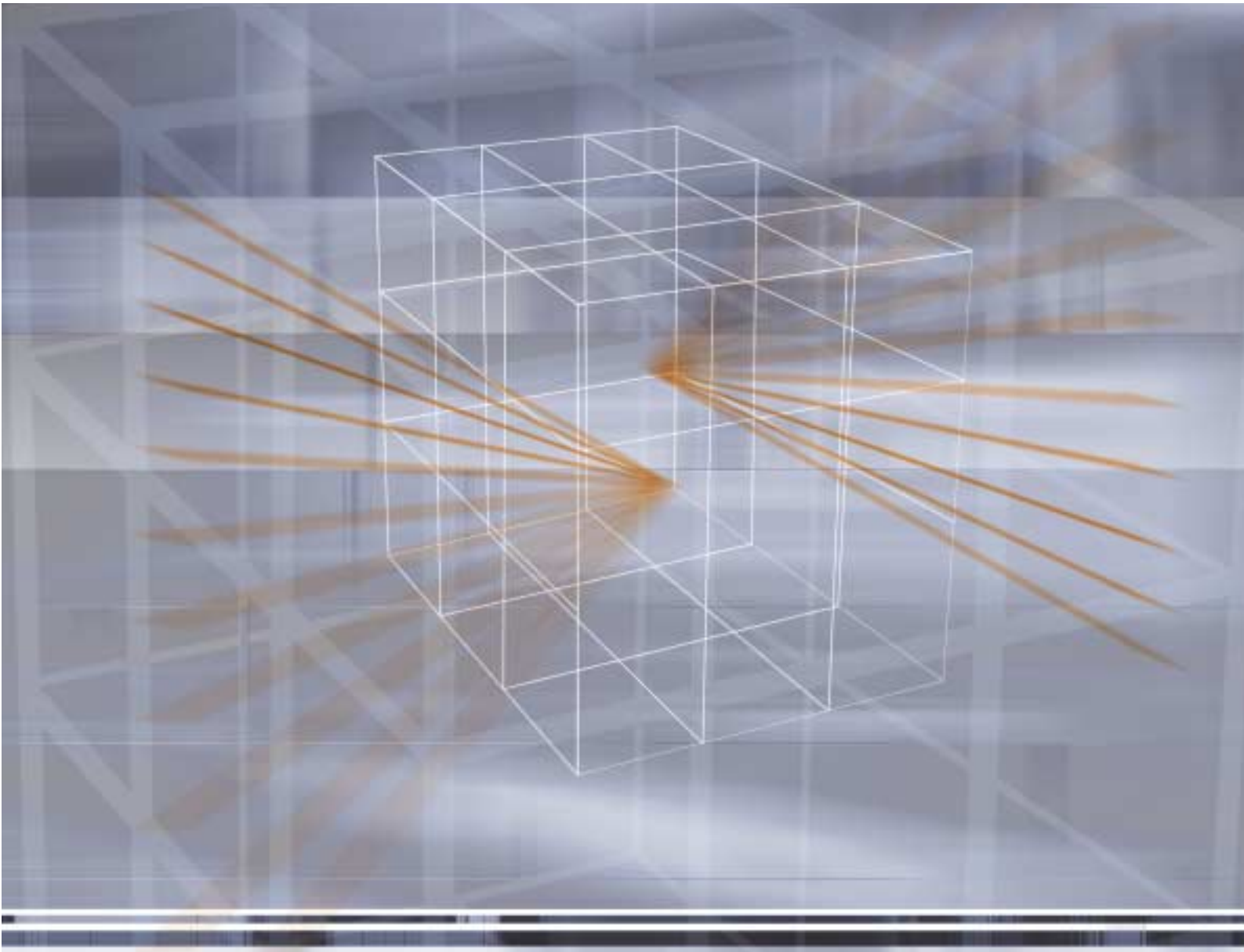
Microsoft .net

Microsoft



Ľuboslav Lacko

Analytické možnosti produktu Microsoft SQL Server 2000

Microsoft SQL Server 2000 Enterprise Edition

- Relačný databázový transakčný (OLTP) systém
- Nástroj pre extrakciu, transformáciu a loading (ETL)
- Systém pre On-line Analytical Processing
- Nástroj pre dolovanie dát
- Otvorené prostredie pre komunikáciu s klientami
- Off -line OLAP

Obsah:

Microsoft SQL Server 2000

Za týmto názvom sa skrýva výkonný databázový server dostupný pre platformu Windows. Je priamo integrovaný s používateľskými kontami a bezpečnostnými prvkami operačných systémov Windows NT, Windows 2000 a Windows XP. Microsoft SQL Server 2000 pokrýva celú škálu hardvérových platforiem pre tento operačný systém od clusterových serverov po prenosné a vreckové počítače s operačným systémom Windows CE.

SQL Server 2000 Enterprise Edition táto najvýkonnejšia verzia je určená pre najväčšie servery. Pre zaujímavosť jeden takýto databázový server dokázal obslúžiť naraz 20 000 používateľov systému SAP s odozvou 1.5 sekundy.

SQL Server Personal Edition je viacpoužívateľská verzia určená pre platformu Windows 98. Táto verzia nájde uplatnenie hlavne ako replikačný server vo firmách, ktoré majú svoje pobočky s niekoľkými klientmi.

SQL Server Developer Edition je verzia určená pre vývoj a testovanie databázových projektov..

MSDE (Microsoft Desktop Engine) je variant SQL Serveru Personal Edition bez administrátorských nástrojov. Je určený k tomu aby sa šíril spolu s aplikáciou vyvinutou napríklad v MS SQL Server Developer Edition. Pod MSDE sa dá rozbehnúť maximálne 5 súbežných procesov (dotazov od klientov). Každý ďalší proces musí čakať na ukončenie niektorého zo spomínaných piatich aktívnych procesov a navyše ešte tieto procesy značne spomaľuje. Toto obmedzenie je zavedené zámerne, z marketingových dôvodov, aby používatelia nenasadzovali túto verziu SQL Servera na výkonné systémy. Pre rôzne aplikácie, napríklad typu hobby, zberateľských katalógov, domácich archívov a podobne je to však vynikajúce riešenie. Absencia administrátorských nástrojov nie je až taký nedostatok, ako by sa mohlo na prvý pohľad zdať, pretože sa pre tento účel dá použiť databázový program **Microsoft Access 2000**, ktorý je súčasťou rozšíreného balíka Microsoft Office 2000 Professional Edition. Výhoda kompatibility MSDE s SQL serverom nespočíva len v možnosti neskoršieho prechodu aplikácie na plnú verziu. Môžeme s výhodou využiť dokumentáciu k SQL Serveru, či už v písomnej, alebo elektronickej podobe, množstvo know - how a v budúcnosti aj prípadné SQL Server Service Pack. Taktiež OLE DB a ODBC ovládače sú plne kompatibilné. Prístup k údajom v databáze je na princípe klient - server. Samotný systém riadenia databázy je systémová služba, ktorú môžeme pomocou jednoduchej dialógovej aplikácie len spustiť, prípadne pozastaviť.

Len pre úplnosť spomenieme aj program **Microsoft Access 2000**, ktorý je určený pre jednoduché „kancelárske“ aplikácie, kde sa nevyžaduje viacpoužívateľský prístup, ani podpora rozsiahlych transakcií (aj keď obidve vlastnosti podporuje). Zálohovanie údajov v databázach je možné len v „manuálnom režime“. Obtiažne je aj zotavenie sa po chybe, hlavne počas transakcie. Preto by sa Microsoft Access 2000 nemal používať vo webových projektoch. Napriek tomu sa občas používa, dôvodom je hlavne vysoká cena SQL serveru. Pre prácu s databázou môžeme využiť možnosti samotného programu Access, prípadne konzolovú utilitu **Microsoft Query**, ktorá sa nainštaluje spolu s programom MS Access a nájdeme ju v tom istom adresári ako program Access.exe. Predtým si však musíme v programe MS Access vytvoriť novú databázu, s ktorou budeme pracovať, napríklad pokus.mdb.

Microsoft SQL Server 2000 a MSDE sú databázové technológie typu klient - server, Microsoft Access 2000 je založený na zdieľaní súborov. Preto ak napriek doporučeniam predsa len použijeme túto technológiu v aplikácii typu klient - server, musíme počítať so značným zaťažením výpočtovej a pamäťovej kapacity počítača na strane klienta, nevyhneme sa ani prenosu veľkého množstva dát po sieti a ukladaniu údajov na disku klienta. Limit súčasne pripojených používateľov databázy, ktorá využíva Microsoft Access 2000 je podľa okolností niekde medzi 20 až 30.

Medzi silné stránky SQL Servera 2000 patrí škálovateľnosť a dostupnosť:

Škálovateľnosť - SQL server 2000 podporuje dve metódy škálovateľnosti

- **Scale Up** – pomocou upgrade HW vytvárame čoraz výkonnejšie mnohoprocesorové servery s väčšou pamäťou. Ich hlavnou nevýhodou je vysoká cena a problémy v prípade výpadku serveru. Technológia AWE (Address Windowing Extensions) umožňuje SQL Serveru 2000 adresovať až 64 GB operačnej pamäti. Teoreticky to nevychádza, 32 bitová adresa pokryje len 4GB pamäti. AWE technológia používa mapovanie cez virtuálny adresný priestor v 32 bitovom priestore procesora.
- **Scale Out** – predstavuje pridávanie ďalších lacných serverov. Budujeme takzvané serverové farmy. Cena rastie lineárne, v prípade serverovej farmy, nie je problém nahradiť výpadok niektorého serveru.

Dostupnosť

SQL Server 2000 obsahuje technológiu **Failover Clustering**. Dva servery majú na spoločnej zbernici (napr. SCSI) dva diskové polia. Druhý server môže byť nainštalovaný ako horúca, alebo studená záloha.

Podrobný popis nových technológií a vlastností SQL Servera 2000 by vydal na rozsiahlu publikáciu, preto viac informácií o tomto produkte hľadajte na www.microsoft.com V tejto publikácii sa zameriame na jeho analytické možnosti

Inštalácia 120 dňovej verzie Microsoft 2QL Serveru 2000

Inštalácia pre operačný systém Windows 2000, alebo Windows XP Professional je pomerne bezproblémová, stačí postupovať podľa pokynov inštalačného programu a vo väčšine prípadov akceptovať default nastavenia v jednotlivých dialógoch. Microsoft SQL Server 2000 Enterprise Trial CD Edition vyžaduje nasledovnú minimálnu hardvérovú konfiguráciu:

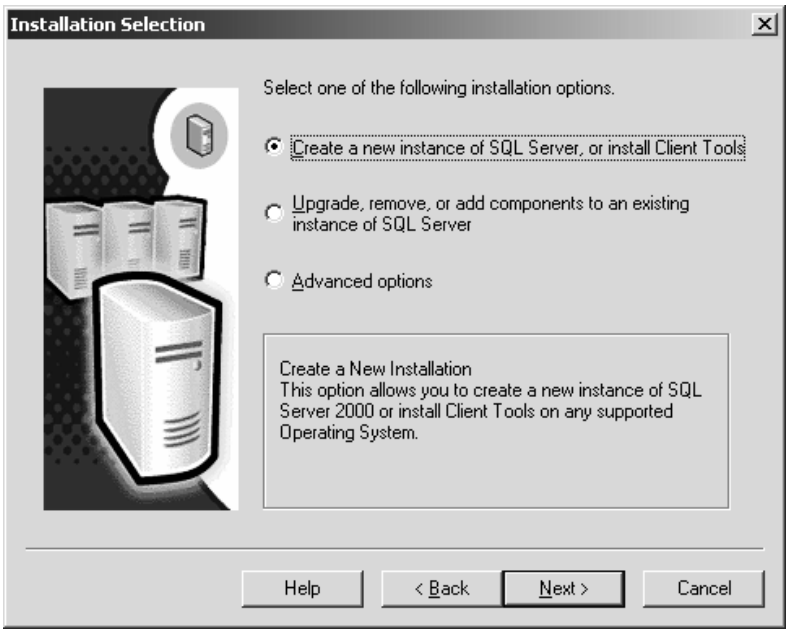
CPU: Intel kompatibilný 166 MHz a vyšší
RAM: 64 MB, lepšie 128 MB. Pre MSDE pod Windows 98 postačí 32 MB RAM
OS: Microsoft Windows NT Workstation 4.0 alebo Windows NT Server 4.0, alebo Windows NT Server 4.0 Enterprise Editions, všetko so SP5 alebo vyšším, alebo Microsoft Windows 2000 Professional, Server, Windows 2000 Advanced Server prípadne Windows 2000 Datacenter Server. (Trial verzia totiž na rozdiel od komerčnej Enterprise verzie funguje taktiež na počítačoch s Windows NT Workstations a Windows 2000 Professional). MSDE - Desktop Engine je možné nainštalovať aj na operačný systém Microsoft Windows 98 alebo Microsoft Windows Me.

Inštalácia MS SQL Servera 2000

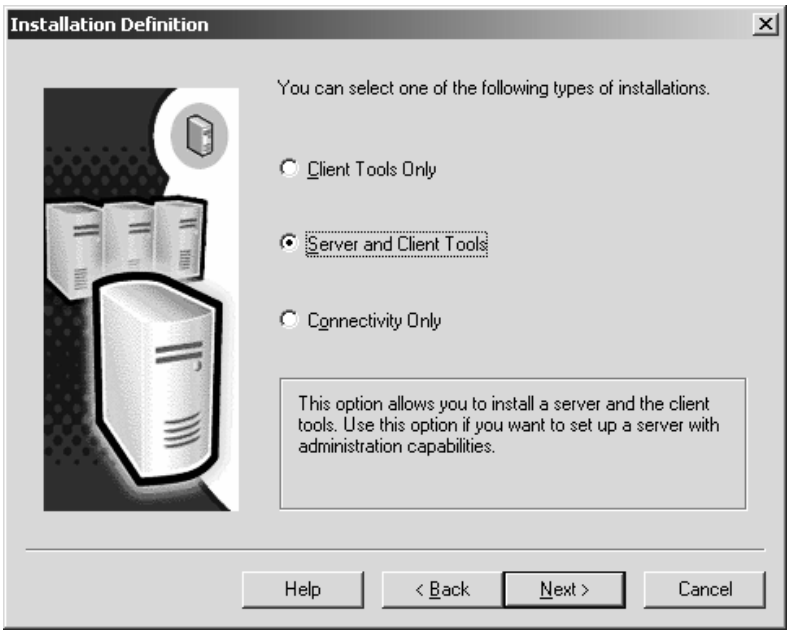
Zvolíme si inštaláciu na lokálny počítač. Vyberieme najjednoduchší spôsob inštalácie: ***Create a new instance of SQL Server, or install Client Tools***

Inštalácia MS SQL Servera 2000 II

Samozrejme musíme súhlasiť s licenčným ujednaním spoločnosti Microsoft. V našom prípade sa jednalo o licenciu typu A Limited 120-Day Edition Evaluation License. Nasleduje ďalšie rozcestie Vyberieme si preddefinovanú možnosť ***Server And Clients Tools***.

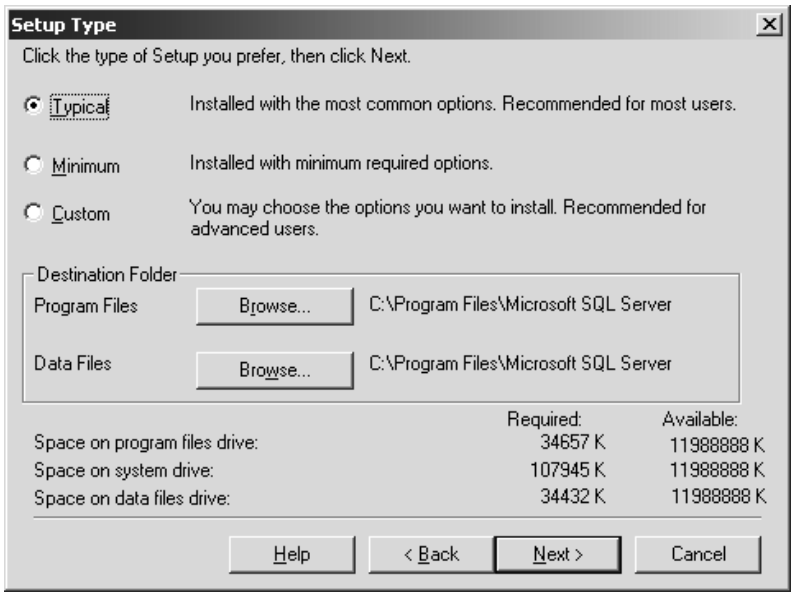


Výber typu inštalácie



V ďalšom dialógovom okne si vyberieme typickú inštaláciu. Zároveň sa dozvieme približné nároky MS SQL Serveru Presonal Edition na diskovú kapacitu, v našom prípade:

Program files drive	34 MB
System drive	107 MB
Data files drive	34 MB



Výber typu inštalácieII

V okne Services Accounts zvolíme možnosť ***Domain User Account,*** zadáme prihlasovacie heslo do systému Windows 2000. Autentifikáciu nastavíme na ***Mixed mode***, kedy sa overí autentifikácia operačného systému Windows NT, alebo Windows 2000 spolu s prihlasovacím heslom SQL Serveru. Zadáme heslo, ktoré budeme používať. Tento údaj je zároveň posledný, ktorý musíme pri inštalácii MS SQL Serveru 2000 zadať.

Business Intelligence / Data Warehouse

Pred niekoľkými rokmi sa firma Microsoft ujala dvoch iniciatív s cieľom rozšírenia data warehousingu a funkcií pre podporu rozhodovania v podnikovej sfére. Jedná sa o **Microsoft Data Warehousing Framework**, ktorý je zameraný na podporu vývoja aplikácií na platforme Microsoft. **Microsoft Alliance for Data Warehousing** je združením priemyselných partnerov, ktorých spája platforma Microsoft. Obidve iniciatívy jednak znižujú náklady na warehouse a umožňujú škálovateľnosť smerom nadol a umožňujú integráciu nástrojov, ktoré dodávajú tretie firmy. Hlavným železkom v ohni firmy Microsoft v tejto oblasti je práve MS SQL Server 2000.

Analytické Možnosti MS SQL Serveru 2000

Produkt MS SQL Server 2000 je možné zhruba rozdeliť do niekoľkých oddelených blokov
- **Microsoft SQL Server** – relačný databázový transakčný (OLTP) systém
- **Microsoft Data transformation Services** – nástroj pre extrakciu, transformáciu a loading (ETL)
- **Microsoft OLAP Server** – systém pre On-line Analytical Processing
- **Microsoft Data Mining services** – nástroj pre dolovanie dát
- **Otvorené prostredie pre komunikáciu s klientmi**, Off - line OLAP

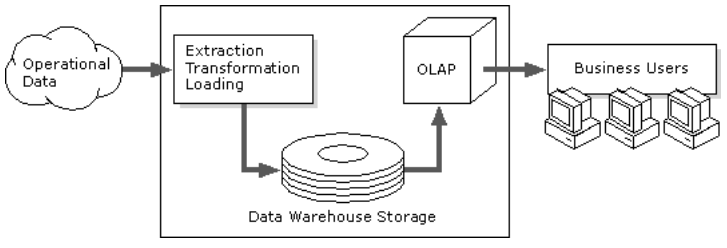
Analytické nástroje pre BI/DW dodávané s SQL Serverom 2000 je možné teda možné rozdeliť do dvoch skupín
- Microsoft Data Transformation Services
- Microsoft Analysis Services

Na úvod taktiež nezaškodí trochu definícii základných pojmov

Business Intelligence je proces transformácie údajov na informácie a prevod týchto informácii na poznatky prostredníctvom objavovania. Inými slovami povedané: účelom Business Intelligence je konvertovať veľké objemy údajov na poznatky, ktoré sú potrebné pre koncových používateľov. Tieto poznatky môžeme potom efektívne využiť napríklad v procese rozhodovania. Pod pojmom informácia nerozumieme len konkrétny záznam, alebo množinu záznamov. Často potrebujeme sledovať trend nejakej veličiny, napríklad pri obchodovaní s cennými papiermi, alebo potrebujeme nájsť medzi údajmi určité závislosti. Preto moderné databázové servery obsahujú rozsiahlu podporu pre **OLAP** (Online Analytical Processing), **Data Mining** (dolovanie, odkrývanie dát) a **Data Warehouse** (dátové sklady)

Dátový sklad je podnikovo štruktúrovaný depozitár subjektovo orientovaných, integrovaných, časovo premenných, historických dát použitých na získavanie informácií a podporu rozhodovania. V dátovom sklade sú uložené atomické a sumárne dáta. *(definícia dátového skladu podľa Billa Inmona)*. Údaje sa získavajú a ukladajú do produkčných (operačných) databáz, ktoré môžu byť v rôznych oddeleniach firiem, alebo dokonca v rozličných geografických lokalitách. Tieto údaje v pravidelných intervaloch zozbierame, predspracujeme a zavedieme do dátového skladu. Dátový sklad je v podstate tiež databáza, len je organizovaná podľa trochu iných pravidiel, tabuľky napríklad nemusia byť normalizované a podobne.

Obrázok 2.0 – Princíp dátového skladu

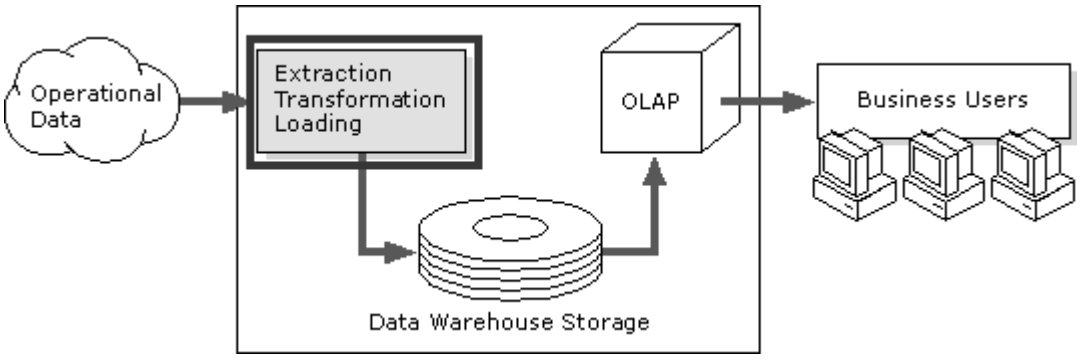


OLTP (Online Transaction Processing) – pod touto skratkou sa skrývajú tzv. transakčné databázy, t. j. databázy, ktorých primárnym účelom je vykonávanie veľkého množstva on - line transakcií, napríklad bankových, obchodných a podobne.

OLAP (Online Analytical Processing) – táto skratka zahŕňa štruktúry údajov a analytické služby, ktoré slúžia pre analýzu veľkého množstva údajov. Podrobnejšie si tento pojem rozoberieme v ďalšej kapitole.

Nástroje pre ETL (Extrakcia, Transformácia a Loading) –

S nasadením technológie business intelligence a data warehouse (BI/DW) prakticky nikdy nezačíname ako sa hovorí „na zelenej lúke“. Obvykle sa pred zavedením týchto technológii pre procesy rozhodovania používajú údaje získavané z primárnych transakčných systémov **OLTP** (On-Line Transaction Processing) v lepšom prípade spracované do zostáv. Tieto zostavy sú potom (spravidla ručne alebo pomocou softvéru typu office) spracovávané do manažérskych podkladov pre účely rozhodovania. Údaje pre proces business intelligence a data warehouse teda pochádzajú z rôznych nehomogénnych zdrojov. Môžu to byť údaje zo súborových databáz (Access, dBase...), údaje z databáz spravovaných niektorým databázovým serverom (Oracle, Informix, Microsoft SQL Server, Sybase, Interbase, Ingres...). Môžu to byť údaje vyexportované nejakou databázovou platformou do tzv. flat file. Pod týmto záhadným pojmom sa skrýva textový súbor, kde sú údaje oddelené nejakým oddeľovačom (čiarka, medzera, tabulátor), prípadne môžeme použiť aj delimitované súbory, kde vychádzame z pevnej štruktúry údajov. Nástroje a postupy **ETL** (Extraction, Transformation & Loading), prípadne vyjadrené inou terminológiou **ETT** (Extraction, Transformation Transport) sú preto veľmi dôležitou súčasťou každého projektu dátového skladu.



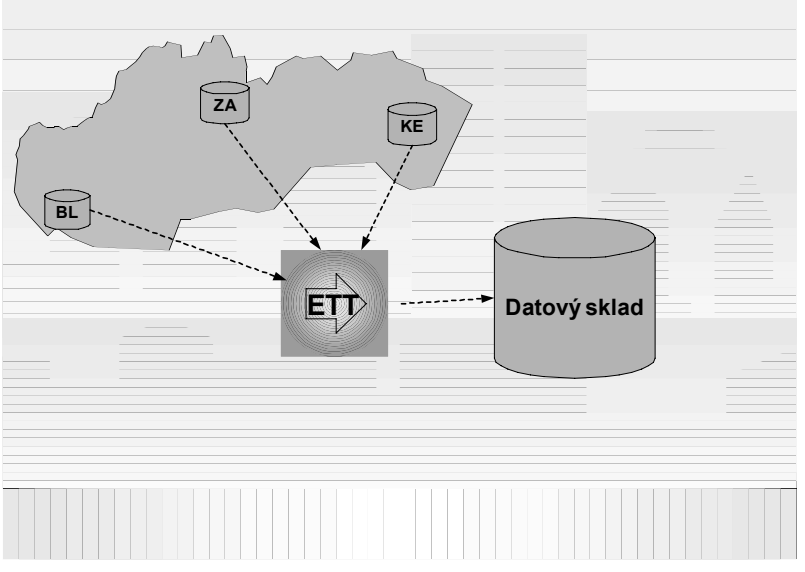
Obrázok 2.1 – miesto procesu ETL v dátovom sklade

Hlavnou úlohou procesu ETL je naplnenie dátového skladu určenými údajmi v požadovanom čase. Z konkrétnych nástrojov môžeme spomenúť Microsoft DTS (Data Transformation Services), DPS (Data Pipeline Services) od českej firmy Adastra, Oracle Warehouse Builder (OWB), a mnohé ďalšie.

