customers that bought agro in
February, 2000 and their sales.</RESTATEMENT>
<ANSWER>The customers that bought agro in February, 2000 and their sales.<QUERY>
<![CDATA[select {Measures.[Actual Price]} on columns,

 nonemptycrossjoin([customer].[customer name].members, nametoset("[product].[product name].[agro]"),
filter(nametoset("[time].[Month].[February]"), ancestor([time].currentmember, [time].[Year]).name="2000"), 1) on rows

 from [prodaja]

]]>
</QUERY>
</ANSWER>
</QUESTION>
<QUESTION>show sales for product agro in february 2000<RESTATEMENT>What were the sales of agro in February,
2000?</RESTATEMENT>
<ANSWER>The February, 2000 agro sales are:<QUERY>
<![CDATA[with

 set set_1 as 'nametoset("[product].[product name].[agro]")*filter(nametoset("[time].[Month].[February]"),
ancestor([time].currentmember, [time].[Year]).name="2000")'

 member Measures.[Actual Price ] as 'sum(set_1, Measures.[Actual Price])'

select {Measures.[Actual Price ]} on columns

 from [prodaja]

]]>
```

```
</QUERY>
</ANSWER>
</QUESTION>
<QUESTION>which bussiness units sell product agro to Ravex<RESTATEMENT>Which bussiness units sell agro to
Ravex?</RESTATEMENT>
<ANSWER>The bussiness units that sell agro to Ravex are:<QUERY>
<![CDATA[select {} on columns,

 nonemptycrossjoin([bussines unit].[bussiness unit name].members, nametoset("[customer].[customer name].[RAVEX]"),
nametoset("[product].[product name].[agro]"), 1) on rows

 from [prodaja]

]]>
</QUERY>
</ANSWER>
</QUESTION>
<QUESTION>which commercialist groups sell product agro to ravex<RESTATEMENT>Which commercialist groups sell agro to
Ravex?</RESTATEMENT>
<ANSWER>The commercialist groups that sell agro to Ravex are:<QUERY>
<![CDATA[select {} on columns,

 nonemptycrossjoin([commercialist group].[commercialist group name].members, nametoset("[customer].[customer
name].[RAVEX]"), nametoset("[product].[product name].[agro]"), 1) on rows

 from [prodaja]

]]>
</QUERY>
</ANSWER>
</QUESTION>
</REGRESSION>
</PROJECT>
</MODEL>
```

PRILOGA: Preveden opis dejavnosti podjetij (slika 3 na strani 7)

PODJETJE	OPIS DEJAVNOSTI
Adaytum	Vodilni dobavitelj rešitev za načrtovanje spletnih poslovnih aplikacij.
Cognos	Največje in najmočnejše svetovno podjetje za poslovno inteligenco
Oracle Express	Močno razvojno okolje za večdimenzijsko OLAP analizo
SAS Institute	Vodilni pri podpori odločanju in skladiščenju podatkov. Zagotavlja povezane informacijske rešitve za podjetja in rešitve elektronskega poslovanja.
WhiteLight Systems	Vodilni dobavitelj analitičnih aplikacij naslednje generacije.

Vir: The OLAP Report, 2001