Ak sa bližšie pozrieme na jednotlivé etapy procesu ETL
- **Extrakcia** – výber dát prostredníctvom rôznych metód
- **Transformácia** – overenie, čistenie, integrovanie a časové označenie dát
- **Loading** – premiestnenie dát do dátového skladu

Hlavným cieľom etapy ETL je centralizácia údajov, t.j ich zhromaždenie z mnohých spravidla nehomogénnych a rôznorodých zdrojov z OLTP databáz.

Obrázok 2.2 – príklad ETL



Najlepšie si možnosti DTS a používateľské rozhranie príslušných aplikácií ukážeme na niekoľkých príkladoch.

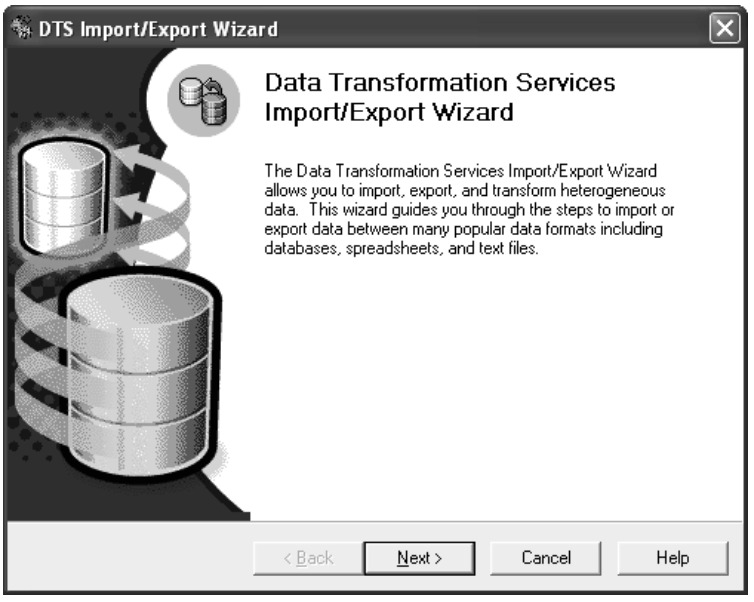
Príklad pre import cvičnej databázy FoodMart.

Cvičná databáza FoodMart sa nainštaluje pri inštalácii analytických služieb. Túto databázu využijeme aj v kapitole venovanej OLAP analýze, takže odporúčame tento jednoduchý príklad precvičiť. Databáza je vo formáte MS Access, t.j. súbor s príponou MDB. Má približne 28 megabajtov a nájdeme ho v adresári ***C:\Program Files\Microsoft Analysis Services\Samples***. Aby sme si tento príklad zjednodušili, môžeme súbor FoodMart 2000.mdb prekopírovať do novovytvoreného adresára . ***C:\FoodMart***. Našou úlohou je preniesť údaje vrátane štruktúr do novovytvorenej databázy SQL Serveru 2000 s názvom Supermarket. Pre úplnosť, novú databázu vytvoríme napríklad pomocou aplikácie Enterprise Manager, pomocou položky menu New Database. Príslušné menu sa zobrazí po kliknutí pravým tlačidlom myši na zložku Databases v ľavom okne aplikácie.

Použijeme dátovú pumpu **DTS** (Data Transformation Services). Čo sa týka používateľského rozhrania, máme k dispozícii dve možnosti, pričom stále pôjde o ten istý proces.

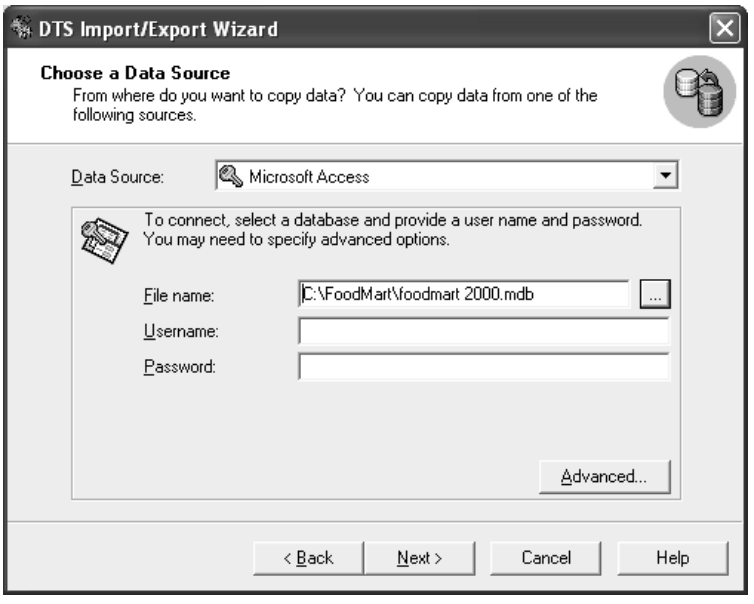
- **Import and Export Data** – táto utilita je dostupná z hlavného Windows menu SQL Servera *(Štart - MS SQL Server - Import and Export Data)*

- **DTS Package** – táto zložka je dostupná napríklad z aplikácie SQL Server Enterprise Manager. (Windows menu: *Štart – MS SQL Server – Enterprise Manager)*



V našom prípade, pretože sa jedná o viac tabuliek, ktoré potrebujeme preniesť vrátane ich štruktúr, pričom nepredpokladáme nijaké transformácie, nakoľko sa jedná o cvičný príklad pre prenos údajov medzi dvoma databázovými platformami tej istej firmy použijeme aplikáciu **DTS Import/Export Wizard.**

Obrázok 2.3 – Úvodná obrazovka DTS Import/Export Wizard



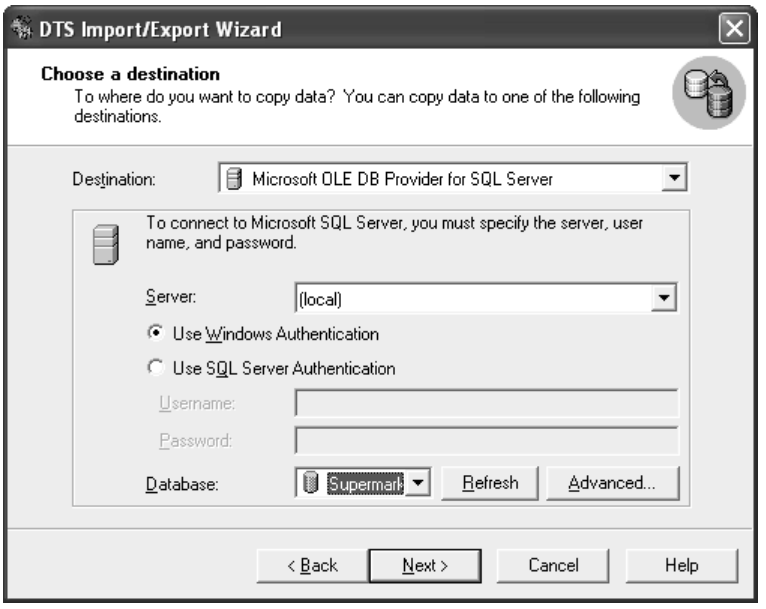
Najskôr musíme špecifikovať zdroj údajov, v našom prípade databázu Access. V dialógovom okne zadáme umiestnenie zdroja údajov, v našom prípade ***C:\FoodMart\foodmart 2000.mdb***

Pri výbere dátového zdroja si všimnime širokú paletu rôznych súborových databáz, ale aj databázových platforiem, ktoré sú službou DTS podporované. Ďalšie možnosti poskytuje rozhranie ODBC (Open Database Connectivity)

Obrázok 2.4 – výber zdroja údajov.

V ďalšom logickom kroku špecifikujeme cieľovú destináciu pre transformované údaje, v našom prípade to bude novovytvorená databáza Supermarket pod správou MS SQL Servera 2000. V reálnej aplikácii by sme museli zadať aj parametre pre prístup k údajom. V našom cvičnom príklade sme všetko ponechali na autentifikáciu operačného systému Windows.

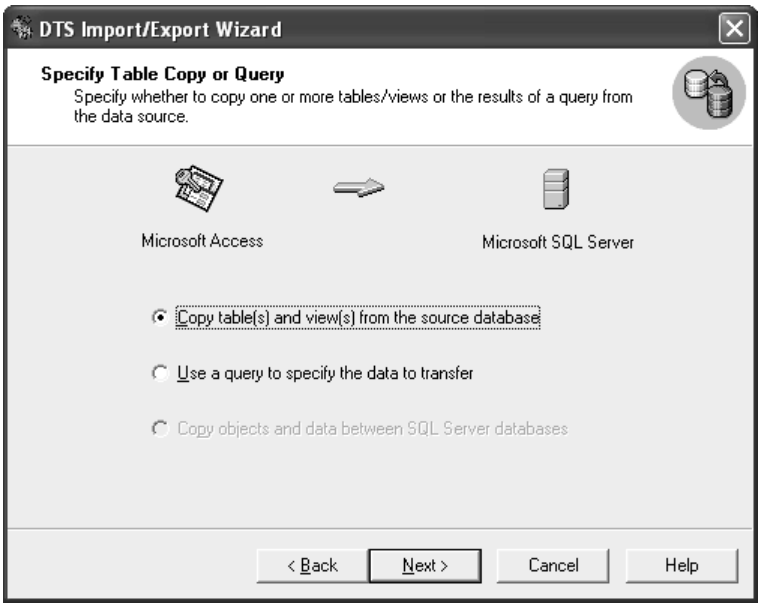
Obrázok 2.5 - výber destinácie.



V nasledujúcom kroku máme možnosť zvoliť si metódu pre výber množiny údajov, ktoré potrebujeme preniesť a prípadne pretransformovať. Máme na výber dve možnosti:

- kopírovať vybrané tabuľky a pohľady ako celky zo zdroja dát do cieľovej databázy
- pomocou Query dotazu špecifikovať podmnožiny údajov, ktoré potrebujeme pretransformovať.

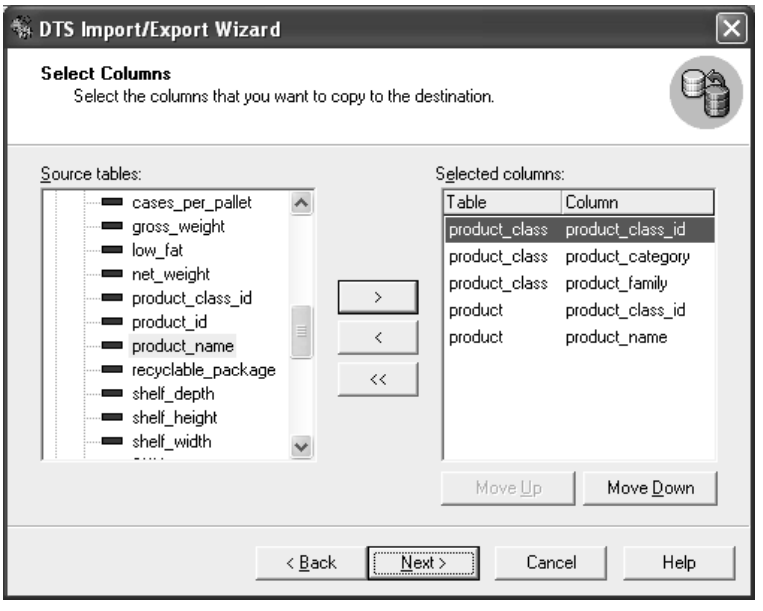
Obrázok 2.6 – výber objektov pre prenos údajov.



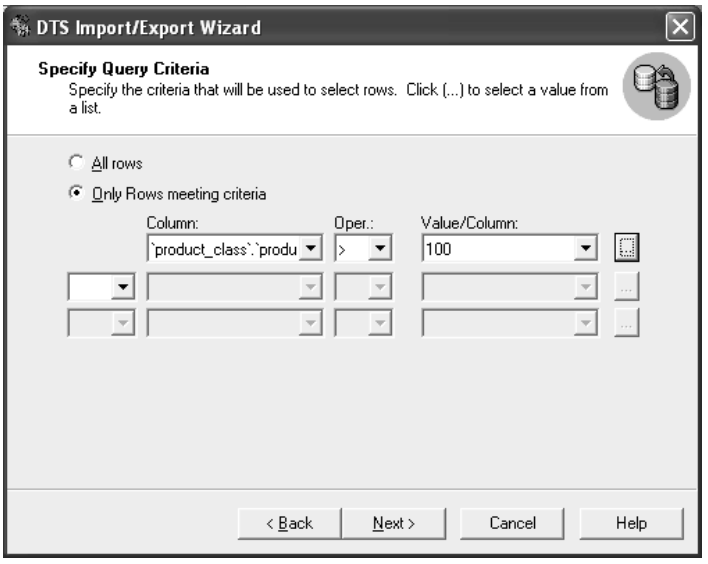
V našom príklade potrebujeme preniesť štruktúry a údaje zo všetkých tabuliek, takže ponecháme označenú prvú variantu ***Copy table(s) and view(s) from the source database.***

Nie vždy však máme také šťastie ako v tomto prípade, kedy cvičná databáza FoodMart obsahuje rozumné množstvo údajov a to dokonca veľmi vhodne štruktúrovaných. Pre proces ETL sú skôr charakteristické rôzne transformácie a očisťovanie údajov, aby sme získali údaje vhodné pre OLAP analýzu. Preto nahliadneme aj do druhej vetvy, kedy môžeme vytvoriť SQL dotaz pre výber údajov, ktoré sa budú transformovať. Tento postup sa hodí napríklad v prípade, kedy už máme vytvorený SQL dotaz pre výber údajov, ktoré nás zaujímajú, alebo ho vieme na základe príslušných požiadaviek jednoducho vytvoriť. Po potvrdení voľby: **Use a query to specify the data to transfer** môžeme zadať SQL dotaz priamo do nasledujúceho dialógového okna, alebo využiť možnosti skupiny dialógových okien s názvom Query Builder, ktoré slúžia pre interaktívne vytvorenie SQL dotazu. V prvom kroku vyberieme požadované stĺpce s príslušných tabuliek

Obrázok 2.7 - Query Builder - výber stĺpcov.



Vytvorenie SQL dotazu dokončíme zadaním obmedzujúcich podmienok pre výber údajov a v prípade potreby stanovíme aj poradie stĺpcov pre utriedenie záznamov.



Obrázok 2.8 - Query Builder - vytvorenie podmienky pre výber údajov.

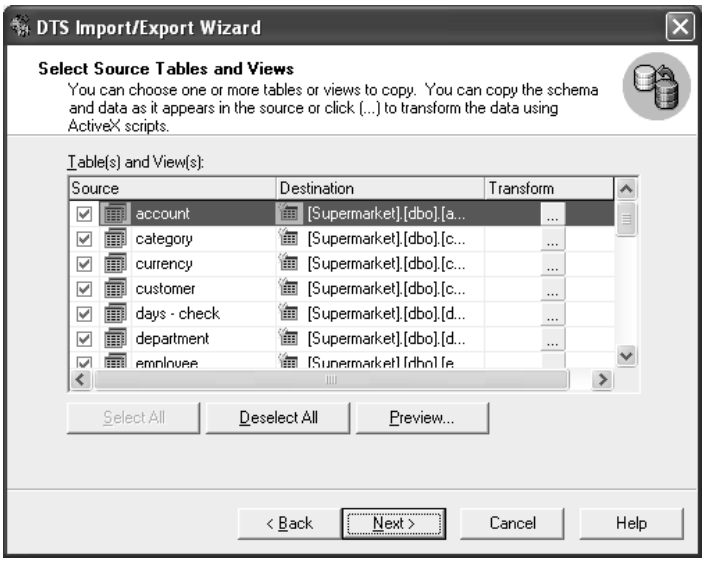
Výsledkom nášho jednoduchého pokusu bol napríklad SQL dotaz:

```
select `product_class`.`product_class_id`, `product_class`.`product_category`, `product_class`.`product_family`, `product`.`product_class_id`, `product`.`product_name`
from `product_class`,`product`
where `product_class`.`product_class_id`>100
order by `product_class`.`product_class_id`
```

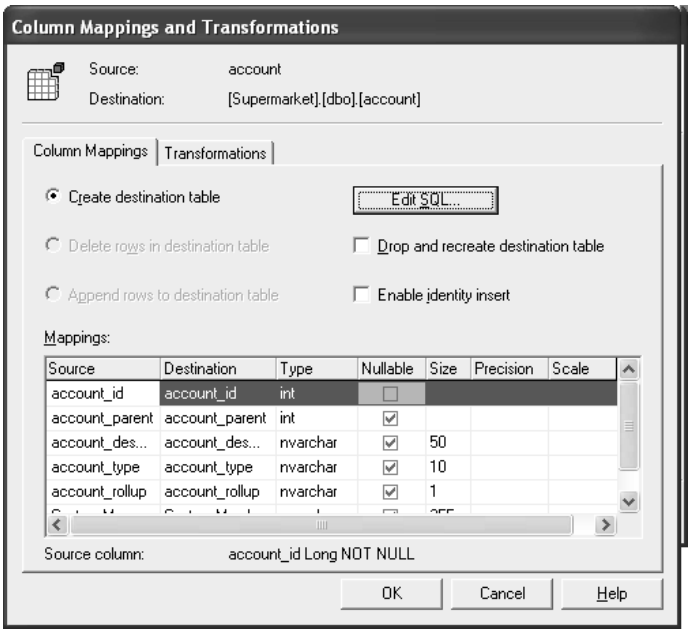
Vráťme sa však k cvičnému príkladu, v ktorom potrebujeme preniesť celé tabuľky.

Obrázok 2.9 - výber tabuliek a pohľadov.

Všimnime si tretí stĺpec tabuľky s názvom **Transform**. Po kliknutí na tlačidlo s tromi bodkami môžeme definovať pravidlá pre transformáciu príslušnej tabuľky.



Obrázok 2.10 - výber jednotlivých stĺpcov.

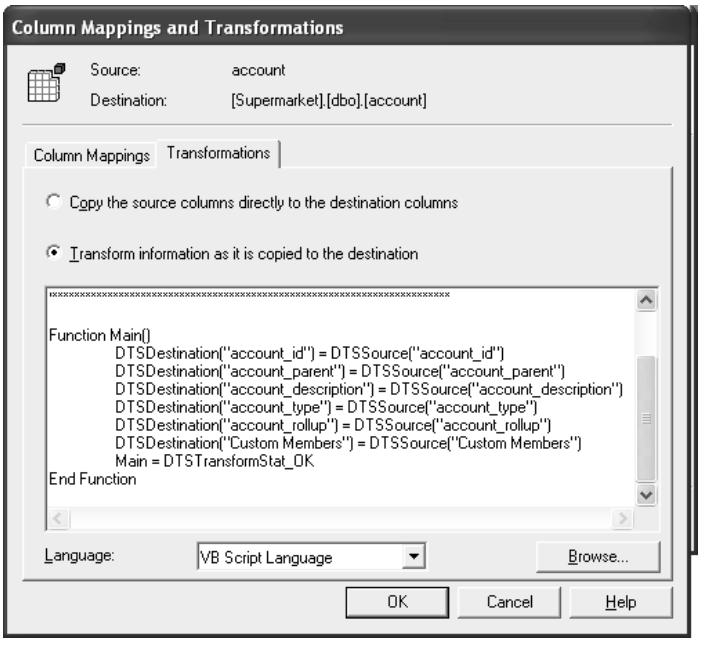


Výsledkom výberu všetkých stĺpcov je SQL dotaz:

```
  CREATE TABLE
[Supermarket].[dbo].[account]
  (
    [account_id] int NOT NULL,
    [account_parent] int NULL,
    [account_description] nvarchar (50)
NULL,
    [account_type] nvarchar (10) NULL,
    [account_rollup] nvarchar (1) NULL,
   [Custom Members] nvarchar (255) NULL
  )
```

Druhá záložka dialógového okna umožňuje definovať skript pre transformáciu údajov

Obrázok 2.11 - skript pre transformáciu údajov.



Na výber máme dokonca k dispozícii viac skriptovacích jazykov, napríklad **VBScript** (default), alebo **JScript**. Ukážeme si pre zaujímavosť príklad jednoduchej transformácie typu 1:1

```
'**********************************************************************
'  Visual Basic Transformation Script
'  Copy each source column to the
'  destination column
'**********************************************************************

Function Main()
    DTSDestination("account_id") = DTSSource("account_id")
    DTSDestination("account_parent") = DTSSource("account_parent")
    DTSDestination("account_description") = DTSSource("account_description")
    DTSDestination("account_type") = DTSSource("account_type")
    DTSDestination("account_rollup") = DTSSource("account_rollup")
    DTSDestination("Custom Members") = DTSSource("Custom Members")
    Main = DTSTransformStat_OK
End Function
```

Zdalo by sa, že cvičná databáza FoodMart neposkytuje žiadne možnosti pre transformáciu. Ak sa však pozornejšie pozrieme na tabuľku ***customer***, ktorú pri OLAP analýze v ďalšej kapitole využijeme ako tabuľku dimenzií regionálneho typu, budeme mať mierne problémy, pretože meno a priezvisko zákazníka máme uložené v dvoch stĺpcoch tabuľky a my budeme potrebovať špecifikovať obidva atribúty meno aj priezvisko ako jednu dimenziu.

Príklad údajov v tabuľke **customer**:

customer_id	lname	fname	city	state_province
1	Nowmer	Sheri	Tlaxiaco	Oaxaca
2	Whelply	Derrick	Sooke	BC
3	Derry	Jeanne	Issaquah	WA
4	Spence	Michael	Burnaby	BC
5	Gutierrez	Maya	Novato	CA
6	Damstra	Robert	Lynnwood	WA

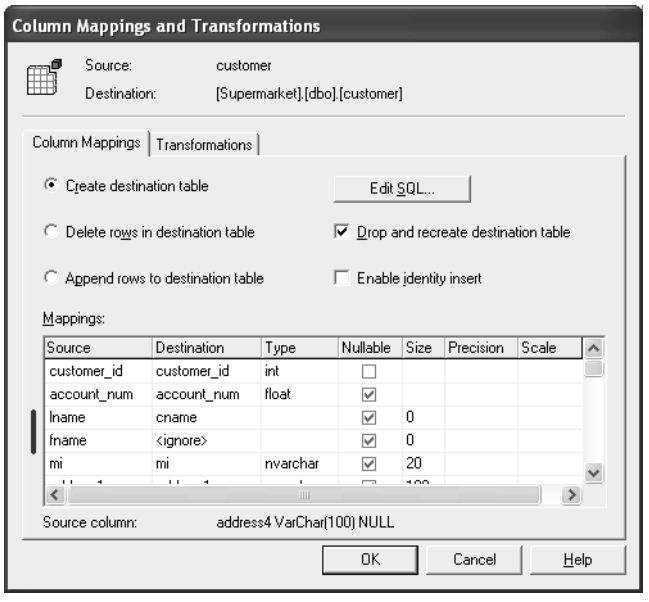
Preto napíšeme jednoduchý skriptový kód, pomocou ktorého zlúčime obidva stĺpce do jedného s názvom **cname**. Skôr než pristúpime k vytvoreniu skriptu pre zlúčenie dvoch stĺpcov, musíme modifikovať skript pre vytvorenie tabuľky v cieľovej databáze. Namiesto pôvodného príkazu CREATE TABLE:

```
CREATE TABLE [Supermarket].[dbo].[customer]
(
[customer_id] int NOT NULL,
[account_num] float NULL,
[lname] nvarchar (100) NULL,
[fname] nvarchar (50) NULL,
...
```

použijeme mierne modifikovaný, v ktorom namiesto stĺpcov **lname** a **fname** vytvorime jeden stĺpec s názvom **cname** a samozrejme s dostatočnou dimenziou rozsahu. Vidíme, že stĺpec **lname** mal povolený rozsah 100 znakov a stĺpec fname 50. Ak k tomu prirátame jeden znak na medzeru medzi priezviskom a menom, zistíme, že pre bezpečnú transformáciu potrebujeme, aby stĺpec cname mal minimálne 151 znakov. Modifikovaná časť príkazu teda bude v tvare:

```
CREATE TABLE [Supermarket].[dbo].[customer]
(
[customer_id] int NOT NULL,
[account_num] float NULL,
[cname] nvarchar (151) NULL,
...
```

Taktiež je potrebné zmeniť mapovanie stĺpcov napríklad takto (zmeny sú v obrázku vyznačené zvislou čiarou vľavo od dotknutých stĺpcov):



Obrázok 2.12 - úprava mapovania stĺpcov.

Skript pre samotné zlúčenie stĺpcov potom bude veľmi jednoduchý, zmeníme len hrubo označený riadok:

```
Function Main()
 DTSDestination("customer_id") = DTSSource("customer_id")
 DTSDestination("account_num") = DTSSource("account_num")
 DTSDestination("cname")= Trim(DTSSource("lname")) + " " + Trim(DTSSource("fname"))
 DTSDestination("mi") = DTSSource("mi")
            ...
 Main = DTSTransformStat_OK
End Function
```

O úspechu vykonanej transformácie sa neskôr môžeme veľmi ľahko presvedčiť SQL príkazom:

```
select customer_id, cname, city from customer;
```

použitím ktorého napríklad prostredníctvom aplikácie SQL Query Analyzer získame údaje:

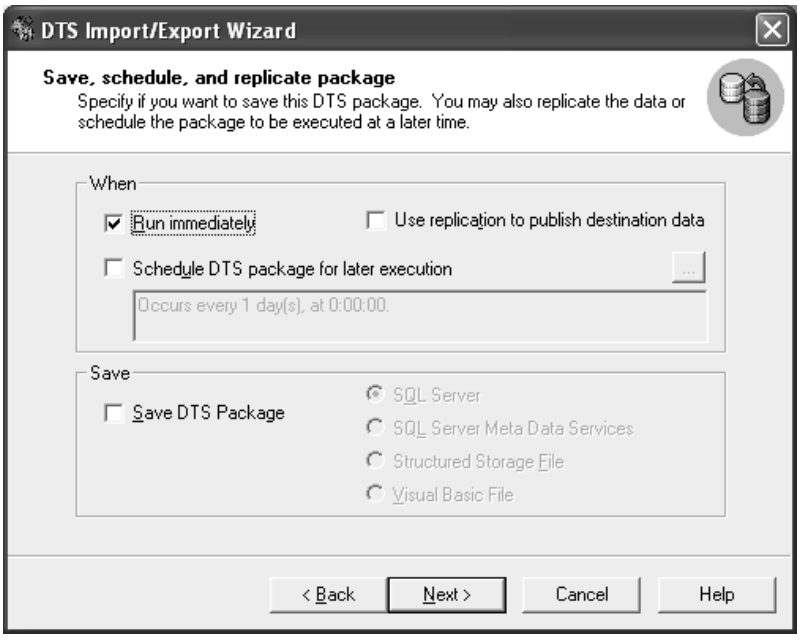
```
customer_id cname
----------- --------------------  ------------------------
1           Nowmer Sheri          Tlaxiaco
2           Whelply Derrick       Sooke
3           Derry Jeanne          Issaquah
4           Spence Michael        Burnaby
...
```

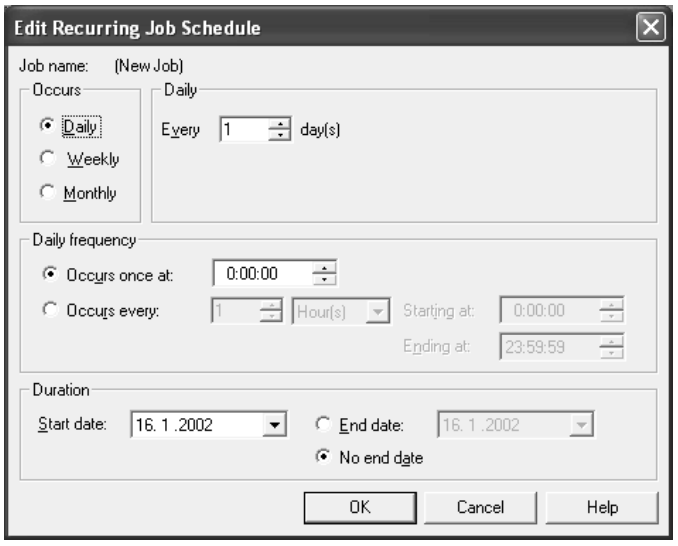
Po tejto nenáročnej transformácii, o ktorej ešte vlastne nevieme ako dopadne sa blížime do finále. Ďalší krok sa bude zdať čitateľom, ktorí nepracujú s veľkým množstvom údajov zbytočný, veď databáza FoodMart sa na priemernom notebooku s procesorom PIII taktovaným na 1000 MHz a 384 MB RAM pretransformuje za necelú minútu. Prevádzkovatelia veľkých databáz však možnosť naplánovať DTS úlohu na určený čas ocenia, pretože transformácie, importy a exporty sa môžu odložiť napríklad na určený čas v noci, kedy je málo iných transakcií a nevykonávajú sa ani analýzy. Príklad časového rozvrhu naplánovania úloh na noc môže byť napríklad nasledovný:

22:00 - 01:00 etapa ETL, t.j zavedenie údajov do dátového skladu
02:00 - 06:00 analytické výpočty kociek tvorba reportov...

Pri takomto naplánovaní úloh majú na ďalší deň manažéri keď prídu do práce ihneď k dispozícii výsledky analýz a reporty pre podporu rozhodovania.

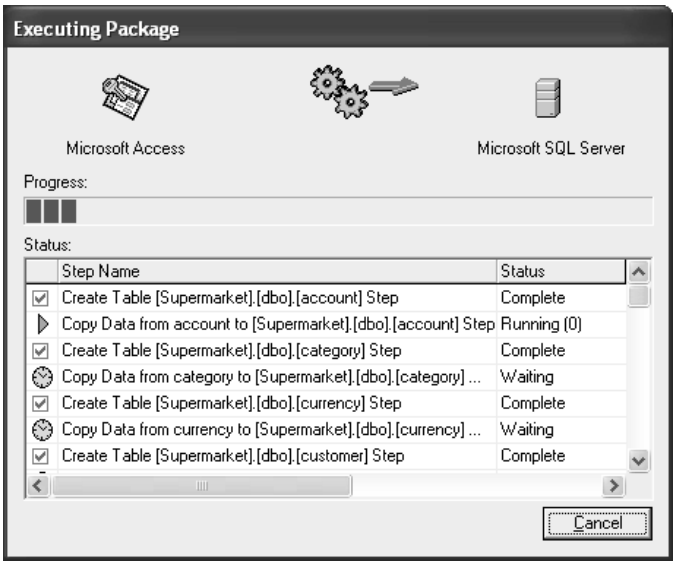
Obrázok 2.13 - plánovanie DTS úloh.





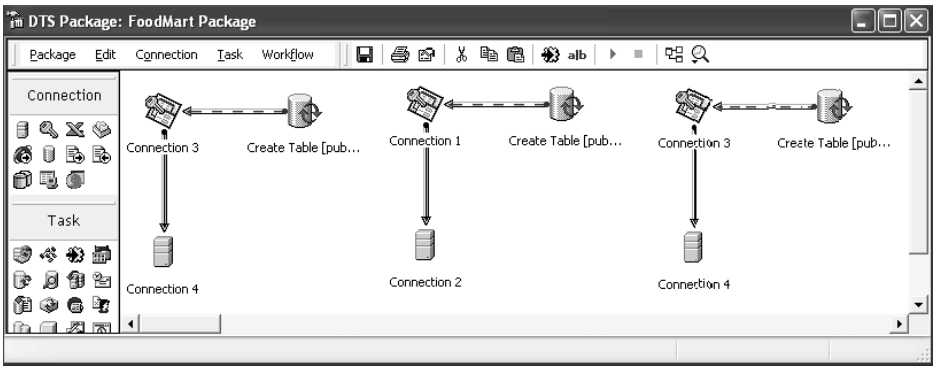
Obrázok 2.14 - dialóg pre plánovanie jednorázových alebo periodických DTS úloh.

Ak sa rozhodneme naplánované akcie odštartovať ihneď, sme o priebehu ich vykonávania informovaní pomocou prehľadného dialógu:



Obrázok 2.15 - informácia o behu DTS úlohy.

Odporúčame v dialógovom okne **„Save, schedule and replicate package“** zaškrtnúť voľbu **„Save DTS Package“**, a tým povoliť uloženie balíčka pod nejakým menom. Neskôr sa pomocou aplikácie Enterprise Manager podívať na sústavu workflow diagramov, ktoré DTS Import / Export Wizard vygeneruje.

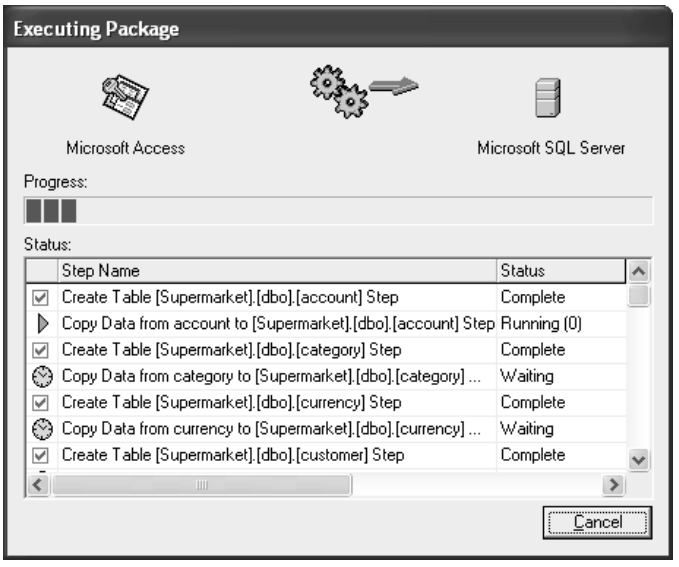


Obrázok 2.16 - workflow diagram vygenerovaný pomocou DTS Import / Export Wizard.

Jednoduchý príklad ETL z reálnej praxe.

Zdrojom údajov pre druhý príklad budú reálne záznamy z prevádzky telefónnej ústredne (konkrétne údaje sú v niektorých výpisoch vymaskované z dôvodu zachovania telekomunikačného tajomstva). Telefónna ústredňa zaznamenáva do databázy údaje o každom uskutočnenom hovore a aj niektoré diagnostické údaje

Mesačný záznam hovorov bol uložený v 170 megabajtovom súbore MDB súbor (Microsoft Access) a obsahoval vyše 1 600 000 záznamov o uskutočnených telefónnych hovoroch. Štruktúra návrhu pôvodnej databázy je jasná z návrhového zobrazenia



Obrázok 2.17 - štruktúra tabuľky hovorov.

Aby sme si urobili ešte lepšiu predstavu o údajoch v databáze, uvedieme zopár ilustračných záznamov

Cislo volajuceho	Skupina	Cas zaciatku	Doba trvania	Pocet impulzov	Volane cislo	Suma	Tarif impulzov	Prihlasenie
7-xx291328	2	14.5.1999 11:49:40	26	0	08387628xxx	0	-2	0
7-xx293964	1	14.5.1999 11:49:40	50	1	65422xxx	2,1	1	21
7-xx294430	2	14.5.1999 11:49:40	56	4	08264211xxx	8,4	3	27
7-xx337191	1	14.5.1999 11:49:41	13	1	58683xxx	2,1	1	24
7-xx339909	2	14.5.1999 11:49:42	114	6	0842224xxx	12,6	5	20
7-xx335242	1	14.5.1999 11:49:43	59	1	396xxx	2,1	1	23

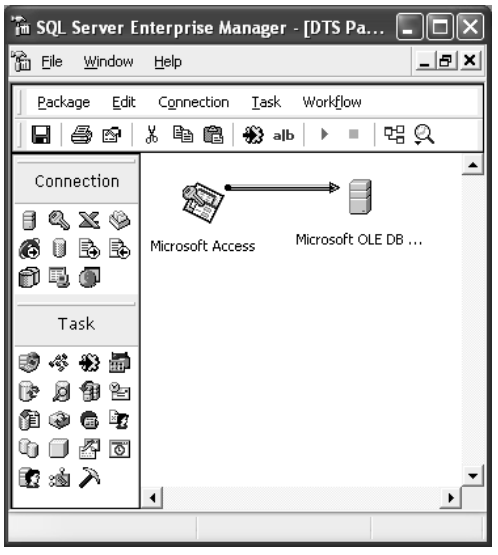
Zvláštne vysvetlenie si zaslúži len stĺpec **Skupina**, ktorý obsahuje čísla 1 až 5. Tento údaj zaraďuje uskutočnený telefónny hovor do jednej z nasledujúcich skupín:

1. Miestne hovory
2. Medzimesto
3. Hovor na mobilný telefón
4. Hovor na audiotexovú službu
5. Medzinárodný hovor

Predpokladajme, že údaje budú pravidelne napríklad mesačne z MDB súborov do databázy pod správou SQL Serveru 2000 s názvom Analýza. Extrakcia a transformácia bola v tomto prípade pomerne jednoduchá a spočívala hlavne vo vynechaní niektorých stĺpcov, ktoré neniesli podstatné informácie, alebo sa dali odvodiť z iných stĺpcov.

V tomto príklade, keďže predpokladáme pravidelné periodické zavádzanie údajov vytvoríme objekt **DTS Package**. Za cenu mierne zvýšeného úsilia pri vývoji si doprajeme aj oveľa vyššiu používateľskú bezpečnosť. Veď čo v prípade že počas transformácie niečo zlyhá?

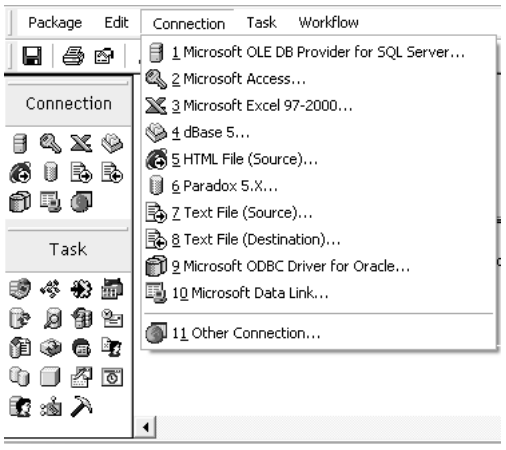
Použijeme aplikáciu **SQL Server Enterprise Manager**. (Windows menu: *Štart - MS SQL Server - Enterprise Manager*). Rozvinieme zložku **Data Transformation Services** a kliknutím ľavého tlačidla na položku **Local Packages** vytvoríme nový „balíček“.



Obrázok 2.18 - používateľské rozhranie pre vytváranie DTS Packages

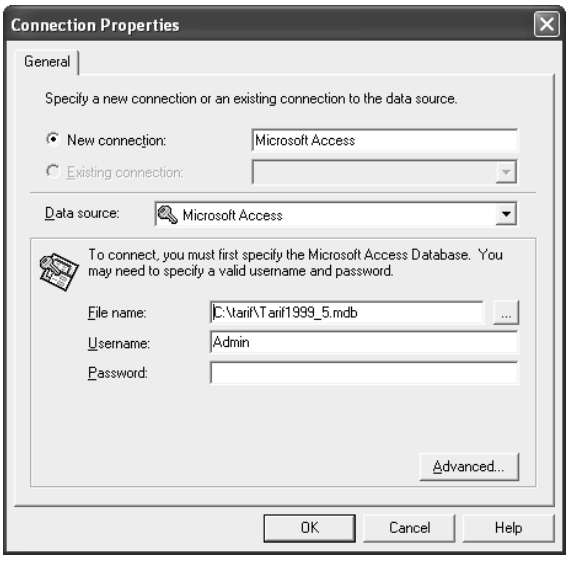
Používateľské rozhranie pre vytváranie DTS Packages je z používateľského hľadiska veľmi dobre vyriešené. Vývoj a implementácia transformačného algoritmu spočíva v podstate vo vytvorení akéhosi vývojového diagramu (workflow) pre návrh logického postupu jednotlivých krokov na ktoré môžeme proces ETL rozčleniť.

Najskôr musíme špecifikovať zdroj údajov, v našom prípade databázu Access. Význam ikon v zložke **Connection** nám bude jasný po rozvinutí menu s rovnakým názvom.



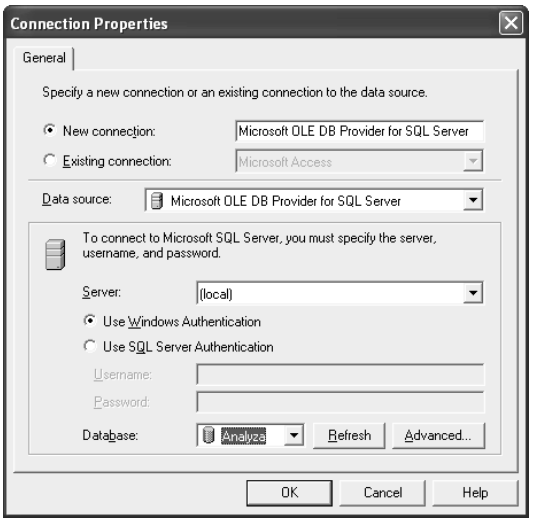
Obrázok 2.19 -význam ikon v zložke Connection

Zo zložky **Connection** v ľavom hornom okne, teda vyberieme ikonu Microsoft Access a v dialógovom okne zadáme umiestnenie konkrétneho zdroja údajov, v našom prípade to bol súbor ***C:\Tarif\tarif1999_5.mdb***



Obrázok 2.20 - pripojenie sa k zdroju údajov

V ďalšom logickom kroku špecifikujeme cieľovú databázu pre transformované údaje, v našom prípade to bude novovytvorená databáza Analýza pod správou MS SQL Servera 2000. V reálnej aplikácii by sme museli zadať aj parametre pre prístup k údajom. V našom prípade sme to ponechali na autentifikáciu operačného systému Windows.

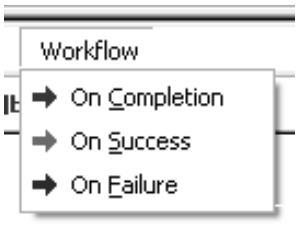


Obrázok 2.21 - výber destinácie

Touto špecifikáciou zdroja a cieľa sme sa dostali do etapy vytvárania workflow diagramu. Niekedy musíme začať tým, že nemáme v cieľovej databáze vytvorenú tabuľku kam by sme pretransformované a očistené údaje uložili. Preto v takom prípade napíšeme príkaz pre jej vytvorenie:

```
CREATE TABLE tarif
(
  volajuci VARCHAR(25),
  volany VARCHAR(25),
  skupina INT,
  zaciatok DATETIME,
  trvanie INT,
  cena INT
)
```

ktorý vložíme do workflow diagramu ako spustiteľný SQL Task. Na tomto mieste je potrebné vysvetliť, ako jednotlivé bloky kódu budú na seba naväzovať. Máme v zásade tri možnosti, ktoré uvidíme pri rozvinutí menu Workflow.



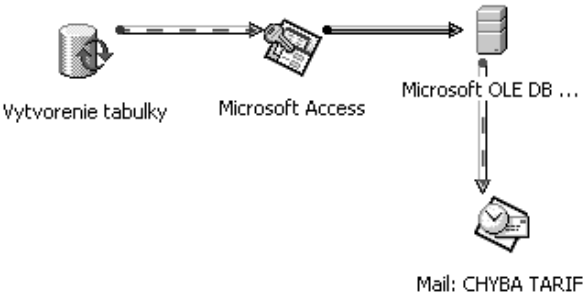
Obrázok 2.22 - menu Workflow

- **On Completion** - ďalší blok bude odštartovaný až po ukončení predchádzajúceho, pričom nezáleží na úspešnosti vykonania predchádzajúceho bloku

- **On Success** - zelená farba nám naznačí, že ďalší (závislý) blok sa bude realizovať len v prípade, ak predchádzajúci blok skončil úspešne

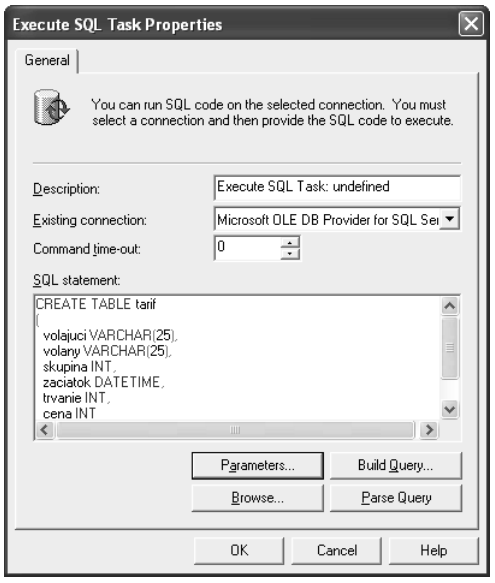
- **On Failure** - závislý blok sa začne vykonávať len vtedy, ak predchádzajúci blok skončil chybou. Táto konštrukcia sa preto bude využívať pre ošetrenie chybových stavov, napríklad na privolanie obsluhy cez mail, SMS pager a podobne.

Trochu predbehneme a ukážeme ako by v takom prípade vyzeral kompletný diagram cvičného príkladu:



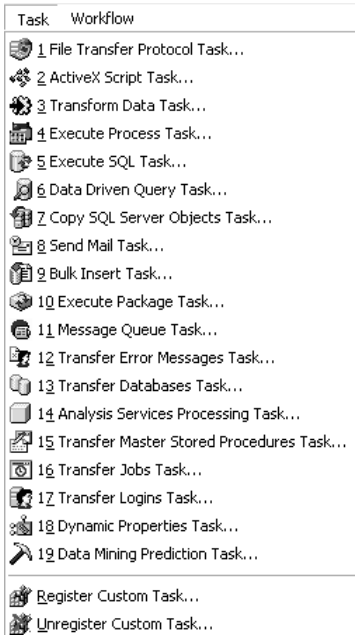
Obrázok 2.23 - diagram cvičného príkladu

Samozrejme v reálnej aplikácii by sme museli najskôr otestovať, či už takáto tabuľka v databáze neexistuje a podobne.



Obrázok 2.24 -SQL task pre vytvorenie tabuľky v cieľovej databáze

V našom cvičnom príklade z dôvodu jednoduchosti a prehľadnosti nebudeme blok pre vytvorenie tabuľky do diagramu transformácie zaraďovať. Tabuľku vytvoríme len raz, v návrhovej etape pri špecifikácii cieľa transformácie.



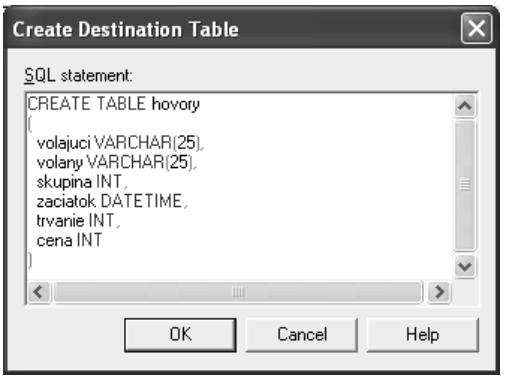
V nasledujúcom kroku definujeme úlohu **Data Transform Pack** zo zložky **Tasks** z ľavého dolného okna. Po zatlačení príslušnej ikony spojíme zdroj údajov z cieľovou destináciou (v tomto poradí). V okne DTS Package máme vytvorený „workflow“ diagram úlohy ktorú pripravujeme. Význam ikon v zložke Tasks nám bude jasný po rozvinutí menu s rovnakým názvom.

Obrázok 2.25 - význam ikon v zložke Tasks

Zostáva nám nastaviť parametre transformácie. Po kliknutí pravým tlačidlom na grafický symbol transformácie (úsečka so sípkou) môžeme v jednotlivých záložkách dialógového okna **Transform Data Task Properties** nastaviť parametre pre transformáciu.

V záložke **Source** nastavíme, ktorá tabuľka, alebo kombinácia tabuliek bude predmetom transformácie. V našom prípade to bude tabuľka **tarif**.

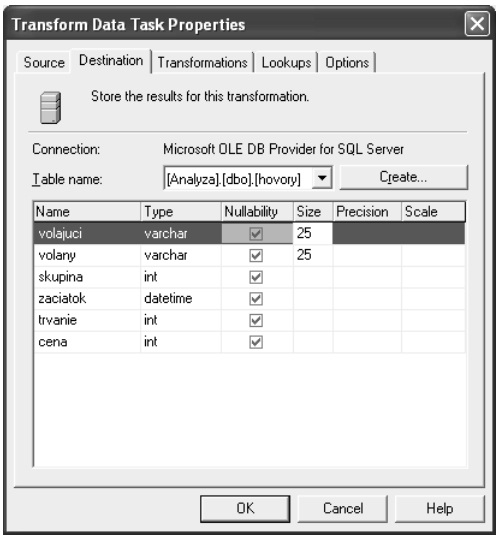
*Obrázok 2.26 -Transform Dtata Tasks Properties - záložka **Source***



Ďalším krokom bude stanovenie tabuľky v cieľovej databáze. V našom prípade tabuľku **hovory** najskôr v tejto etape vytvoríme. Príkaz pre vytvorenie tabuľky zadáme v dialógovom okne, ktoré sa otvorí po zatlačení tlačidla **Create** v záložke **Destination**.

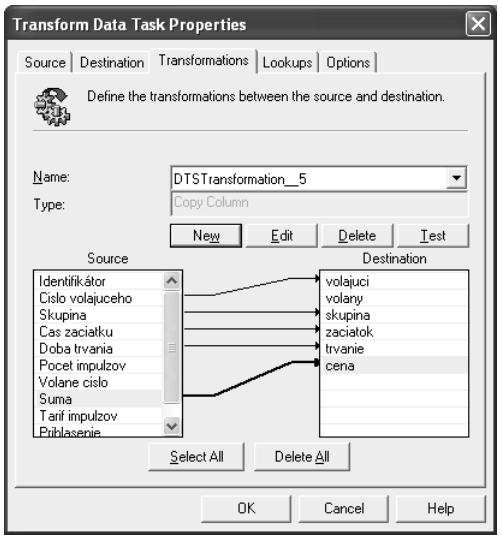
Obrázok 2.27 - vytvorenie tabuľky hovory v cieľovej databáze.

V záložke **Destination** vidíme názvy a typy všetkých stĺpcov tabuľky, ktorú sme vybrali ako cieľovú.



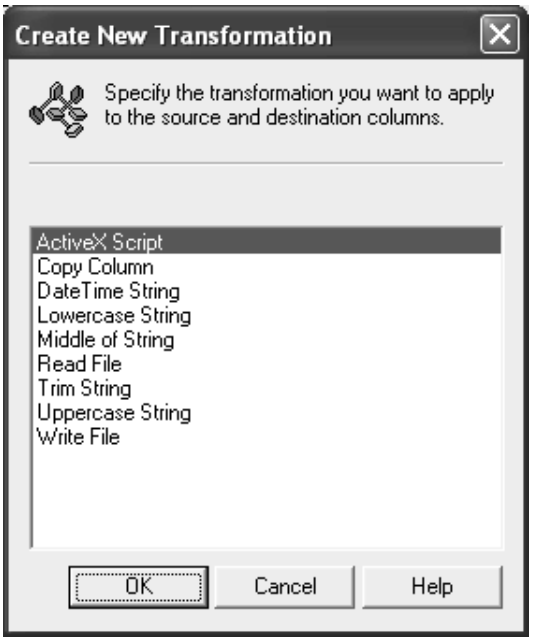
*Obrázok 2.28 -Transform Dtata Tasks Properties - záložka **Destination***

V záložke **Transformations** definujeme metódou drag and drop pravidlá pre transformáciu údajov medzi jednotlivými stĺpcami zdrojovej a cieľovej tabuľky.



*Obrázok 2.29 -Transform Dtata Tasks Properties - záložka **Transformations***

Možnosti transformácií sú také široké, že značne prevyšujú rozsah tejto publikácie. Preto len pre ilustráciu uvedieme zoznam možností, ktoré máme k dispozícii prostredníctvom dialógu **Create New Transformation**.



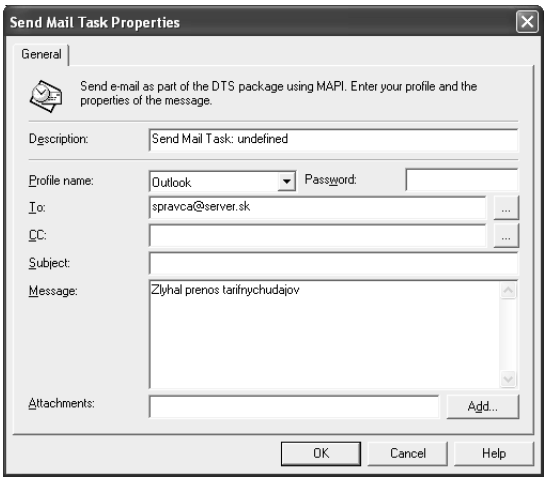
Obrázok 2.30 -dialóg pre vytvorenie novej transformácie.

My sme pre každú transformáciu zvolili tú najjednoduchšiu možnosť **Copy Column** pre jednoduché prekopírovanie údajov zo stĺpca zdrojovej tabuľky do stĺpca cieľovej tabuľky.

Každú transformáciu si môžeme zvlášť ako sa hovorí „nanečisto“ vyskúšať.

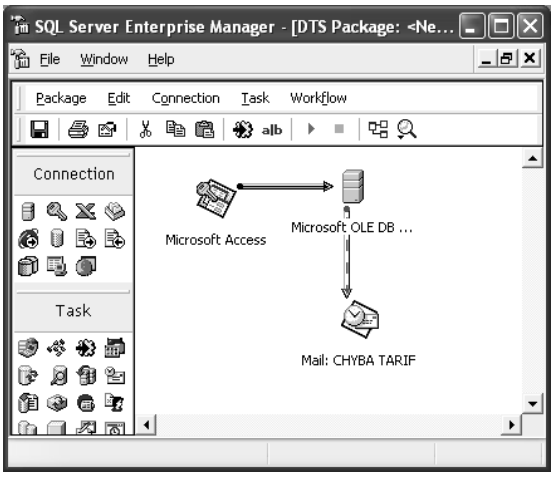
V poslednej záložke **Options** by sme mohli nadefinovať napríklad formát súboru v ktorom sa zobrazia výnimky a podobne.

Tým máme základ „balíčka“ pre transformáciu údajov vytvorený a dokonca sme si aj jednotlivo otestovali všetky transformácie. Vývojár softvéru však musí počítať vždy s tým, že sa nachádza v oblasti platnosti Murphyho zákonov, a teda čo sa môže pokaziť sa spravidla aj pokazí. Mali by sme teda ošetriť prípadnú chybu, v našom prípade by sme napríklad mohli poslať administrátorovi email s chybovým hlásením.



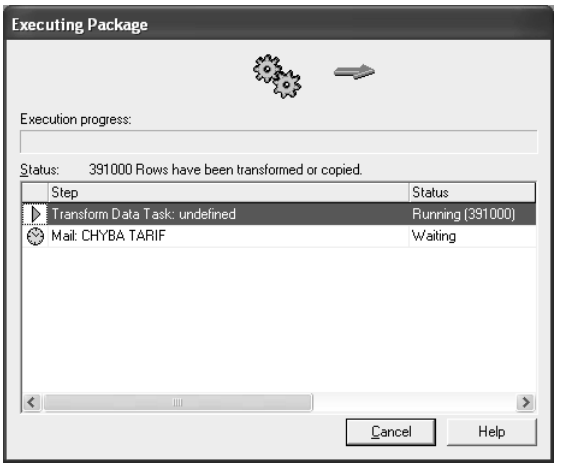
Obrázok 2.31 - definovanie mailu, ktorý sa odošle v prípade výskytu chyby.

Finálny návrh transformácie bude potom vypadať nasledovne:



Obrázok 2.32. -kompletný workflow diagram cvičného súboru.

Stačí teda pripraviť údaje pre prvé testovacie spustenie DTS Package, ktoré vykonáme samozrejme pod dozorom. O priebehu celej akcie sme samozrejme podrobne informovaní:



Obrázok 2.33 -informácie o priebehu transformácie.

Proces ETL v tomto jednoduchom prípade pre takmer dva milióny záznamov trval necelé tri minúty (konfigurácia počítača: Pentium III 1000MHz, 512 MB RAM):

Záver

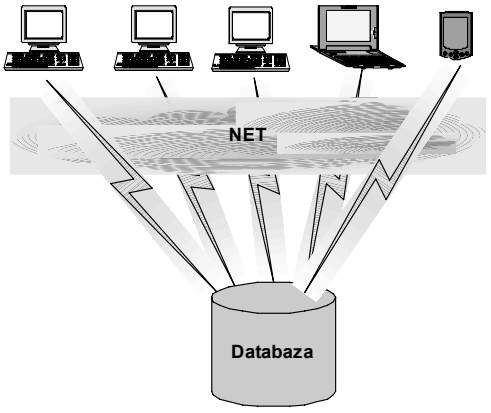
Rozhodne odporúčame vyskúšať **Microsoft Data transformation Services** aj tým, ktorí primárne používajú iné databázové platformy, čo sa týka používateľského komfortu a možností počnúc definovaním jednoduchých transformácií cez možnosti naprogramovať zložité algoritmy transformácie, väzby na OLE a iné črty operačného systému Windows. Kto by sa domnieval, že tu možnosti DTS končia, veľmi by sa mýlil. Tu končia len priestorové možnosti kapitoly tejto publikácie venovanej DTS. Nespomenuli sme totiž napríklad **Object Package**, ani úlohy typu **Dynamic Properties**. Veľké možnosti v procese ETL nám poskytuje aj **viacfázová dátová pumpa** a podobne

OLAP analýza údajov v databázach

V predchádzajúcej kapitole sme už stručne naznačili základné rozdiely medzi systémami typu **OLTP** a **OLAP**. Pochopenie teoretického rozdielu nie je pre aplikácie typu BI/DW až také dôležité. Oveľa dôležitejšie je správne navrhnúť štruktúru databázu pre danú oblasť použitia.

OLTP (On-line Transaction Processing)

Oblasť použitia transakčných databáz je takmer neobmedzená. Primárnym cieľom pri ich návrhu je umožniť klientom databázového servera vykonávanie veľkého množstva on - line transakcií, napríklad bankových, obchodných a podobne.

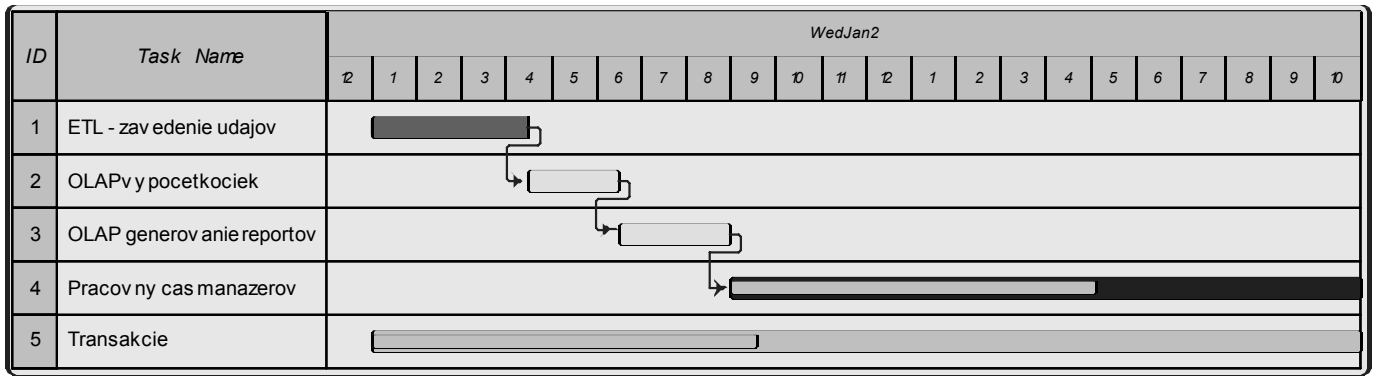


Obrázok 3.1 - OLTP

K zdroju údajov teda v rovnakom čase pristupuje veľké množstvo používateľov, ktorí údaje z databázy čítajú, iní do nej zapisujú, prípadne niektorí vykonávajú aj jednoduchšie analýzy. Áno, aj nad OLTP databázami je možné vybudovať OLAP analytické aplikácie, ale ako sa ľudovo povie: „Nie je to pravé orechové“. Nielen z teórie, ale aj z praktických skúseností totiž vyplýva, že údaje v transakčných databázach by mali byť uložené v normalizovaných tabuľkách, ktoré by mali vyhovovať podmienkam druhej, alebo tretej normálnej formy. To znamená veľa atomických, relačne zviazaných tabuliek. Analýza veľkého množstva takto uložených údajov by bola preto veľmi neefektívna a značne pomalá. Taktiež návrh analytickej aplikácie, hlavne tabuliek dimenzií nad OLTP databázou nie je práve jednoduchá záležitosť.

OLAP (On-line Analytical Processing)

Typické využitie systémov OLAP je pre analýzu veľkého množstva údajov. Výsledkom analýzy sú súhrny a reporty, ktoré slúžia manažérom ako podklady pre ich rozhodnutia, či už v oblasti riadenia firmy, riadenia ekonomických a technologických procesov a podobne. Pre výpočet OLAP kociek je potrebné vykonať veľké množstvo výpočtov a agregácií a to takmer v reálnom čase. Typickým príkladom pre výraz „takmer v reálnom čase“ je nasledovná časová tabuľka, ktorá zobrazuje časovú následnosť procesov analýzy. **Pozor, údaje v prvých troch riadkov sa týkajú predchádzajúceho dňa** to znamená že v našom prípade sa po polnoci vždy analyzujú údaje z predchádzajúceho dňa.



Obrázok 3.2 - časová náväznosť procesu OLAP rámci jedného dňa.

Pri slovnom popise celého procesu znázorneného v časovej tabuľke bude lepšie postupovať zdola nahor.

Transakcie

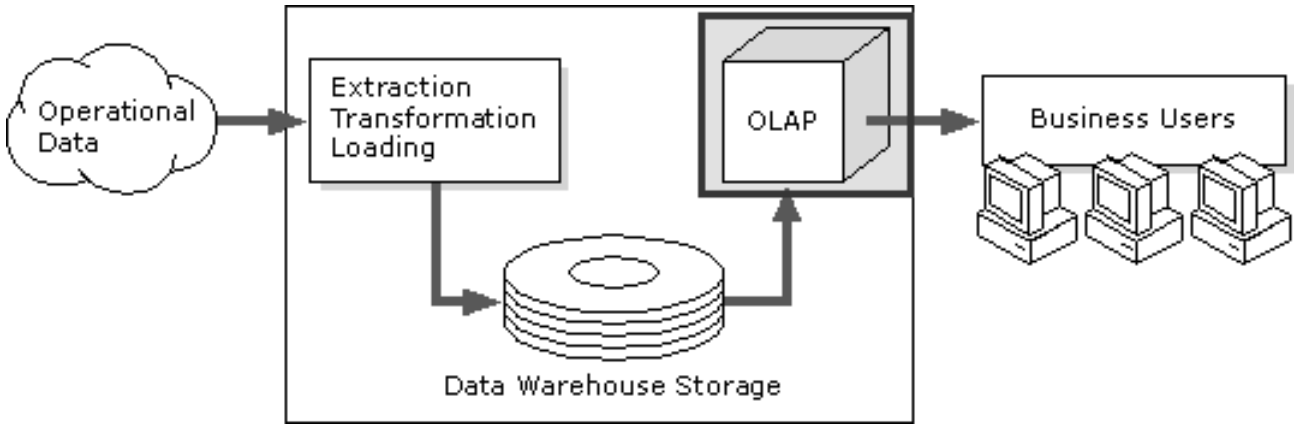
V poslednom riadku (5) vidíme, že transakcie dnešného dňa bežia nepretržite 24 hodín. Môže sa jednať o výrobný proces, elektronický obchod, bankové transakcie, proste OLTP proces ktorý je zdrojom údajov pre analýzy. Tieto analýzy dnešných údajov sa budú vykonávať až po polnoci, aby zajtra na začiatku pracovnej doby boli k dispozícii Ak by sme sa rozhodli využiť OLTP systém aj pre OLAP analýzy v čase od 00:00 do 09:00 budú bežať obidva procesy súbežne, t. j. dnešné transakcie a analýza včerajších údajov.

Pracovný čas manažérov

Začiatok pracovného času manažérov (v našej tabuľke 09:00) je rozhodujúcim okamihom, kedy by mali byť výsledky analýzy a príslušné reporty týmto manažérom k dispozícii. Na našom grafe sme dokonca znázornili fakt, že pracovná doba manažérov obvykle nekončí obvyklým firemným „padla“ o 16:00, to preto že práve takýto pracovný čas je pre mnohých manažérov charakteristický.

Proces prípravy údajov a analýza

Aby mohli manažéri získať v stanovenom čase kvalitné podklady pre podporu rozhodovania, musí prebehnúť príprava údajov pre analýzu, v prípade warehousingu ich zavedenie do dátového skladu a ich následná analýza. V našom príklade proces ETL *(riadok 1 časovej tabuľky)* začne ihneď po polnoci o 00:00. Po zhruba troch hodinách je táto etapa ukončená a môže začať napočítavanie kociek a ostatné agregačné a analytické výpočty *(riadok 2 časovej tabuľky)*. Na tento proces nadviaže generovanie zostáv, reportov a podobne *(riadok 3 časovej tabuľky)*.



Obrázok 3.3 - zaradenie OLAP v blokovej schéme dátového skladu

Termín **OLAP** zaviedol Dr. E . F. Codd na popísanie technológie, ktorá by pomohla preklenúť medzery medzi využitím osobných počítačov a riadením podnikových dát. Pre OLAP existuje viacero rôznych definícií, napríklad:

OLAP je voľne definovaný rad princípov, ktoré poskytujú dimenzionálny rámec pre podporu rozhodovania.

Pojem OLAP sa pomerne často zamieňa s iným pojmom **DSS** (Decision Support Systems) - systémy na podporu rozhodovania. Tieto systémy umožňujú pracovníkom prijímajúcim rozhodnutia prístup k údajom potrebným na „tvorbu“ takýchto rozhodnutí.

Okrem jednej z definícií pojmu OLAP je pomerne známe aj tzv. **„dvanástoro“ pôvodných pravidiel OLAP od Dr. E . F. Codda.**

Pravidlo prvé: **Multidimenzionálny konceptuálny pohľad:** OLAP by mal poskytovať používateľovi multidimenzionálny model zodpovedajúci jeho podnikateľským potrebám tak aby tento model mohol využívať pre analýzu zhromaždených údajov.

Pravidlo druhé: **Transparentnosť:** Technológia systému OLAP, podriadená databáza a architektúra výpočtov by malí byť pre používateľa transparentné, aby tento mohol naplno využívať svoju produktivitu a odbornosť pri použití front - end nástrojov a prostredia. Pre platformu Microsoft môžeme ako klientský nástroj použiť napríklad program Excel z balíka Office, ktorý bude napojený na OLAP Server. Dôležitá je samozrejme aj heterogénnosť vstupných dát, ktorú zaistíme v procese ETL.

Pravidlo tretie: **Dostupnosť:** Systém OLAP by mal pristupovať len k tým údajom, ktoré sú potrebné na analýzu. Systém by mal byť navyše schopný pristupovať k všetkým údajom potrebným pre analýzu, nezávisle na tom, z ktorého heterogénneho podnikového zdroja tieto údaje pochádzajú, ako často sú obnovované a podobne. Znovu zdôrazňujeme význam etapy ETL, v ktorej sa údaje očistia, zahustia a pretransformujú.

Pravidlo štvrté: **Konzistentné vykazovanie:** Aj keď počet záznamov a teda aj veľkosť databáz postupom času rastie, používateľ by nemal pocítiť žiadne podstatné zníženie výkonu.

Pravidlo piate: **Architektúra klient - server:** Systém OLAP musí zodpovedať princípom architektúry klient - server s prihliadnutím na maximálnu cenu a výkon, flexibilitu a interoperatibilitu.

Pravidlo šieste: **Generická dimenzionalita:** Každá dimenzia údajov musí byt ekvivalentná v štruktúre aj operačných schopnostiach.

Pravidlo siedme: **Dynamické ošetrenie riedkych matíc:** Systém OLAP by mal byt schopný adaptovať svoju fyzickú schému na konkrétny analytický model, ktorý optimalizuje ošetrenie riedkych matíc, pričom dosiahne a udrží požadovanú úroveň výkonu.

Pravidlo ôsme: **Podpora pre viacerých používateľov:** Systém OLAP musí byť schopný podporovať pracovnú skupinu používateľov pracujúcich súčasne na konkrétnom modeli.

Pravidlo deviate: **Neobmedzené krížové dimenzionálne operácie:** Systém OLAP musí dokázať rozoznať dimenzionálne hierarchie a automaticky vykonať asociované kumulované kalkulácie v rámci dimenzií a aj medzi nimi.

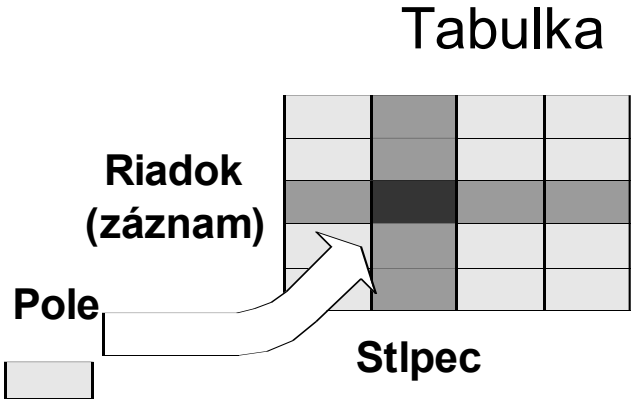
Pravidlo desiate: **Intuitívna manipulácia s údajmi:** Pravidlo definuje konsolidované preorientovanie ciest na detailnú úroveň a späť (drill down, drill up). Používateľské rozhranie by malo umožňovať všetky manipulácie spôsobom „ukázať a kliknúť, prípadne zachytiť a premiestniť“ v bunkách kocky.

Pravidlo jedenáste: **Flexibilné vykazovanie:** Musí existovať schopnosť usporiadať riadky, stĺpce a bunky spôsobom, ktorý umožní analýzu intuitívnou vizuálnou prezentáciu analytických zostáv.

Pravidlo dvanáste: **Neobmedzené dimenzie a úrovne agregácie:** V závislosti od požiadaviek podnikania môže mať analytický model viacero dimenzií, pričom každá z nich môže mať viacnásobné hierarchie. Systém OLAP by nemal zavádzať (napríklad z technických dôvodov) žiadne umelé obmedzenia počtu dimenzií alebo úrovní agregácie.

Prechod od relačného k multidimenzionálnemu databázovému modelu.

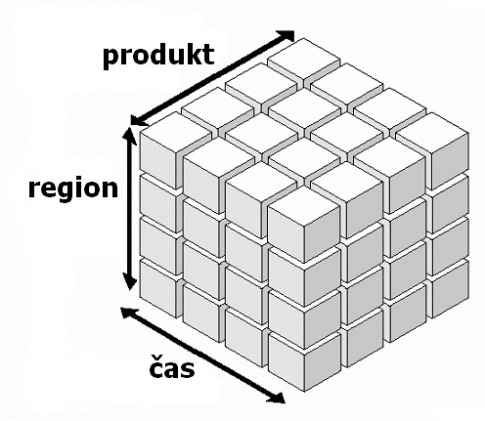
Prevažná väčšina údajov je organizovaná v relačnej databáze v dvojrozmerných relačných tabuľkách. Každý riadok takejto tabuľky sa vzťahuje k nejakému predmetu, udalosti, alebo k ich časti. Výsledkom analýzy údajov býva obvykle multidimenzionálna dátová štruktúra - kocka. Zjednodušene by sa dalo povedať, že kocka je v multidimenzionálnom dátovom modeli akýmsi ekvivalentom tabuľky v relačnej databáze.



Obrázok 3.4 - dvojrozmerná databázová tabuľka

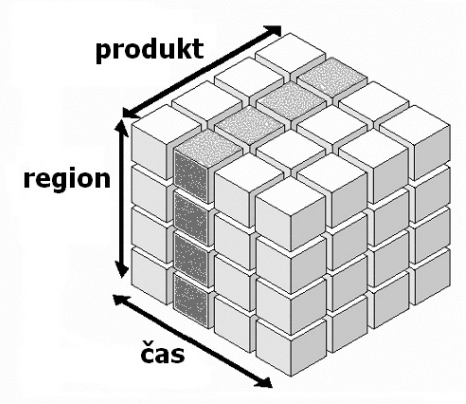
Multidimenzionálny databázový model si môžeme najjednoduchšie predstaviť ako priestorovú kocku. Každá kocka má niekoľko dimenzií. Na rozdiel od geometrickej kocky môže mať multidimenzionálny databázový model aj viac dimenzií ako tri. MS SQL Server 2000 umožňuje použitie až 64 dimenzií. Príkladom typického trojdimenzionálneho modelu môže byť kocka s dimenziami:

- čas
- región
- produkt

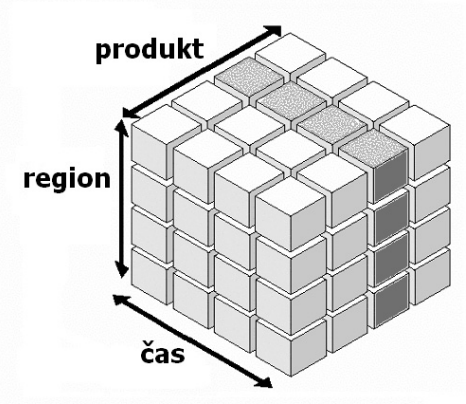


Obrázok 3.5 - princíp multidimenzionálnej kocky

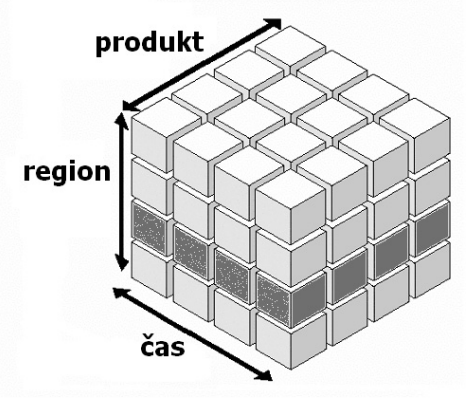
Údaje sa nachádzajú v prienikoch jednotlivých dimenzií. Môžeme analyzovať údaje len za určité časové obdobie, napríklad aby sme vyhodnotili výsledky reklamnej kampane, alebo sledovanosť webovej stránky za určité obdobie a podobne. Iným príkladom môžu byť údaje z určitého regiónu, ku ktorým má prístup regionálny riaditeľ, pre potreby jeho rozhodovania. Prípadne marketingové údaje pre jednotlivé produkty, alebo skupiny produktov môžu byť k dispozícii pre produktových manažérov.



Obrázok 3.6 - analýza údajov za určité časové obdobie



Obrázok 3.7 - analýza údajov pre určitú skupinu produktov



Obrázok 3.8 - analýza údajov za pre určitý región

Údaje v kocke sú po jej vytvorení a pravidelných aktualizáciách okamžite k dispozícii pre rôzne analýzy.

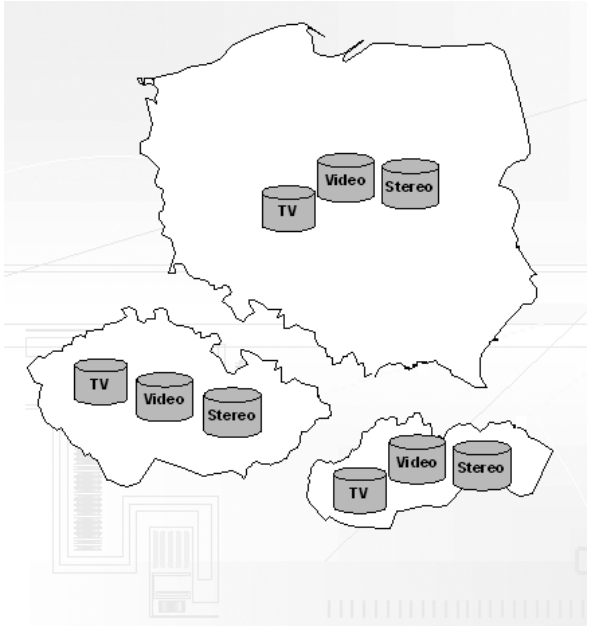
Pre vytvorenie kocky nám MS SQL Server 2000 poskytuje v zásade dve možnosti:

– naprogramovať všetko potrebné pre vytvorenie kocky v jazyku T-SQL
– využiť možnosti Sprievodcu vytvorením kocky

Ukážeme si obidve možnosti, pričom prvá možnosť, vytvorenie kocky pomocou príkazov v jazyku T-SQL poslúži viac pre pochopenie princípov, než pre praktické využitie.

Klauzula CUBE (aplikácia v jazyku T-SQL).

Pomocou klauzule CUBE získame multidimenzionálny prehľad všetkých možných kombinácií podľa vybraných dimenzií. Môžeme takto vykonať analýzu a sumarizáciu údajov v tabuľke súčastne podľa viacerých kritérií. Praktické použitie pre vytvorenie multidimenzionálnej kocky si najlepšie ukážeme na príklade fiktívnej americkej firmy, ktorá predáva široký sortiment tovaru (vybrali sme tri tovarové skupiny: TV, video, stereo) zásielkovou formou si vytvorí pobočku pre Strednú Európu. O tomto obchodnom priestore zatiaľ veľa informácií nemá a tak si ako svoje prvotné sídlo vyberie nejaké pekné historické mesto v Strednej Európe, napríklad Prahu. Vo všetkých okolitých krajinách (Čechy, Slovensko a Poľsko) rozvinie rovnakú štartovaciu reklamnú kampaň a už jej len zostáva čakať, aká bude v spomínaných krajinách reakcia zákazníkov.

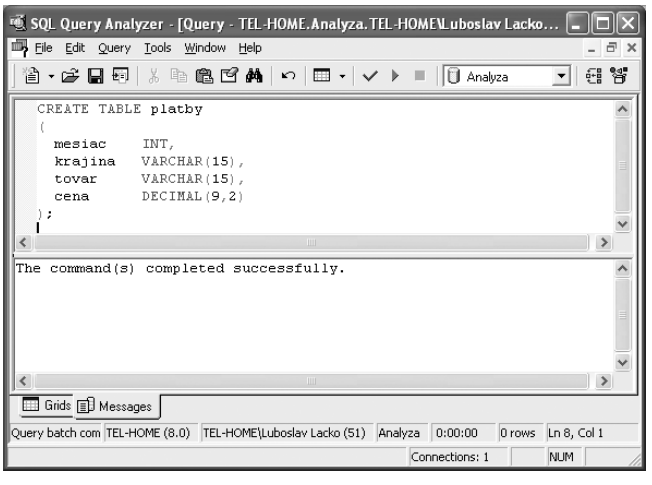


Obrázok 3.9 - námet príkladu pre klauzulu CUBE

Každá stredná alebo veľká firma má informačný systém, ktorý využíva informácie z mnohých relačne zviazaných tabuliek. Skutočne je však dôležitá len jedna tabuľka, a síce tabuľka, ktorá obsahuje údaje o predanom tovare a platbách zákazníkov Neskôr ju budeme nazývať **Tabuľka faktov**. Ak by táto tabuľka zívala prázdnotou, darmo by PR (Public Relation) manažér vysvetľoval marketingovému manažérovi, že povedomie o značke firmy v danej krajine stúplo sedem a pol krát, firma by nemala tržby a vo veľmi krátkej dobe by zanikla.

Tento jednoduchý príklad si môžeme precvičiť celý, príkazy v jazyku T-SQL budeme zadávať pomocou konzolovej aplikácie **SQL Query Analyzer**.

Obrázok 3.10 - námet príkladu pre klauzulu CUBE



Majme tabuľku platieb zákazníkov za dodaný tovar za tri mesiace s nasledovnou štruktúrou:

```
CREATE TABLE platby
(
  mesiac     INT,
  krajina    VARCHAR(15),
  tovar      VARCHAR(15),
  cena       DECIMAL(9,2)
   );
```

Tabuľka je samozrejme maximálne zjednodušená, reálna tabuľka by obsahovala oveľa viac stĺpcov, ktoré by boli relačne zviazané s inými tabuľkami. Tržby budeme simulovať pomocou funkcie **RAND**, ktorá vracia hodnotu typu float v rozsahu 0 - 1.

Za prvé tri mesiace simulácie činnosti našej fiktívnej firmy obsahuje tabuľka približne štvrť milióna záznamov.

```
DECLARE @N int
SET @N=10000
WHILE (@N >0)
BEGIN
INSERT INTO platby VALUES (1, 'Cechy', 'TV', RAND(@N)*2000);
INSERT INTO platby VALUES (1, 'Cechy', 'video', RAND(@N)*1200);
INSERT INTO platby VALUES (1, 'Cechy', 'stereo', RAND(@N)*800);

INSERT INTO platby VALUES (1, 'Slovensko', 'TV', RAND(@N)*1000);
INSERT INTO platby VALUES (1, 'Slovensko', 'video', RAND(@N)*600);
INSERT INTO platby VALUES (1, 'Slovensko', 'stereo', RAND(@N)*400);

INSERT INTO platby VALUES (1, 'Polsko', 'TV', RAND(@N)*6000);
INSERT INTO platby VALUES (1, 'Polsko', 'video', RAND(@N)*4400);
INSERT INTO platby VALUES (1, 'Polsko', 'stereo', RAND(@N)*3200);

INSERT INTO platby VALUES (2, 'Cechy', 'TV', RAND(@N)*2000);
INSERT INTO platby VALUES (2, 'Cechy', 'video', RAND(@N)*1200);
INSERT INTO platby VALUES (2, 'Cechy', 'stereo', RAND(@N)*800);

INSERT INTO platby VALUES (2, 'Slovensko', 'TV', RAND(@N)*1000);
INSERT INTO platby VALUES (2, 'Slovensko', 'video', RAND(@N)*600);
INSERT INTO platby VALUES (2, 'Slovensko', 'stereo', RAND(@N)*400);

INSERT INTO platby VALUES (2, 'Polsko', 'TV', RAND(@N)*6000);
INSERT INTO platby VALUES (2, 'Polsko', 'video', RAND(@N)*4400);
INSERT INTO platby VALUES (2, 'Polsko', 'stereo', RAND(@N)*3200);

INSERT INTO platby VALUES (3, 'Cechy', 'TV', RAND(@N)*2000);
INSERT INTO platby VALUES (3, 'Cechy', 'video', RAND(@N)*1200);
INSERT INTO platby VALUES (3, 'Cechy', 'stereo', RAND(@N)*800);

INSERT INTO platby VALUES (3, 'Slovensko', 'TV', RAND(@N)*1000);
INSERT INTO platby VALUES (3, 'Slovensko', 'video', RAND(@N)*600);
INSERT INTO platby VALUES (3, 'Slovensko', 'stereo', RAND(@N)*400);

INSERT INTO platby VALUES (3, 'Polsko', 'TV', RAND(@N)*6000);
INSERT INTO platby VALUES (3, 'Polsko', 'video', RAND(@N)*4400);
INSERT INTO platby VALUES (3, 'Polsko', 'stereo', RAND(@N)*3200);

SET @N=@N-1
END
```

Niektoré informácie zistíme aj pomocou klasických SQL dotazov, napríklad počet záznamov,

```
SELECT COUNT(*) FROM platby
```

alebo celkovú inkasovanú sumu.

```
SELECT SUM(cena) FROM platby
```

Zistili sme, že v našej cvičnej tabuľke máme 270 000 záznamov a od zákazníkov sme utŕžili 474 367 549.62 obeživa. Na základe týchto informácií môžeme iba konštatovať, či našej firme takýto dosahovaný obrat postačuje, alebo nepostačuje. Nevieme však nič o napríklad o predaji jednotlivých druhov tovaru v jednotlivých krajinách a podobne. Preto sa pustíme do analýzy, v našom prípade si vytvoríme takzvanú kocku, to znamená multidimenzionálny prehľad všetkých možných kombinácií podľa vybraných dimenzií. Použijeme teda SQL príkaz obsahujúci klauzulu CUBE. Dimenzie „kocky“ budú v našom prípade mesiac, krajina a tovar.

```
SELECT mesiac, krajina, tovar,
SUM(cena) AS SUMA FROM platby
GROUP BY mesiac, krajina, tovar WITH CUBE;
```

Výsledkom je výpis všetkých kombinácií:

```
mesiac      krajina         tovar           SUMA
----------- --------------- --------------- -------------------------------------
1           Cechy           stereo          6453980.24
1           Cechy           TV              16134950.67
1           Cechy           video           9680970.48
1           Cechy                           32269901.39
1           Polsko          stereo          25815921.03
1           Polsko          TV              48404852.00
1           Polsko          video           35496891.45
1           Polsko                          109717664.48
1           Slovensko       stereo          3226990.16
1           Slovensko       TV              8067475.33
1           Slovensko       video           4840485.18
1           Slovensko                       16134950.67
1                                           158122516.54

2           Cechy           stereo          6453980.24
2           Cechy           TV              16134950.67
2           Cechy           video           9680970.48
2           Cechy                           32269901.39
2           Polsko          stereo          25815921.03
2           Polsko          TV              48404852.00
2           Polsko          video           35496891.45
2           Polsko                          109717664.48
2           Slovensko       stereo          3226990.16
2           Slovensko       TV              8067475.33
2           Slovensko       video           4840485.18
2           Slovensko                       16134950.67
2                                           158122516.54

3           Cechy           stereo          6453980.24
3           Cechy           TV              16134950.67
3           Cechy           video           9680970.48
3           Cechy                           32269901.39
3           Polsko          stereo          25815921.03
3           Polsko          TV              48404852.00
3           Polsko          video           35496891.45
3           Polsko                          109717664.48
3           Slovensko       stereo          3226990.16
```

```
3           Slovensko       TV              8067475.33
3           Slovensko       video           4840485.18
3           Slovensko                       16134950.67
3                                           158122516.54
                                            474367549.62

            Cechy           stereo          19361940.72
            Cechy           TV              48404852.01
            Cechy           video           29042911.44
            Cechy                           96809704.17
            Polsko          stereo          77447763.09
            Polsko          TV              145214556.00
            Polsko          video           106490674.35
            Polsko                          329152993.44
            Slovensko       stereo          9680970.48
            Slovensko       TV              24202425.99
            Slovensko       video           14521455.54
            Slovensko                       48404852.01
1                           stereo          35496891.43
2                           stereo          35496891.43
3                           stereo          35496891.43
                            stereo          106490674.29
1                           TV              72607278.00
2                           TV              72607278.00
3                           TV              72607278.00
                            TV              217821834.00
1                           video           50018347.11
2                           video           50018347.11
3                           video           50018347.11
                            video           150055041.33

(64 row(s) affected)
```

Ak manažéri ktorejkoľvek firmy dostanú k dispozícii takéto komplexné a trochu aj prehľadne usporiadané informácie, (samozrejme musia porozumieť ich štruktúre), môžu zodpovedne analyzovať danú situáciu a operatívne prijímať kvalifikované rozhodnutia.

Pre účely napríklad obchodných prezentácií je možné zobraziť tieto údaje v trochu zhustenom a prehľadnejšom formáte. Neskôr tomu budeme hovoriť „Riedka kocka“. Naša „kocka“ má dimenzie **mesiac**, **krajina** a **tovar** (ďalej len **MKT**). Pomocou funkcie **GROUPING** môžeme vytvoriť masku jednotlivých dimenzií.

```
SELECT mesiac, krajina, tovar,
SUM(cena) AS SUMA,
GROUPING (mesiac)AS M,
GROUPING (krajina)AS K,
GROUPING (tovar)AS T
FROM platby
GROUP BY mesiac, krajina, tovar WITH CUBE;
```

Výpis potom bude v tvare:

```
mesiac      krajina         tovar           SUMA            M    K    T
----------  --------------- --------------- --------------- ---- ---- ----
1           Cechy           stereo          6453980.24      0    0    0
1           Cechy           TV              16134950.67     0    0    0
1           Cechy           video           9680970.48      0    0    0
1           Cechy                           32269901.39     0    0    1
1           Polsko          stereo          25815921.03     0    0    0
1           Polsko          TV              48404852.00     0    0    0
1           Polsko          video           35496891.45     0    0    0
1           Polsko                          109717664.48    0    0    1
1           Slovensko       stereo          3226990.16      0    0    0
1           Slovensko       TV              8067475.33      0    0    0
1           Slovensko       video           4840485.18      0    0    0
1           Slovensko                       16134950.67     0    0    1
1                                           158122516.54    0    1    1
...
...
```

Všimnime si „masky“ jednotlivých dimenzií. Ak má množina atribútov **MKT** pre daný záznam hodnotu **(0,0,0)**, a chceme vytvoriť riedku kocku, ktorá bude obsahovať len súhrny za jednotlivé kombinácie dimenzií, môžeme takýto riadok vypustiť. Záznamy, ktorých masky obsahujú aspoň jednu jednotku predstavujú súhrny. Napríklad záznam s maskou **MKT (0,0,1)**

```
mesiac      krajina         tovar           SUMA            M    K    T
----------  --------------- --------------- --------------- ---- ---- ----
1           Cechy                           32269901.39     0    0    1
```

predstavuje súhrn všetkého predaného tovaru (stereo, TV aj video) v Čechách. Necháme vypísať teda len záznamy, kde maska **MKT** obsahuje aspoň jednu jednotku.

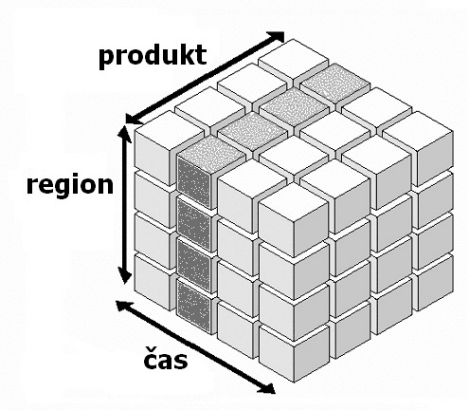
```
SELECT mesiac, krajina, tovar,
SUM(cena) AS SUMA
FROM platby
GROUP BY mesiac, krajina, tovar WITH CUBE
HAVING (GROUPING(mesiac)=1)OR(GROUPING (krajina)=1)OR(GROUPING (tovar)=1);
```

Výsledný výpis bude obsahovať v našom prípade už len 37 záznamov

```
mesiac      krajina         tovar           SUMA
----------- --------------- --------------- ---------------
1           Cechy                           32269901.39
1           Polsko                          109717664.48
1           Slovensko                       16134950.67
1                                           158122516.54
2           Cechy                           32269901.39
2           Polsko                          109717664.48
2           Slovensko                       16134950.67
2                                           158122516.54
3           Cechy                           32269901.39
3           Polsko                          109717664.48
3           Slovensko                       16134950.67
3                                           158122516.54
                                            474367549.62
            Cechy           stereo          19361940.72
            Cechy           TV              48404852.01
            Cechy           video           29042911.44
            Cechy                           96809704.17
```

```
             Polsko          stereo          77447763.09
             Polsko          TV              145214556.00
             Polsko          video           106490674.35
             Polsko                          329152993.44
             Slovensko       stereo          9680970.48
             Slovensko       TV              24202425.99
             Slovensko       video           14521455.54
             Slovensko                       48404852.01
1                            stereo          35496891.43
2                            stereo          35496891.43
3                            stereo          35496891.43
                             stereo          106490674.29
1                            TV              72607278.00
2                            TV              72607278.00
3                            TV              72607278.00
                             TV              217821834.00
1                            video           50018347.11
2                            video           50018347.11
3                            video           50018347.11
                             video           150055041.33

(37 row(s) affected)
```

Môžeme analyzovať údaje len za určité obdobie, napríklad za druhý mesiac, čiže vykonáme „výrez“ z kocky za určité časové obdobie.:



Obrázok 3.11 - výrez z kocky za určité časové obdobie.

```
SELECT mesiac, krajina, tovar,
SUM(cena) AS SUMA
FROM platby WHERE mesiac = 2
GROUP BY mesiac, krajina, tovar WITH CUBE
HAVING (GROUPING(mesiac)=1)OR(GROUPING (krajina)=1)OR(GROUPING (tovar)=1);
```

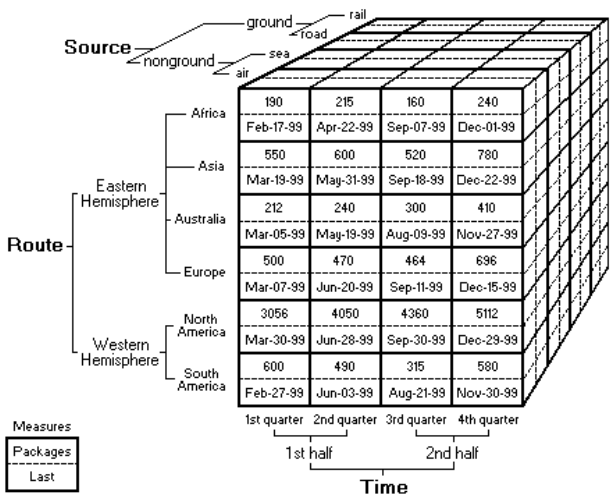
```
mesiac      krajina         tovar           SUMA
----------- -------------- --------------- ---------------------------------------
2           Cechy                          32269901.39
2           Polsko                         109717664.48
2           Slovensko                      16134950.67
2                                          158122516.54
                                           158122516.54
            Cechy           stereo         6453980.24
            Cechy           TV             16134950.67
            Cechy           video          9680970.48
            Cechy                          32269901.39
            Polsko          stereo         25815921.03
            Polsko          TV             48404852.00
            Polsko          video          35496891.45
            Polsko                         109717664.48
            Slovensko       stereo         3226990.16
            Slovensko       TV             8067475.33
            Slovensko       video          4840485.18
            Slovensko                      16134950.67
2                           stereo         35496891.43
                            stereo         35496891.43
2                           TV             72607278.00
                            TV             72607278.00
2                           video          50018347.11
                            video          50018347.11

(23 row(s) affected)
```

Podobným spôsobom môžeme analyzovať údaje len pre jeden produkt alebo pre jeden konkrétny štát. Bohužiaľ, keďže v našom príklade nemáme tabuľky dimenzií, nemôžeme sa do kocky „zavŕtať“ (drill down) trochu hlbšie a urobiť napríklad analýzu pre jednotlivé dni.

V niektorých prípadoch trvá výpočet všetkých hodnôt kocky pomerne dlhý čas. Preto sa v praxi vypočítajú kocky za určité časové obdobie, napríklad za každý deň len raz (napríklad z údajov aktualizovaného dátového skladu) a uložia do samostatnej databázy. Aj keby tento proces prebiehal niekoľko hodín, môžeme ho automaticky odštartovať po polnoci pre údaje za predchádzajúci deň. Ráno, keď manažéri a analytici prídu do práce, majú všetky údaje za predchádzajúci deň okamžite k dispozícii. Ak porovnáme čas odozvy servera, predtým trval výpočet kocky niekoľko desiatok sekúnd až hodiny, oproti tomu výber údajov z vopred vypočítanej tabuľky je prakticky okamžitý.

Tabuľky faktov a tabuľky dimenzií.

Použitím klauzuly CUBE získame prehľadný výpis napočítaných súhrnov súm pre dané rozmery. Na prvý pohľad jednoduché, ale - v praxi málo využiteľné. Kocka sa v našom jednoduchom príklade vytvárala na základe jednoduchej tabuľky, ktorá obsahuje dimenzie aj údaje. Taktiež zistíme, že jednotlivé dimenzie kocky **(mesiac, krajina, tovar)** sú už „ďalej nedeliteľné“. V reálnych kockách sa totiž často využíva tzv „drilovanie“, to znamená, že niektorú dimenziu zjemníme *(drill down)*, alebo ak naopak nás zaujímajú globálnejšie údaje, použijeme hrubšie nastavenie dimenzie *(drill up)*. Ľahko to pochopíme napríklad na časovej dimenzii. Niekedy potrebujeme údaje za mesiac, inokedy za týždeň, alebo dokonca za jeden deň, inokedy zasa potrebujeme údaje za jednotlivé roky. A to už nehovoríme o tom, koľko riadkov by v takejto interpretácii obsahovala aj značne riedka kocka, keby mala viac dimenzií.



Obrázok 3.12 - stromová štruktúra dimenzií kocky, ktorá umožňuje drillovanie.

Pre vytvorenie prakticky použiteľnej kocky potrebujeme teda logicky dva druhy tabuliek:
– tabuľky faktov.
– tabuľky dimenzií,

Na tieto tabuľky sú kladené trochu iné nároky a požiadavky, než na bežné tabuľky v relačných databázach. Tabuľky dimenzií obvykle nie sú normalizované.

Pre zaujímavosť uvedieme aspoň základné definície pre normalizované tabuľky. Návrhári tabuliek pre OLAP analýzu si môžu zopakovať, čoho budú na rozdiel od návrhárov relačných tabuliek ušetrení. Databázová tabuľka je normalizovaná vtedy ak spĺňa požiadavky pre zaradenie do niektorej normálnej formy. Ukážeme si aspoň definície prvých dvoch:

***I. normálna forma** - tabuľka sa skladá len z atomických stĺpcov. Hodnotou teda nemôže byť relácia.*

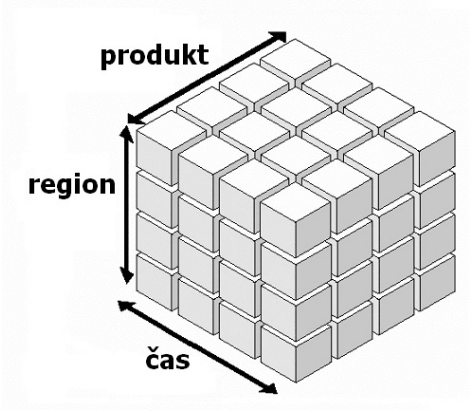
***II. normálna forma** - definuje návrh tabuľky, ktorá má viac primárnych kľúčov. Ak má tabuľka v prvej normálnej forme len jeden primárny index, automaticky spĺňa podmienku pre druhú normálnu formu. Definícia hovorí, že tabuľka spĺňa podmienky pre druhú normálnu formu vtedy, ak spĺňa podmienky prvej normálnej formy, pričom každý stĺpec, ktorý nie je primárnym kľúčom, je na primárnom kľúči úplne závislý. Tento problém je možné vyriešiť pomocou dekompozície tabuľky. Tabuľku rozdelíme na dve, alebo viac jednoduchších tabuliek, ktoré spĺňajú všetky náležitosti druhej normálnej formy.*

Tabul'ky faktov

Fakty sú numerické merné jednotky obchodovania. Tabuľka faktov je pochopiteľne najväčšia tabuľka vo hviezdicovej schéme a obsahuje veľký objem dát. Hoci hviezdicová schéma zvyčajne obsahuje len jednu tabuľku faktov, iné, hlavne DSS schémy môžu obsahovať aj viaceré tabuľky faktov. Prvotné fakty, napríklad dolárový predaj, sa môžu kombinovať alebo vypočítať pomocou iných faktov a vytvoriť tak merné jednotky. Merné jednotky sa môžu uložiť v tabuľke faktov, prípadne vyvolať, ak je to nevyhnutné, na účely vykazovania.

Tabuľky dimenzií

Obsahujú logicky alebo organizačne hierarchicky usporiadané údaje. Dimenzie sú textové popisy obchodovania. Tabuľky dimenzií sú zvyčajne menšie ako tabuľky faktov a dáta v nich sa nemenia tak často. Tabuľky dimenzií vysvetľujú všetky „prečo“ a „ako“ pokiaľ ide o obchodovanie a transakcie prvkov. Kým dimenzie vo všeobecnosti obsahujú relatívne stabilné dáta, dimenzie zákazníkov sa aktualizujú častejšie.



Obrázok 3.13 - príklad dimenzií

Tabuľky dimenzií obvykle obsahujú stromovú štruktúru. Dimenzie sa vyberajú obvykle napríklad podľa geografického **regiónu**, napríklad:

```
Region
  • Kontinent,
  • • Krajina,
  • • • Územný celok,
  • • • • Mesto
```

alebo podľa **produktovej** klasifikácie:

```
Produkt
  • Druh produktu,
  • • Kategória,
  • • • Subkategória,
  • • • • Názov produktu
```

Veľmi často sa používa ako dimenzia **čas**, napríklad:

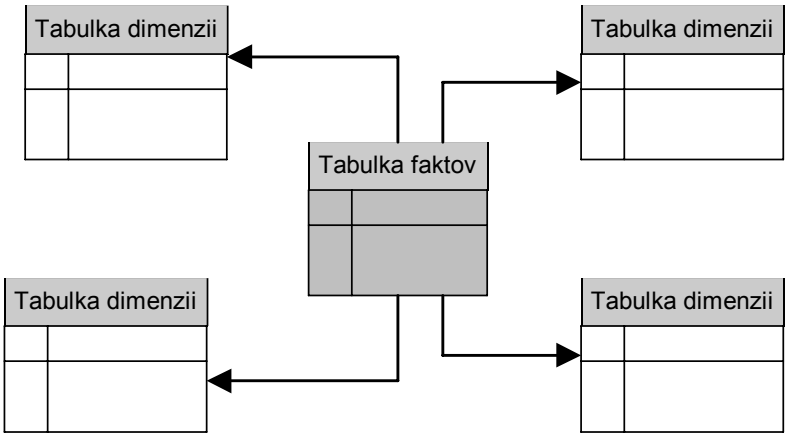
```
Čas
  • Rok,
  • • Kvartál,
  • • • Mesiac,
  • • • • Týždeň
```

počet bodiek znamená úroveň, (vnorenie) jednotlivých atribútov dimenzie

Schémy tabuliek dimenzií

Kocku vytvárame na základe dimenziálneho modelu, ktorý môže mať **hviezdicovú schému** (star schema), alebo schému **„snehovej vločky“** (snowflake schema)

Hviezdicová schéma sa skladá z tabuľky faktov obsahujúce cudzie kľúče, ktoré sa vzťahujú k primárnym kľúčom v tabuľkách dimenzií. Hviezdicová schéma nemá normalizované dimenzie ani relačné prepojenie medzi tabuľkami dimenzií, preto je veľmi ľahko pochopiteľná, ale v dôsledku nenormalizovaných dimenzií je vytvorenie takéhoto modelu relatívne pomalé, ale na druhej strane, tento model poskytuje vysoký „dopytovací výkon“ *(pozn. ako sme už uviedli, normalizácia v tomto prípade znamená, že tabuľka vyhovuje pre zaradenie do niektorej z tzv. normálnych foriem.)*



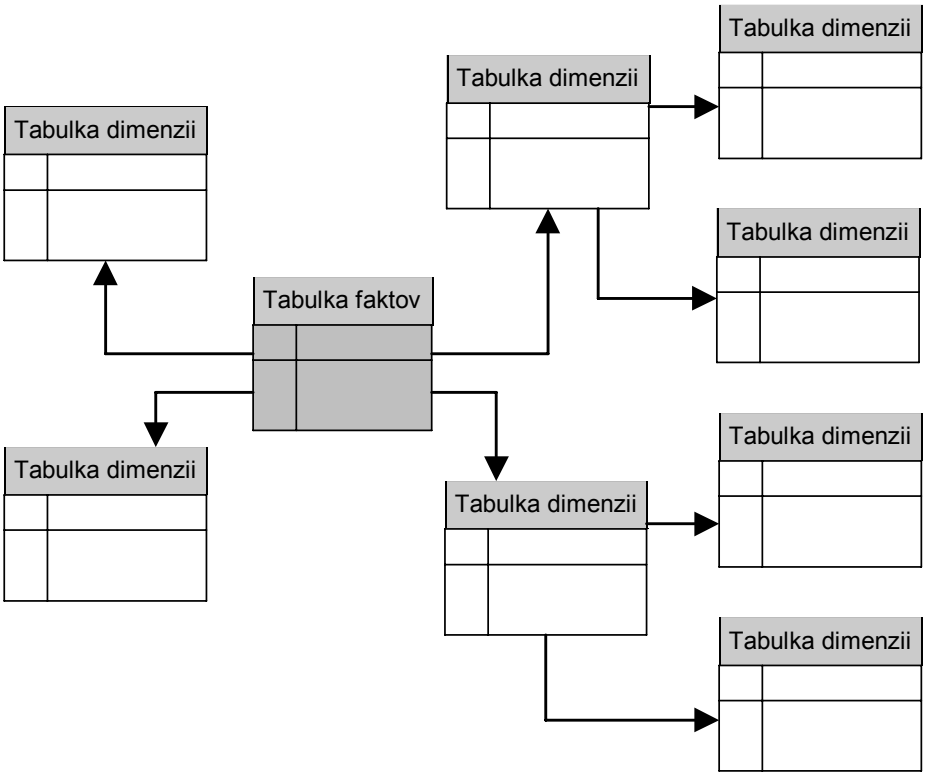
Obrázok 3.14 - Star schema

Ako príklad takejto nenormalizovanej tabuľky dimenzii ukážeme niekoľko záznamov z tabuľky **time_by_day** z databázy FoodMart.

time_id	the_date	the_day	the_month	the_year	day_of_mon	week_of_y	month_of_y	quarter
367	1997-01-01	Wednesday	January	1997	1	2.0	1	Q1
368	1997-01-02	Thursday	January	1997	2	2.0	1	Q1
369	1997-01-03	Friday	January	1997	3	2.0	1	Q1
370	1997-01-04	Saturday	January	1997	4	2.0	1	Q1
371	1997-01-05	Sunday	January	1997	5	3.0	1	Q1
372	1997-01-06	Monday	January	1997	6	3.0	1	Q1

Vidíme, že o nejakej normalizácii nemôže byť ani reči a o úspore miesta už ani nehovoríme. Len ťažko si môžeme predstaviť väčšiu nadbytočnosť. Zostaviť takúto tabuľku vyžaduje aj určité úsilie, ale pre každý jeden deň okamžite poznáme jeho poradie v týždni (vlastne názov dňa), v mesiaci, v kvartáli, v roku a podobne. Teraz už chápeme príčinu vysokého dopytovacieho výkonu, pretože všetky údaje získame naraz a nemusíme ich skladať z relačných tabuliek.

Schéma „snehovej vločky“ obsahuje niektoré dimenzie zložené z viacerých relačne zviazaných tabuliek. Tento model umožňuje rýchlejšie zavedenie údajov do normalizovaných tabuliek, ale má podstatne nižší dopytovací výkon, lebo obsahuje väčšie množstvo spojení tabuliek



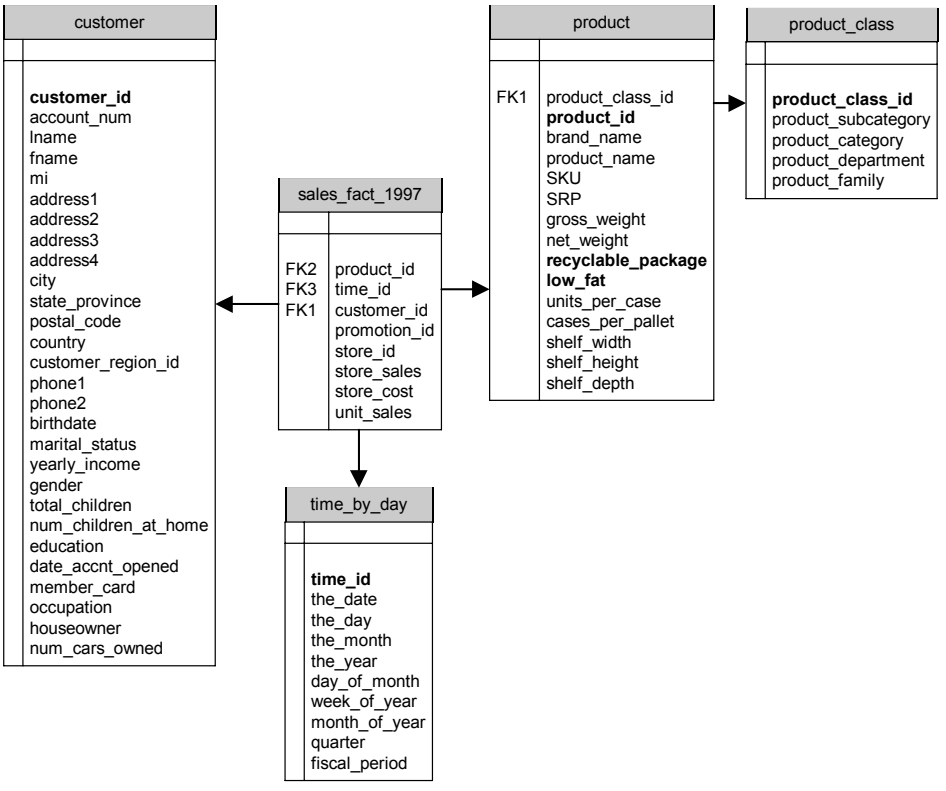
Obrázok 3.15 - Snowflake schema

O metodike a teórii návrhu tabuliek faktov a tabuliek dimenzií by sme mohli popísať veľa papiera, ale viac než teória nám určite poslúži názorný príklad nad cvičnou databázou.

FoodMart

Cvičná databáza **FoodMart** dodávaná s SQL Serverom 2000 (nainštaluje sa pri inštalácii analytických služieb). FoodMart je databáza veľkého potravinárskeho obchodného reťazca, ktorý má svoje markety v USA, Mexiku a Kanade. Pri inštalácii analytických služieb sa vytvoria aj tri cvičné kocky (Marketing, Human Resources (HR) a Expense Budget) a aj všetky potrebné dimenzie, no z pedagogického hľadiska bude rozhodne zaujímavejšie začať od začiatku a prejsť komplet celý príklad od pripojenia sa k zdroju údajov, cez vytvorenie dimenzií, výpočet kocky a pripojenie sa k analytickým údajom. Aby sme túto cvičnú databázu mohli s úspechom použiť, mali by sme ju dôkladne spoznať. (V reálnej praxi toto vývojárovi nehrozí, v procese vytvárania a ladenia ETL spoznáme štruktúru databázy alebo dátového skladu oveľa detailnejšie, než by sme si možno želali J).

Aby sme teda štruktúru cvičnej databázy mohli čo najrýchlejšie spoznať a vykresliť jej diagram, môžeme na chvíľu odbočiť od SQL Serveru 2000 a odporučíme do pozornosti ďalší vynikajúci produkt firmy Microsoft s názvom **MS Visio 2002** (staršia verzia mala označenie MS Visio 2000). Postup reverznej analýzy je v **prílohe č. 2** na konci tejto kapitoly. Budeme pracovať s tabuľkami: **sales_fact_1997** (tabuľka faktov), **customer, time_by_day, product a product_class** (tabuľky dimenzií)



Obrázok 3.16 - príklad schémy dimenzií typu „STAR“ a „SNOWFLAKE“ (diagram bol vytvorený pomocou Microsoft Visio 2002. Postup je uvedený v prílohe č. 2 na konci tejto kapitoly)

Tabuľka faktov (sales_facts_1997) je v tomto prípade pomerne jednoduchá a obsahuje záznamy o jednotlivých predajných transakciách.

product_id	time_id	customer_id	promotion_id	store_id	store_sales	store_cost	unit_sales
869	367	3449	0	6	10.6000	3.8160	5.0
1472	367	3449	0	6	6.6000	2.5080	3.0
76	367	3449	0	6	6.7600	3.2448	4.0
320	367	3449	0	6	9.7800	3.6186	3.0
4	367	3449	0	6	14.5600	4.8048	4.0
952	367	3449	0	6	11.6400	5.8200	4.0
1222	367	3449	0	6	9.8000	3.7240	4.0
517	367	7859	0	6	13.6000	5.8480	4.0

Ak sa pozrieme na tabuľku **time_by_day,** už vieme, že prvý záznam z nášho výpisu je z prvého januára 1997.

Tabuľky dimenzií - ako dimenzie pre našu jednoduchú cvičnú kocku sme vybrali tabuľky Product, Store, Customer a Time_by_day. Zatiaľ ukážeme len schému stromovej štruktúry jednotlivých tabuliek. Podrobnejšie si o jednotlivých tabuľkách a ich obsahu povieme pri návrhu jednotlivých dimenzií

Product
- Product Family,
- Department,
- Product Category,
- Product Subcategory,
- Product Name

Store
- Store Country,
- Store State,
- Store City,
- Store Name

Customers
- Country,
- State Province,
- City,
- Name

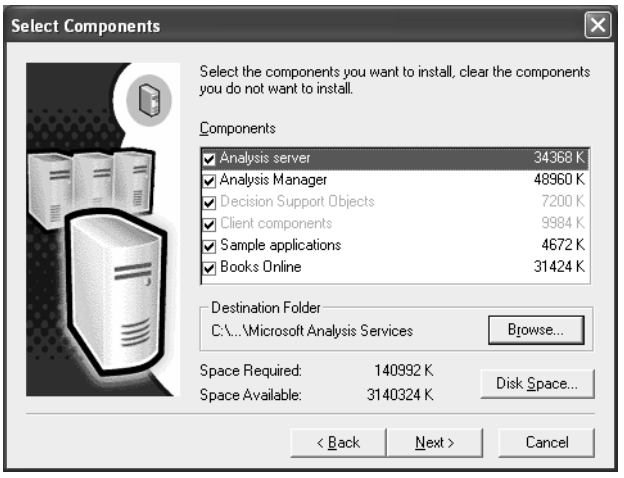
Time
- Year,
- Quarter,
- Month

Po týchto príkladoch by mala byť pozornejšiemu čitateľovi väzba medzi tabuľkou faktov a tabuľkami dimenzií pomerne jasná. Nástroje pre návrh kociek a dimenzií obvykle umožňujú aj grafické zobrazenie väzieb medzi tabuľkami faktov a dimenzií.

Vytvorenie kocky pomocou Sprievodcu vytvorením kocky

Použijeme MS SQL Server 2000, konkrétne jeho organickú súčasť OLAP Server. V závere odstavca ukážeme pripojenie k výsledkom analýzy prostredníctvom programu MS Excel z kancelárskeho balíka MS Office. Dimenzie pre cvičný príklad navrhneme tak, aby ju tvorili aj relačne zviazané tabuľky. Takúto schému dimenzií nazývame schémou „snehovej vločky" (snowflake schema). Celý postup rozčleníme do niekoľkých logických krokov. Pre prácu s analytickým serverom využijeme utilitu **Analysis Manager**, ktorú nájdeme v menu SQL Servera.

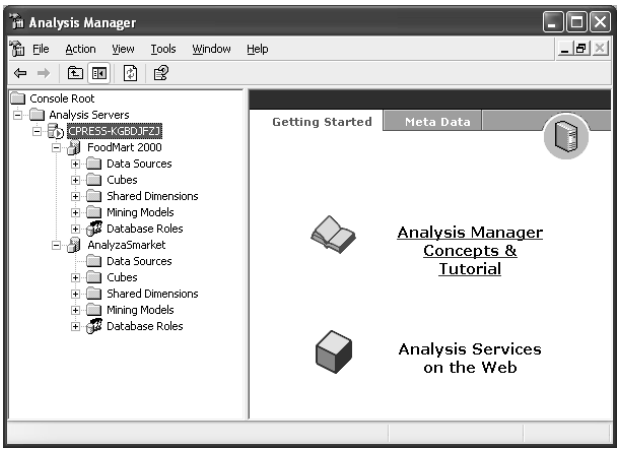
Nezabudnime si predtým z inštalačného CD okrem SQL Servera nainštalovať aj analytické služby vrátane dokumentácie a príkladov. Pre používateľov Windows XP odporúčame nainštalovať aj Service Pack 1 pre SQL Server



Obrázok 3.17 - Výber komponentov pri inštalácii analytických služieb

Pripojenie sa k zdroju údajov.

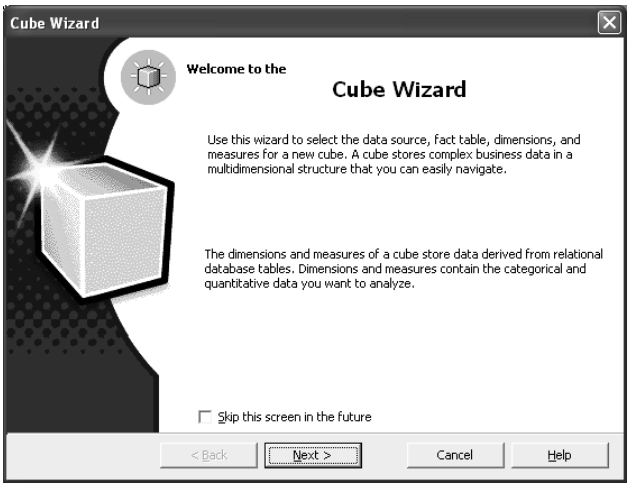
Pomocou utility **Analysis Manager**, ktorá je súčasťou konzoly **Microsoft Management Console** vytvoríme novú analytickú databázu s vhodným názvom, v našom prípade napríklad ***AnalyzaSupermarket***. Všimnime si, že jednu analytickú databázu s názvom *FoodMart 2000* už pod správou analytického servera máme nainštalovanú. Do zložky **Data Sources** pridáme odkaz na relačnú databázu pod správou SQL Servera s názvom ***Supermarket***. (vytvorili sme ju v druhej kapitole portovaním databázy FoodMart 2000.mdb do SQL Serveru). Môžeme sa pripojiť aj priamo k databáze ***FoodMart 2000*** napríklad prostredníctvom rozhrania ODBC (Open Database Connectivity). Postup konfigurácie rozhrania ODBC je popísaný v prílohe č. 1 na konci tejto kapitoly.



Obrázok 3.18 - Analysis Manager

Vytvorenie novej OLAP kocky

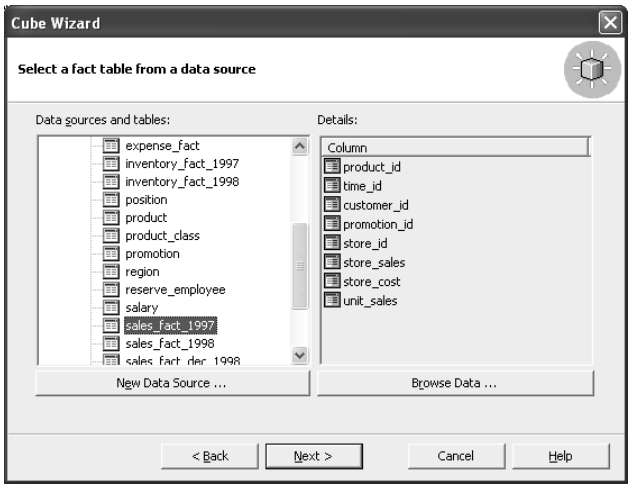
V záložke **Cubes** vytvoríme novú kocku prostredníctvom sprievodcu vytvorenia kocky. (Cube Wizard) . Celý postup je týmto sprievodcom vynikajúco navigovaný, takže ho zvládne úspešne na prvý krát aj začiatočník. Preto len stručne ukážeme jeho jednotlivé kroky.



Obrázok 3.19 - Cube Wizard

– V prvom kroku zvolíme **tabuľku faktov**, v našom prípade to bude napríklad tabuľka ***sales_fact_1997***. Nie je ťažké už podľa názvu uhádnuť, že táto tabuľka obsahuje údaje o obchodovaní v roku 1997.

Naša mierne zjednodušený výpis tabuľky faktov obsahuje tri stĺpce `product_id, time_id a customer_id`, ktoré predstavujú väzby na tabuľky dimenzií. Ďalšie tri stĺpce `store_sales, store_cost a unit_sales` obsahujú merné jednotky obchodovania.

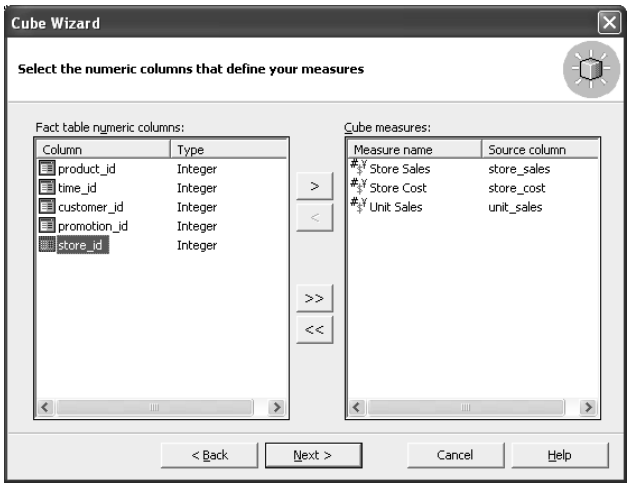


Obrázok 3.20 - Návrh tabuľky faktov

Príklad údajov uložených v tabuľke faktov

```
product_id  time_id   customer_id store_sales  store_cost  unit_sales
----------- --------  ----------- ------------ ----------  ------------
173         748       2094        4.2900       1.8447      3.0
1119        748       2094        9.5100       3.5187      3.0
1242        748       2094        7.9200       2.8512      4.0
460         748       2094        6.4400       2.7048      4.0
104         748       2094        11.6700      3.9678      3.0
...
```

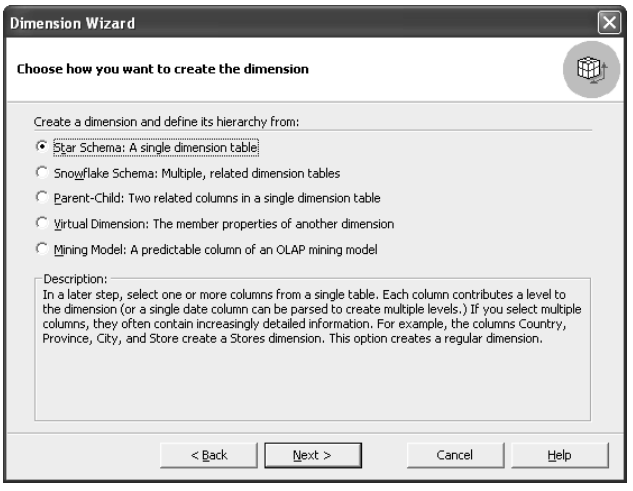
– V nasledujúcom kroku vyberieme stĺpce z tabuľky faktov, ktoré budú **mernými jednotkami pre analýzu**, v našom prípade to budú logicky stĺpce `store_sales, store_cost a unit_sales`.



Obrázok 3.21 - Výber merných jednotiek v tabuľke faktov

– **vytvorenie dimenzií**. Po výbere tabuľky faktov sa prostredníctvom nástroja **Cube Wizard** dostaneme k vytvoreniu jednotlivých dimenzií.

Dve dimenzie tvoria jednoduché tabuľky ***customer*** a ***time_by_day***. Prvá z nich charakterizuje zákazníkov, v našom príklade nás bude zaujímať hlavne ich regionálna charakteristika. Druhá tabuľka predstavuje časovú dimenziu. Tretiu dimenziu budú tvoriť dve relačne zviazané tabuľky ***product*** a ***product_class***. Aj pre vytvorenie dimenzií nám bude ponúknutý ďalší vynikajúci sprievodca - **Dimension Wizard**, ktorý na chvíľu prevezme žezlo od Cube Wizarda Môžeme si zvoliť vhodné schémy dimenzií.



Obrázok 3.22 - výber typu dimenzie

Z ponúkaných možností využijeme v našom cvičnom príklade len prvé dve.

Star Schema – dimenziu bude tvoriť jeden, alebo viac atribútov (stĺpcov) z jednej tabuľky. Každý z atribútov tvorí jednu z hierarchických úrovní stromovej štruktúry dimenzie, napríklad stĺpce štát, kraj, región, mesto pre dimenziu regionálneho typu. Takúto schému použijeme pre dimenziu **Zakaznici** ***(Customers)***

Zakaznici

- Country,
- • State Province,
- • • City,
- • • • Name

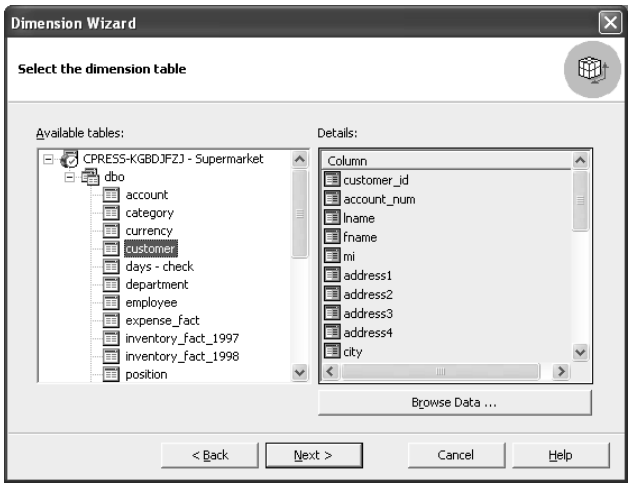
Pri návrhu dimenzie postupujeme podľa pokynov sprievodcu. Dôležitý je hlavne dialóg v ktorom sa rozhodujeme pre štandardnú, alebo časovú dimenziu (viď obrázok pri návrhu dimenzie čas). Dimenzia **Zákazníci** je samozrejme typickým príkladom štandardnej dimenzie. Postupne navrhneme jednotlivé úrovne tejto dimenzie jednoduchým premiestnením stĺpcov z ľavej strany dialógu na pravú. Mierny a veľmi ľahko riešiteľný problém sa nám vyskytol pri mene zákazníka. V tabuľke ***customer*** máme meno a priezvisko zákazníka uložené v dvoch stĺpcoch *fname a lname*.

customer_id	lname	fname	city	state_province
1	Nowmer	Sheri	Tlaxiaco	Oaxaca
2	Whelply	Derrick	Sooke	BC
3	Derry	Jeanne	Issaquah	WA
4	Spence	Michael	Burnaby	BC
5	Gutierrez	Maya	Novato	CA
6	Damstra	Robert	Lynnwood	WA

Riešení je niekoľko, napríklad vytvoriť novú tabuľku (alebo pohľad) v ktorej zlúčime stĺpce *fname* a *lname*. Kto pozorne čítal kapitolu o ETL už najefektívnejšie a zároveň aj najelegantnejšie riešenie pozná. Pri transformácii tabuľky sme namiesto stĺpcov **lname** a fname vytvorili jeden stĺpec s názvom **cname** takže tabuľka potom bola v tvare:

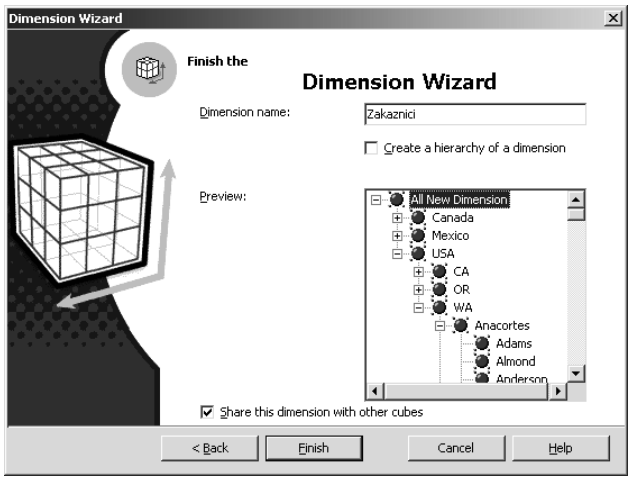
customer_id	**cname**	city	...
1	**Nowmer Sheri**	Tlaxiaco	
2	**Whelply Derrick**	Sooke	
3	**Derry Jeanne**	Issaquah	
4	**Spence Michael**	Burnaby	
...			

Ak sme túto etapu preskočili, môžeme sa uspokojiť len s priezviskom zákazníka **lname**.



*Obrázok 3.23 - Návrh dimenzie **Zakaznici***

Nakoniec dimenziu pomenujeme a v okne Preview si môžeme prípadne prehliadnuť jej jednotlivé úrovne. Nezabudnime v dolnej časti dialógu zaškrtnúť políčko *Share this dimension with other cubes*, aby sme mohli už raz dobre navrhnutú dimenziu využívať aj vo viacerých kockách príslušného projektu.



*Obrázok 3.24 - dokončenie návrhu dimenzie **Zákazníci** a jej pomenovanie.*

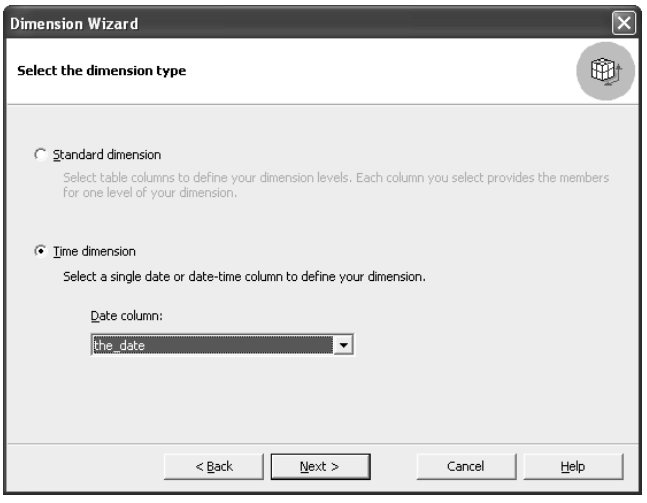
Rovnaký typ **Star schema** použijeme aj pre návrh časovej dimenzie.

Čas
- Year,
- Quarter,
- Month

Časovú dimenziu vybudujeme na základe tabuľky ***time_by_day***.

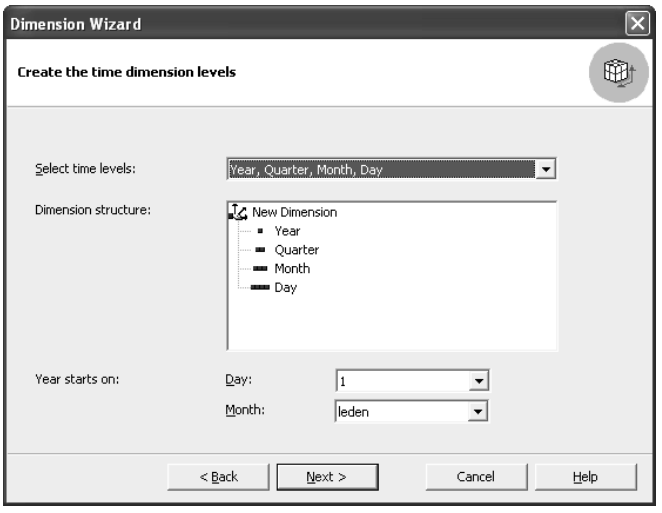
time_id	the_date	the_day	the_month	the_year	day_of_month	month_of_year	quarter
738	1998-01-07	Wednesday	January	1998	7	1	Q1
739	1998-01-08	Thursday	January	1998	8	1	Q1
740	1998-01-09	Friday	January	1998	9	1	Q1

Snáď nie je ani potrebné zdôrazňovať, že v dialógu pre voľbu typu dimenzie sa rozhodneme pre časovú dimenziu.



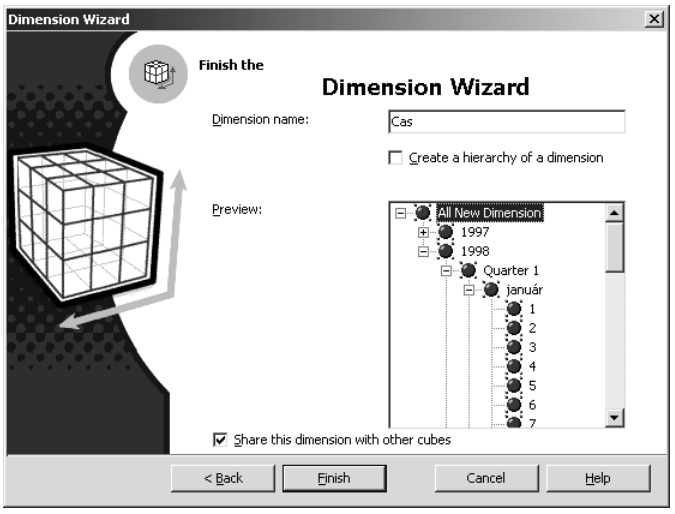
Obrázok 3.25 - výber medzi štandardným a časovým typom dimenzie.

Návrh jednotlivých úrovní časovej dimenzie už zvládne sprievodca sám a ponúkne nám výsledný dialóg:



Obrázok 3.26 - sprievodcom navrhnuté úrovne časovej dimenzie.

Na prvý pohľad sa to nezdá, ale sprievodca pri návrhu časovej dimenzie za nás vykonal pomerne veľa práce. Zostáva už len pomenovať a prehliadnuť si dimenziu **Čas**.



Obrázok 3.27 - dokončenie návrhu dimenzie ***Čas*** *a jej pomenovanie*

Zostáva nám navrhnúť poslednú - **produktovú** dimenziu. Tu použijeme **Snowflake Schema** - pri ktorej dimenzie sú tvorené jedným, alebo viacerými stĺpcami z niekoľkých relačne zviazaných tabuliek. Podobne ako pri hviezdicovej schéme každý z atribútov tvorí jednu z hierarchických úrovní stromovej štruktúry dimenzií.

Produkty

- Product Family,
- Product Department,
- Product Category,
- Product Subcategory,
- Product Name

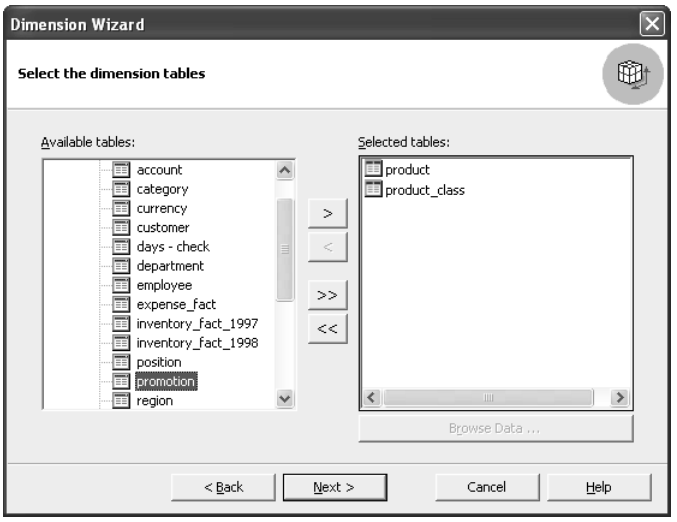
Ak sa pozrieme na zjednodušenú tabuľku **product**:

```
product_class_id product_id  product_name                    ...
---------------- ----------- ------------------------------
30               1           Washington Berry Juice
19               6           Washington Cola
35               12          Jeffers Oatmeal
35               13          Jeffers Corn Puffs
62               16          Blue Label Canned Beets
62               17          Blue Label Creamed Corn
...
```

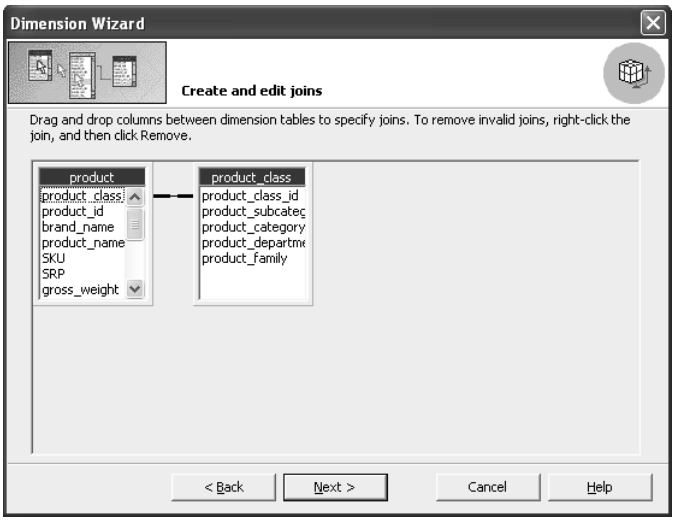
a tabuľku **product_class**:

```
product_class_id product_subcategory product_category  product_department  product_family
---------------- ----------------------------------- -------------------- --------------
1                Nuts                Specialty         Produce             Food
2                Shellfish           Seafood           Seafood             Food
3                Canned Fruit        Fruit             Canned Products     Food
4                Spices              Baking Goods      Baking Goods        Food
5                Pasta               Starchy Foods     Starchy Foods       Food
6                Yogurt              Dairy             Dairy               Food
...
```

vidíme, že dimenziu **Produkty** môžeme podľa našich predstáv vytvoriť len z obidvoch relačne zviazaných tabuliek. Postup je rovnaký ako pri Star schema, s tým rozdielom, že si vyberieme dve tabuľky: *product a product_class*.



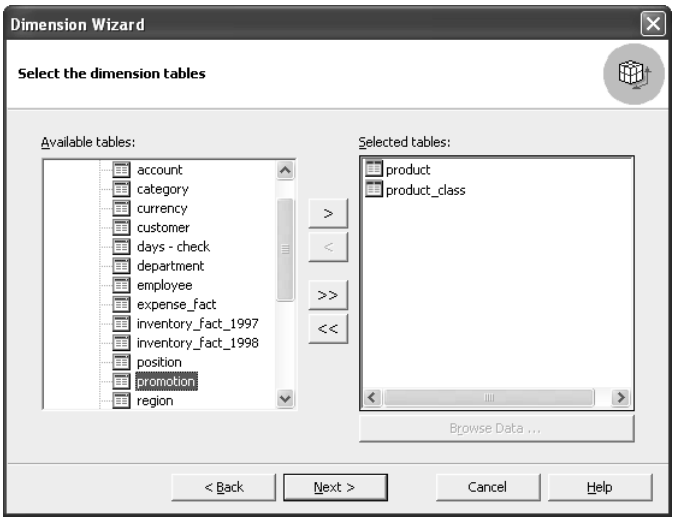
*Obrázok 3.28 - výber relačne zviazaných tabuliek pre dimenzie **Produkty***



Obrázok 3.29 - výber relačne zviazaných tabuliek pre snowflake schema

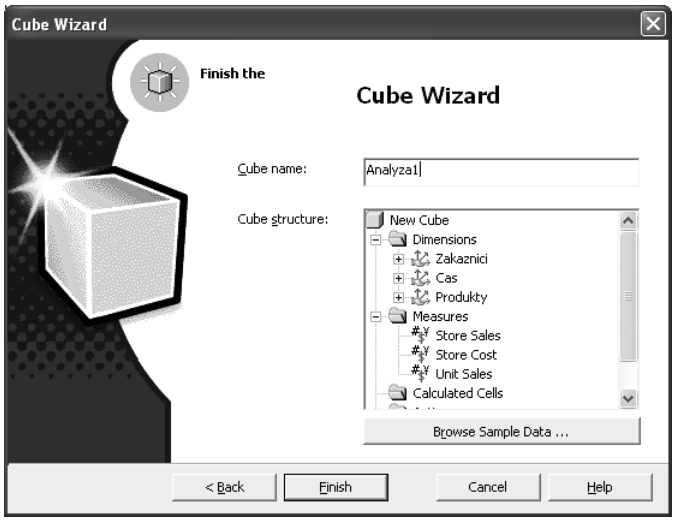
S výnimkou úrovne dimenzie *product_name*, ktorá je z tabuľky **product**, ostatné dimenzie
- *product_family,*
- *product_department,*
- *product_category a product_subcategory*

sú z tabuľky **product_class**.



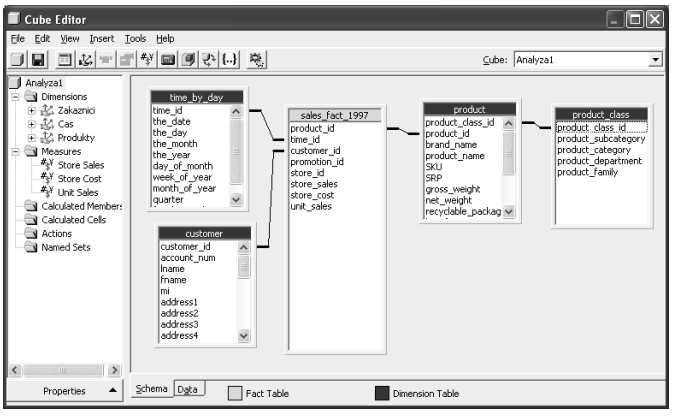
Obrázok 3.30 - Návrh dimenzie Produkty

Pomenovaním dimenzie **Produkty** sme ukončili návrh dimenzií. Réžiu znovu preberie Cube Wizard. Po úspešnom návrhu dimenzií prakticky finišujeme aj s návrhom kocky. Kocku pomenujeme napríklad **Analýza1**



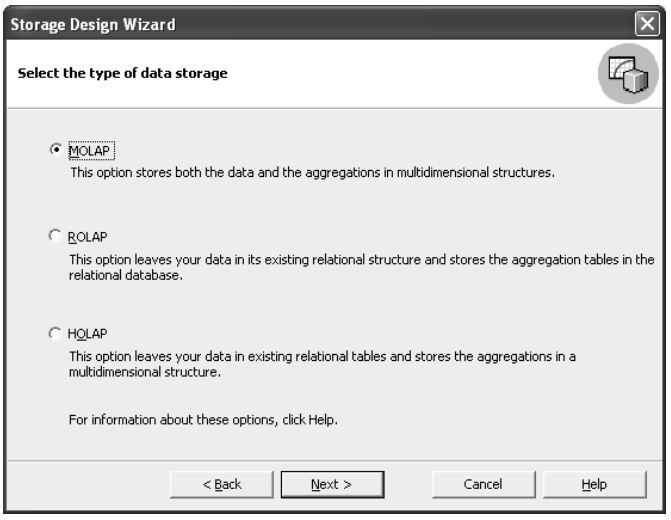
Obrázok 3.31 - Dokončenie návrhu kocky a jej pomenovanie

Po ukončení návrhu kocky nás Analysis Manager prepne do utility **Cube Editor**. Tento nástroj má v pravej časti dve záložky: **Schema** a **Data**. Záložka **Schema** obsahuje schému tabuliek faktov a dimenzií. Pozor, v záložke Data máme zatiaľ len cvičné údaje. Dôvod je jednoduchý. Výpočet zložitejšej kocky obvykle trvá aj niekoľko hodín a ak by sme chceli návrh a zobrazenie takejto kocky demonštrovať napríklad na nejakej prezentácii, ocenili by sme, že aby sme ukázali štruktúru údajov, a nemuseli pritom čakať na dopočítanie celej kocky.



Obrázok 3.32 - Cube Editor

My máme kocku pomerne jednoduchú, takže môžeme pokračovať výpočtom kocky a uložením napočítaných hodnôt. Po aktivácii tlačidla **Process Cube** v nástroji Cube Editor, nám ponúkne pomocnú ruku ďalší sprievodca - **Storage Design Wizard**, ktorý nám pomôže s návrhom úložiska pre kocku.

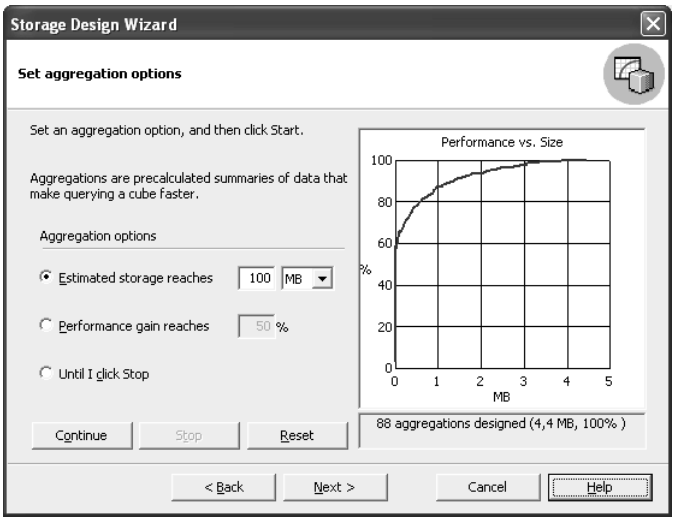


Obrázok 3.33 - optimalizácia úložiska pre kocku

Hlavne pri viacrozmerných veľkých kockách je dôležité stanoviť optimum medzi priestorom, ktorý naša kocka zaberie na disku a rýchlosťou výpočtu a prístupu k údajom. Sprievodca nám ponúkne tri typy úložisk"

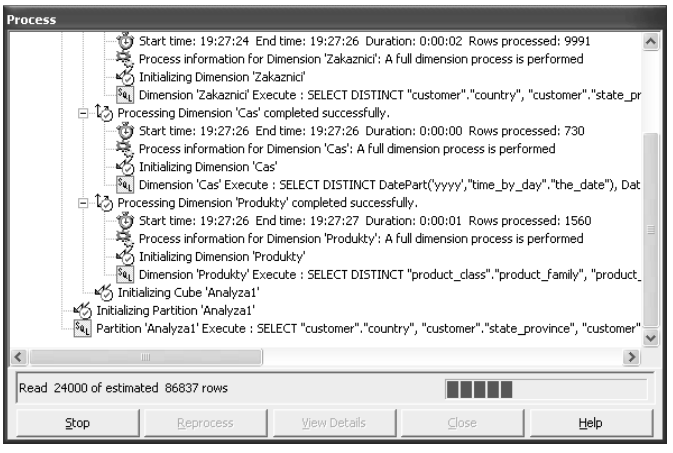
MOLAP – úložisko optimalizované pre multidimenzionálne štruktúry
ROLAP – úložisko optimalizované pre relačné štruktúry.
HOLAP – kombinácia MOLAP a ROLAP, kedy údaje ostanú v relačných štruktúrach a vypočítané agregácie budú uložené v multidimenzionálnych štruktúrach.

Zvolíme si napríklad úložisko údajov typu MOLAP a v spolupráci so sprievodcom vypočítame optimalizovanú veľkosť úložiska. V našom prípade hoci sme boli veľkorysí a nastavili sme hranicu pre úložisko kocky na 100 MB, naša kocka zaberie len 4.4 Megabajtov.



Obrázok 3.34 - optimalizácia úložiska pre kocku

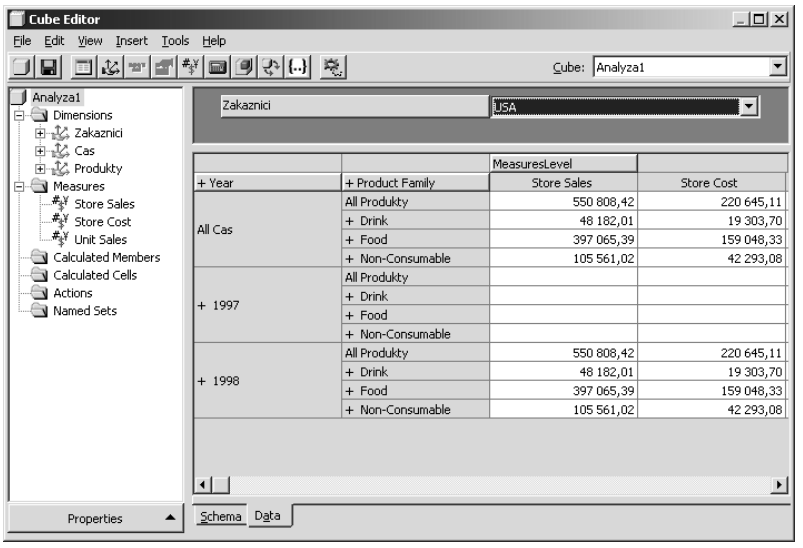
Po zatlačení tlačidla Continue ideme do finále a zahájime výpočet kocky. Ako sme už uviedli, tento proces môže trvať aj niekoľko hodín. O jeho priebehu sme informovaní v prehľadnom dialógu. Proces je náročný na pamäť, preto pred jeho odštartovaním odporúčame ukončiť všetky ostatné aplikácie.



Obrázok 3.35 - priebeh výpočtu kocky

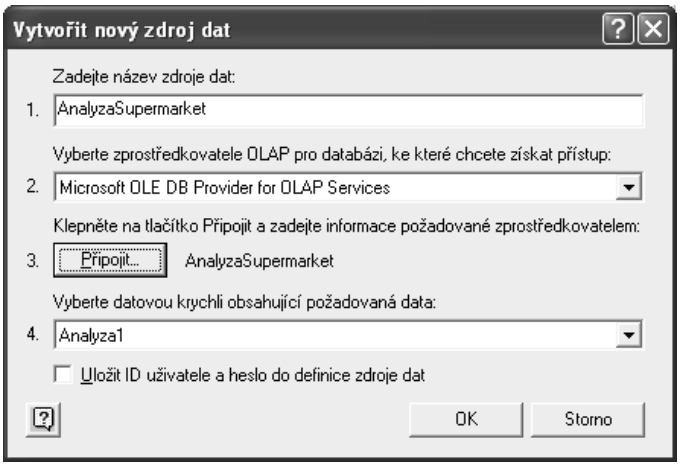
Po ukončení výpočtu máme v záložke Data utility Cube Editor skutočné údaje, ktoré môžeme metódou drag - and - drop ľubovoľne v tabuľke preskupovať a organizovať. Môžeme sa zavrtávať čoraz hlbšie do jednotlivých dimenzií, alebo naopak skúmať len globálne súhrny údajov.

Obrázok 3.36 - výsledok analýzy



Prezeranie výsledkov analýzy pomocou MS Excel

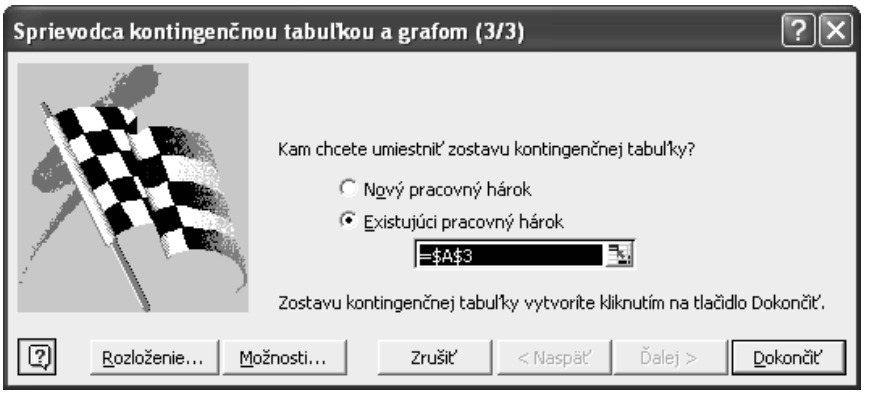
Pre prezeranie výsledkov analýzy môžeme použiť aj rôzne klientské programy. Najdostupnejší bude pravdepodobne program **MS Excel 2000** (alebo XP) z balíka MS Office 2000, alebo Office XP. Pripojiť sa môžeme z ľubovoľného klientského počítača, ktorý má povolený prístup k výsledkom analýzy. Zvolíme položku menu **Data → Načíst externí data → Nový Databázový dotaz**. Zobrazí sa nám dialóg pre pripojenie sa k externému zdroju údajov so záložkami **Databáze**, **Dotazy** a **Datové krychle OLAP**. Vyberieme záložku **Datové krychle OLAP** a pripojíme sa k našej kocke Analyza1.



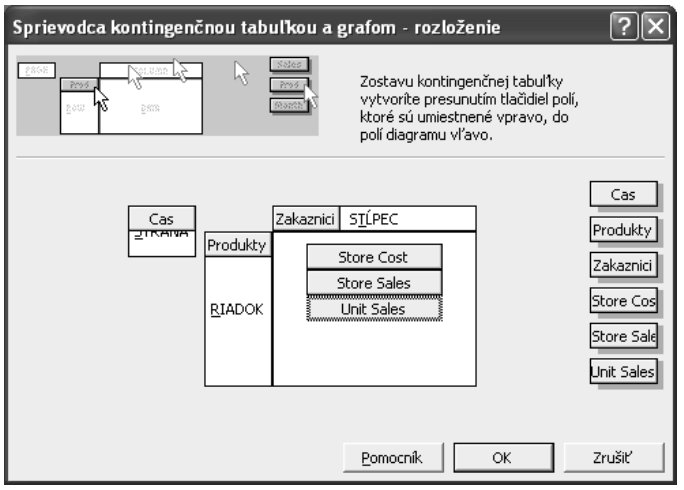
Obrázok 3.37 - pripojenie sa ku kocke Analyza1 z programu MS Excel 2000

Ani program MS Excel 2000 nás nenechá trápiť sa zo zobrazovaním výsledkov analýzy samých a ponúkne nám **Sprievodcu kontingenčnou tabuľkou**. O kontingenčných tabuľkách (pivot table) sa viac dočítame k dokumentácii k Excelu, prípadne z odbornej literatúry venovanej tomuto tabuľkovému procesoru. Novú kontingenčnú tabuľku umiestnime na existujúci hárok zošitu Excel do ľavej hornej časti.

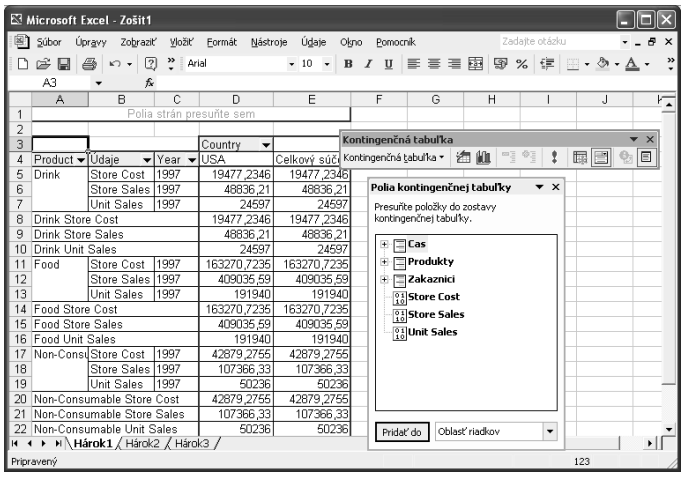
Obrázok 3.38 - sprievodca kontingenčnou tabuľkou a grafom



V tejto fáze máme dve možnosti. Môžeme po zatlačení tlačidla **Rozloženie** využiť ďalšieho kúzelníka, ktorý nás bude sprevádzať pri presune tlačidiel s názvami faktov a dimenzií z ľavej časti dialógového okna do jednotlivých polí návrhového formulára. My sme vytvorili takýto návrh



Obrázok 3.39 - sprievodca rozložením polí kontingenčnnej tabuľky



Obrázok 3.40 - výsledok návrhu kontingenčnnej tabuľky

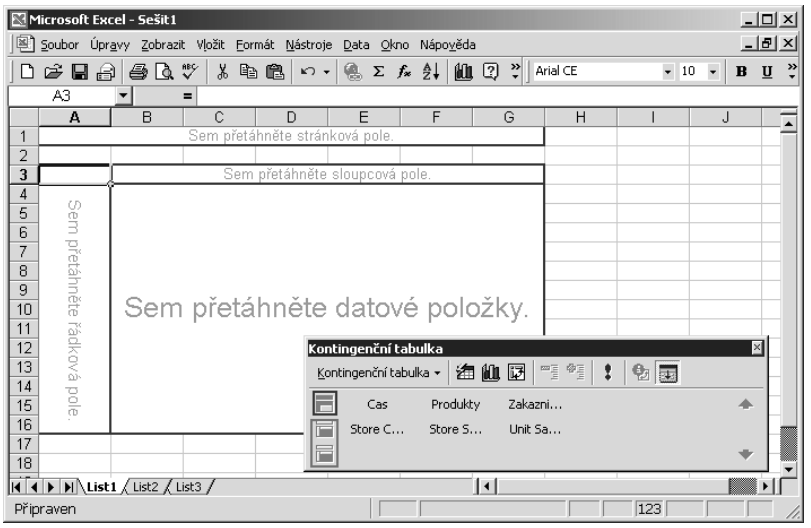
Ďalšie podrobnosti sa dozvieme z interaktívnej nápovedy, ktorú nám pomocník v každej fáze návrhu ponúka.

Ak namiesto tlačidla Rozloženie zatlačíme v prvom dialógu sprievodcu tlačidlo **Dokončiť**, dostaneme sa do návrhového listu pre kontingenčnú tabuľku priamo v zošite Excell. Tento spôsob návrhu kontigenčnej tabuľky je možno ešte pohodlnejší, ako spôsob popísaný predtým

Na ploche listu máme návrhovú šablónu s rámcovými pokynmi:

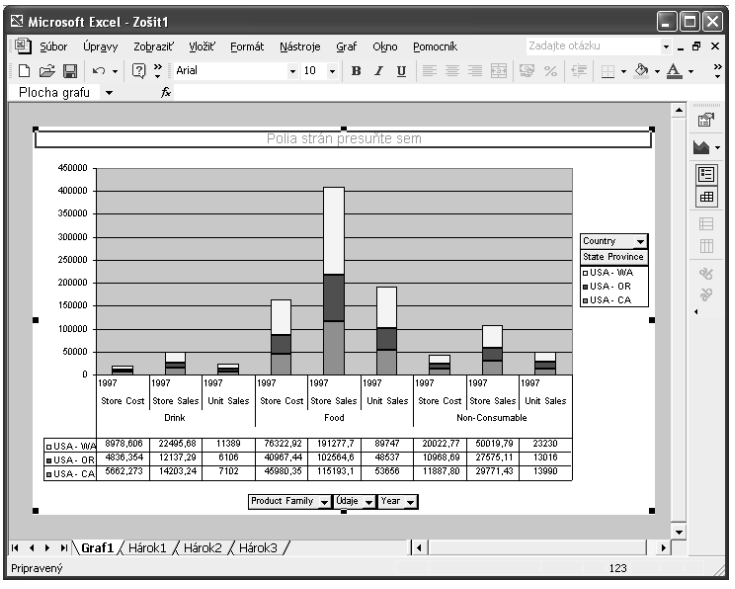
– Sem presunte stránkové polia
– Sem presunte stĺpcové polia
– Sem presunte riadkové polia
– Sem presunte dátové položky

Obrázok 3.41 - interaktívny návrh kontingenčnej tabuľky



Podľa toho, ako chceme mať tabuľku usporiadanú, na stránkové, stĺpcové a riadkové polia presunieme vhodné dimenzie a na dátové položky vhodné atribúty z tabuľky faktov. V pomocnom dialógovom okne máme aj naznačené ktoré položky patria na ktoré miesto. (ikony pre a ikony pre fakty sme dali pre názornosť do rámčekov.). Po vytvorení tabuľky sa môžeme ľubovoľne vnárať (drill down) a vynárať (drill up) v jednotlivých dimenziách.

Týmto sa možnosti programu MS Excel zďaleka nekončia. Stačí napríklad kliknúť na ikonu grafu a máme výsledky analýzy v prehľadnej forme, ktoré môžeme predložiť príslušnému manažérovi.



Obrázok 3.42 - zobrazenie výsledkov analýzy vo forme grafu.

Musíme si uvedomiť, že Excel vystupuje voči OLAP serveru ako klient, to znamená, že ak uložíme pracovný hárok a neskôr ho otvoríme a budeme chcieť napríklad meniť rozsah, alebo presnosť dimenzií, podarí sa nám to len za predpokladu, že budeme mať spojenie na OLAP server. Ak si zobrazíme po zatlačení ľavého tlačidla myši dialóg pre nastavenie parametrov, môžeme napríklad nastaviť, aby sa po každom otvorení hárku jeho obsah aktualizoval, alebo automatickú periodickú aktualizáciu v nastavenom časovom intervale.

Záver

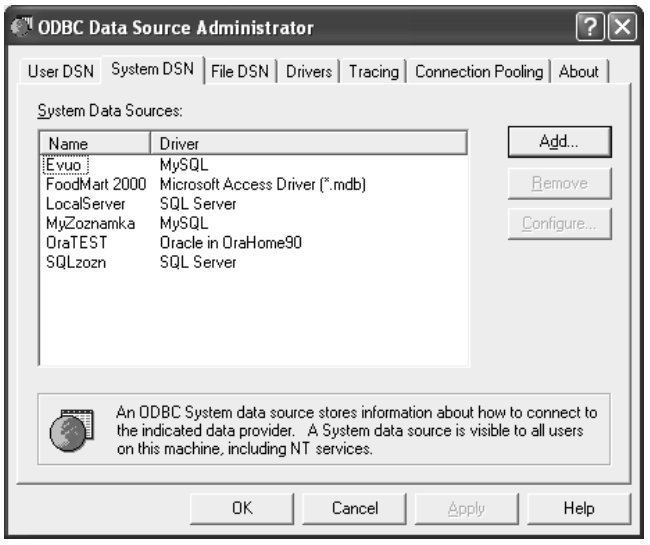
Pomocou jednoduchého príkladu sme demonštrovali kompletný postup pre vytvorenie analytickej databázy vrátane vytvorenia dimenzií a pripojenie klientskej aplikácie (Excelu k vytvorenej kocke)

Príloha 1 ku kapitole OLAP

pripojenie sa k zdroju údajov cez ODBC

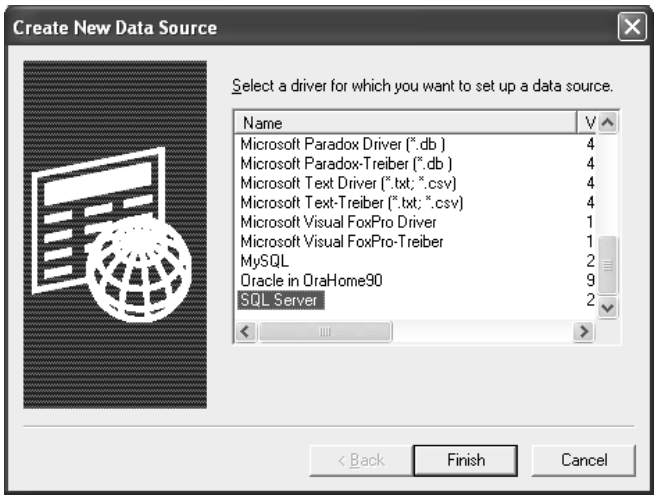
V kapitole venovanej OLAP analýze preberáme rôzne možnosti pripojenia sa k zdroju údajov. Jedným z najčastejšie používaných rozhraní je **ODBC** (Open Database Connectivity) je otvorené databázové rozhranie, ktoré definuje spôsob prístupu windowsovských aplikácií k údajom v databázach. Ukážeme si príklad konfigurácie pripojenia pre databázu pod správou SQL Serveru. Pre mnohých je to samozrejmosť, preto sme to zahrnuli ako prílohu ku kapitole, ale každý raz začínal...

Pomocou nástroja **ODBC Data Source Administrator**. (*Menu Windows **Control Panels**, vo Windows 2000 a Windows XP je v zložke **Administrative Tools**)*



*Obrázok p1 - **ODBC** Data Source Administrator*

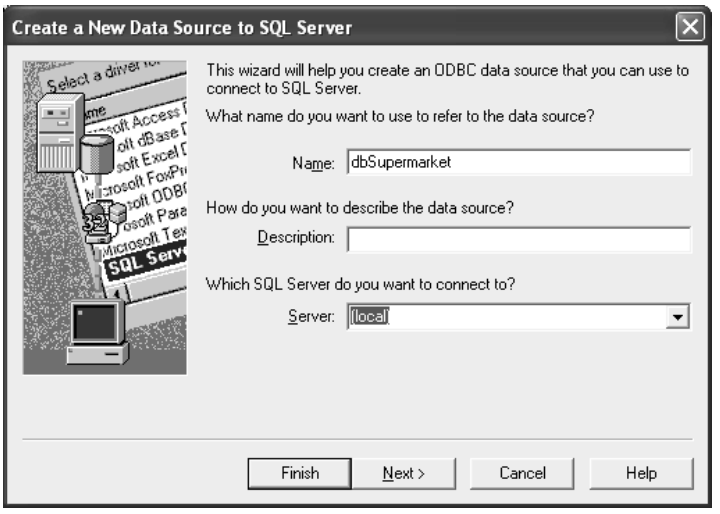
tlačidlo Add... vytvoríme nové pripojenie typu **System DSN** s názvom napríklad **dbSupermarket**. V dialógu **Create new datasource** zvolíme možnosť **SQL Server**.



*Obrázok p2- **ODBC** Create New Datasource*

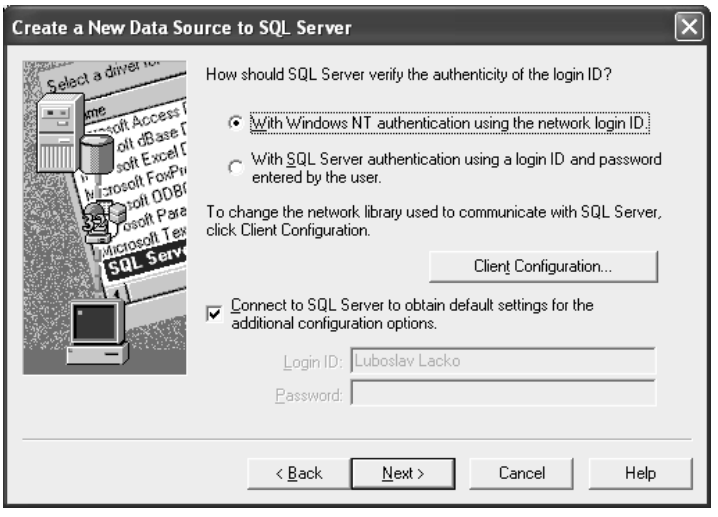
V ďalšom dialógu nastavíme potrebné parametre pre pripojenie k našej databáze **dbSupermarket**.

*Obrázok p3 - **ODBC** Create a New Datasource to SQL Server*



*Name: **dbSupermarket***
*Server: názov servera, ak sa jedná o lokálny server, napríklad na notebooku nastavíme **local***

Nasleduje nastavenie parametrov pre pripojenie k vybranému SQL Serveru a autentifikáciu klienta.

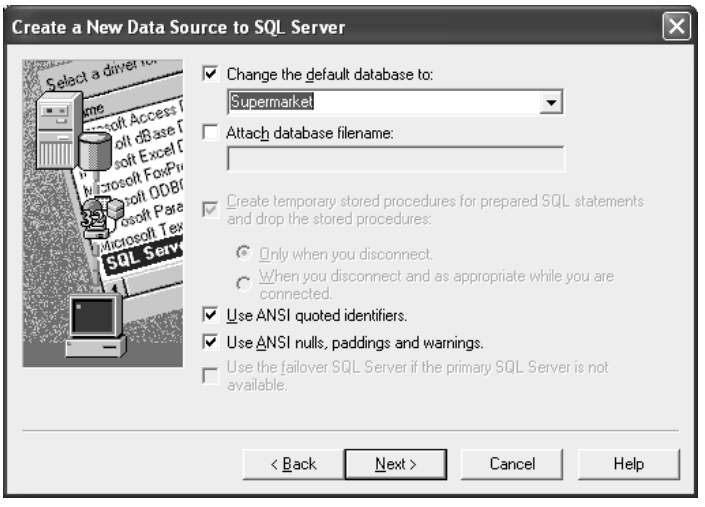


*Obrázok p4 - **ODBC** Create a New Datasource to SQL Server II*

*Autentifikácia: **With Windows NT authentication...***

V ďalšom dialógu nastavíme meno databázy, ku ktorej sa chceme pripojiť, v našom prípade to bude Supermarket.

Obrázok p5 - ODBC Create a New Datasource to SQL Server III



Môžeme nastaviť jazyk, v ktorom nám bude server oznamovať chybové hlásenia a nakoniec otestujeme pripojenie.

Obrázok p6 - ODBCMicrosoft SQL Server Setup

Optimistický oznam svedčí o úspešnom pripojení:

```
Microsoft SQL Server ODBC Driver Version 03.81.7713
Running connectivity tests...
Attempting connection
Connection established
Verifying option settings
Disconnecting from server

TESTS COMPLETED SUCCESSFULLY!
```

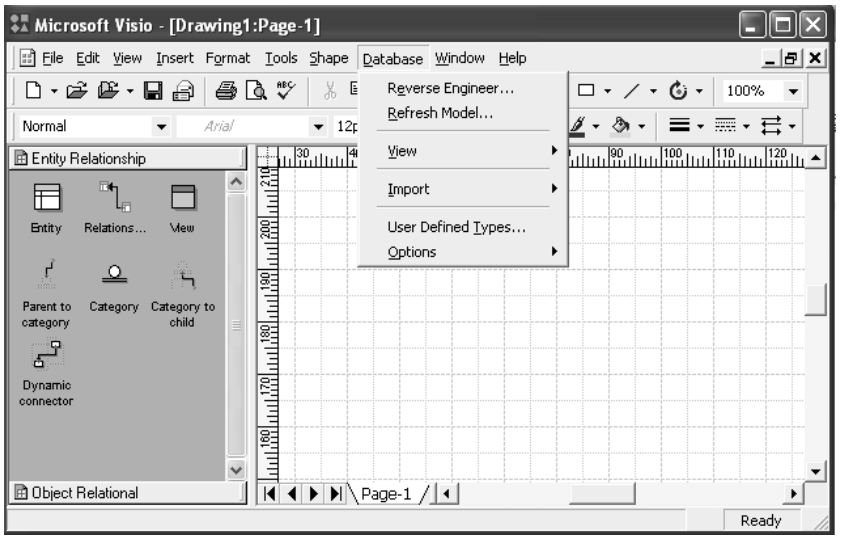
Príloha ku kapitole OLAP

MS Visio 2002 - reverzná analýza štruktúry cvičnej databázy FoodMart

Pri štarte produktu Microsoft Visio 2002 vyberieme z ponúknutých možností typ diagramu, ktorý budeme vytvárať, v našom prípade Database Model Diagram. Nebojte sa, predchádzajúca veta bola trochu zavádzajúca, sami nebudeme vytvárať takmer nič, všetko za nás vykoná Visio 2000 (pre tento príklad sme použili staršiu verziu Visio 2000).

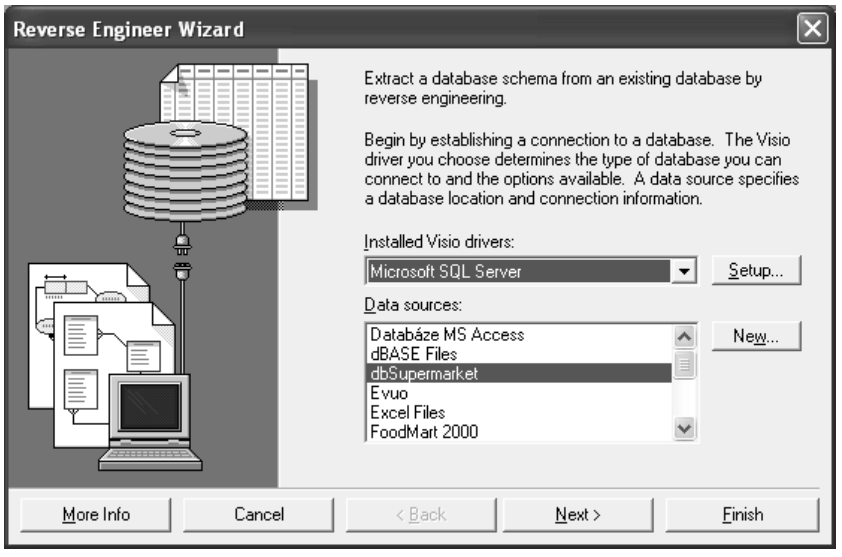
Obrázok p7 - Visio 2000 úvodný dialóg, výber typu diagramu

Ako sme naznačili v predchádzajúcom odstavci, bolo by chybou pustiť sa v tejto fáze do „manuálnej" tvorby diagramu, preto si symboly pre tabuľky, relácie a pohľady v pravej časti programu nevšímame, ale použijeme položku menu: ***Database - Reverse Engineer***



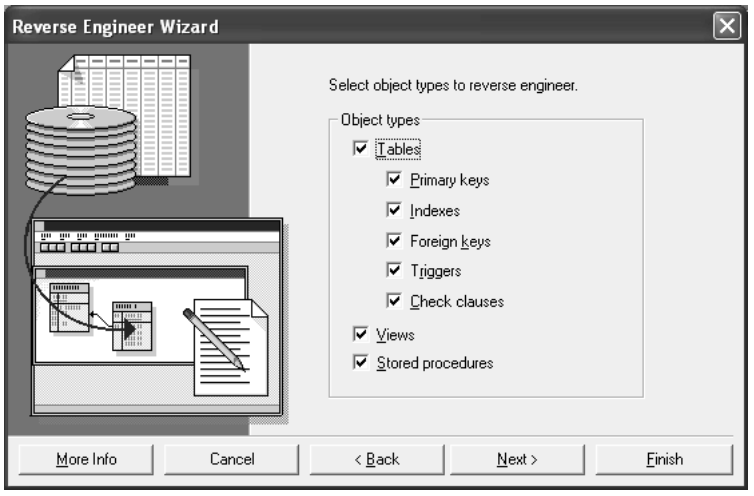
*Obrázok p8 - **Visio 2000** výber funkcie Reverse Engineer*

Zobrazí sa úvodný dialóg sprievodcu s názvom Revese Engineer Wizard. Pomocou neho vyberieme zdroj údajov, ktorý je zaregistrovaný cez rozhranie ODBC. Postup pre zaregistrovanie databázy SQL Serveru 2000 je uvedený v prílohe na konci tejto kapitoly.



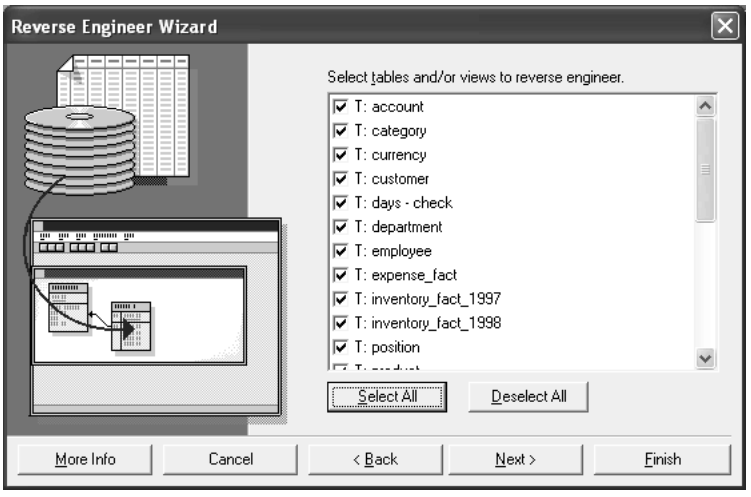
*Obrázok p9 - **Visio 2000** úvodný dialóg Reverse Engineer Wizard*

V nasledujúcom dialógu vyberieme druhy objektov v databáze, ktoré budú predmetom reverznej analýzy.



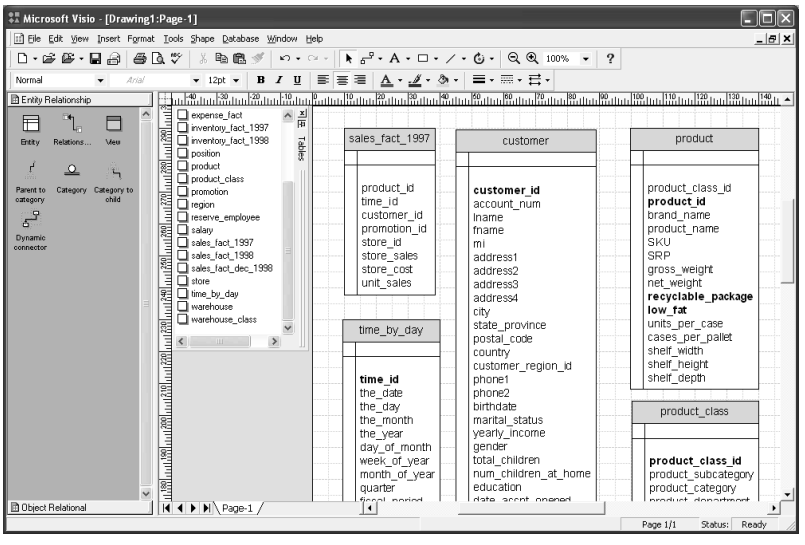
*Obrázok p10 - **Visio 2000** výber objektov pre reverznú analýzu*

Pokračujeme výberom tabuliek.



Obrázok p11 - Visio 2000 výber konkrétnych objektov () pre reverznú analýzu

Po zatlačení tlačidla **Next** postúpime do finále. Kostru diagramu vytvoríme kliknutím na štvorcový symbol vedľa názvu tabuľky a jeho presunutím na kresliacu plochu. Tento postup opakujeme pre všetky tabuľky, ktoré sú predmetom nášho záujmu.



*Obrázok p12 - **Visio 2000** tvorba diagramu*

Nakoniec usporiadame tabuľky na ploche podľa našich predstáv, zadefinujeme väzby a... program Visio 2002 nám toho poskytne oveľa viac, ale vytýčenú úlohu zobraziť štruktúru databázy FoodMart sme splnili, takže prípadný ďalší záujem databázového vývojára o tento produkt uspokojí štúdium produktovej dokumentácie.

Data Mining
(dolovanie, odkrývanie dát)

Ako vyplýva z názvu, jedná sa v podstate o ťažbu údajov. Na základe nazbieraných údajov sa pokúšame zistiť (odkrývať) rôzne závislosti, ktoré tieto údaje v sebe skrývajú. Data mining umožňuje vyhľadávať vzory informácií v údajoch. Je založený na heuristických algoritmoch, neurónových sieťach a iných pokročilých softvérových technológiách a metódach umelej inteligencie. Data Mining pomáha sledovať a analyzovať trendy a predvídať udalosti. Ako príklad môžeme uviesť spoločnosti, ktoré vydávajú kreditné karty. Na základe transakcií, nákupov a iných operácii, ktoré vykonáva majiteľ kreditnej karty je možné získať veľké množstvo údajov o správaní sa klienta aj o jeho pohyboch v geografických dimenziách. Časom by sa im určite podarilo zistiť rôzne súvislosti a závislosti a vypracovať akýsi model správania každého majiteľa kreditnej karty. Nepopulárne dôsledky zneužitia takéhoto modelu si dokáže každý domyslieť sám (smerovanie reklamných akcií, tajná služba, súkromný detektív najatý žiarlivou manželkou, konkurenčná firma, marketingové a reklamné agentúry...). Málo kto si uvedomí, že prípadný model správania sa majiteľa kreditnej karty môže byť v niektorých prípadoch užitočný aj pre samotného majiteľa. Predstavme si situáciu, keď mu bude karta odcudzená a niekto sa naučí napodobniť jeho podpis. Prevádzkovateľ kreditnej karty na základe modelu doterajšieho správania sa klienta môže zistiť, že klient sa odrazu správa „nejako divne" a ak usúdi, že toto správanie je s najvyššou pravdepodobnosťou spôsobené odcudzením kreditnej karty, môže kartu sám zablokovať a účastníkovi zachrániť nemalú sumu. Algoritmus analýzy a vyhodnotenia však musí byť presný, pretože majiteľ kreditnej karty sa môže začať odrazu správať „nejako divne" aj z iných dôvodov, napríklad sa ožení.

Podľa komplexnosti spracovania môžeme Data Mining rozdeliť do troch kategórií.
- Jednoduché výpisy
- Grafické zobrazenie
- Vizualizácia údajov

Jednoduché výpisy

Najjednoduchší prostriedok pre Data Mining, príkaz SELECT pozná určite každý. Výsledkom vyhľadávania môže byť jeden záznam, prípadne množina záznamov. Údaje sa zobrazia vo forme textového výpisu, alebo formou tabuľky. Do tejto skupiny môžeme zaradiť aj jednoduché analýzy.

Grafické zobrazenie

V tomto prípade sa údaje zobrazia formou grafu, ktorý môže byť podľa geometrie zobrazenia dvojrozmerný (2D) alebo trojrozmerný (3D). Podľa typu použitých grafických zobrazovacích prostriedkov môžeme navrhnúť graf stĺpcový, koláčový...

Vizualizácia údajov

Prakticky každá množina údajov poskytuje informácie k popisu nejakého objektu, javu, zákonitosti, časovej závislosti a podobne. Na základe poznania takýchto zákonitostí môžeme zdokonaliť grafické zobrazenie výsledkov analýzy. Napríklad pre zobrazenie výsledkov sčítania ľudu použijeme geografické zobrazenie vo forme mapy. Údaje o miere nezamestnanosti v jednotlivých krajoch môžeme prehľadne zobraziť v trojrozmernom obraze mapy. Oblasti, ktorá predstavuje plochu kraja priradíme tretí rozmer, výšku. Pri správne zvolenej axonometrii pohľadu získame prehľadné a aj pre laika ľahko pochopiteľné zobrazenie výsledku analýzy. Na základe prehľadného trojrozmerného zobrazenia výsledkov analýzy je možné prijať rýchle a kvalitné rozhodnutia. Preto je Data Mining neoceniteľný nástroj hlavne v oblasti marketingu, kde na základe údajov o predaji našich výrobkov a cieľových skupinách zákazníkov môžeme účinne plánovať reklamnú a marketingovú stratégiu

Príklad dataminingu z reálnej praxe - databáza Mushrooms

Na rozdiel od štruktúry cvičnej databázy FoodMart dodávanej s SQL Serverom 2000, kedy sme pre pochopenie jej štruktúry potrebovali aspoň nejaké „ekonomické minimum" (bola to databáza potravinárskeho obchodného reťazca, čiže ekonomicko - marketingová záležitosť), pochopenie štruktúry databázy Mushrooms, ktorá obsahuje databázu húb nebude pravdepodobne nikomu robiť problémy. Aj v tomto prípade chceme spoznať štruktúru databázy. Podobne ako v kapitole OLAP môžeme pre tento účel použiť program MS Visio 2002 Postup reverznej analýzy je v prílohe č. 2 na konci predchádzajúcej kapitoly.

tblMushroomsCases	
PK	**id**
	Pozivatelnost TvarKlobouku PovrchKlobouku BarvaKlobouku OxidacniZmenaBarvy Zapach NasazeniZebrovani VzdalenostZebrovani VelikostZebrovani BarvaZebrovani TvarNozky ZakonceniNozky PovrchNozkyNadPrstynkem PovrchNozkyPodPrstynkem BarvaNozkyNadPrstynkem BarvaNozkyPodPrstynkem TypZavoje BarvaZavoje PocetPrstynku TypPrstynku BarvaVytrusu Populace Prostredi

Obrázok 4.1 -tabuľka charakteristických vlastností húb

Tabuľka ***tblMushroomsCases***, ako vyplýva z názvu, obsahuje údaje o hubách. Pre názornosť si niekoľko údajov z tejto tabuľky vypíšeme. Pretože má veľa stĺpcov, rozdelili sme výpis do niekoľkých blokov.

id	Pozivatelnost	TvarKlobouku	PovrchKlob	BarvaKlob	Ox.ZmenaBarvy	Zapach
1	nejedlá, jedovatá	vypouklý	hladký	hnědá	ano	štiplavý, čpavý
3	jedlá	zvonovitý	hladký	bílá	ano	anýzový
5	jedlá	vypouklý	hladký	šedá	ne	žádný
7	jedlá	zvonovitý	hladký	bílá	ano	po mandlích
9	nejedlá, jedovatá	vypouklý	šupinatý	bílá	ano	štiplavý, čpavý
11	jedlá	vypouklý	šupinatý	žlutá	ano	anýzový
13	jedlá	zvonovitý	hladký	žlutá	ano	po mandlích
15	jedlá	vypouklý	vláknitý	hnědá	ne	žádný
17	jedlá	plochý	vláknitý	bílá	ne	žádný
19	nejedlá, jedovatá	vypouklý	šupinatý	bílá	ano	štiplavý, čpavý
21	jedlá	zvonovitý	hladký	žlutá	ano	po mandlích

... vodorovné pokračovanie tabuľky...

NasazeniZebrovani	VzdalenostZebrovani	VelikostZebr	BarvaZebr	TvarNozky
volné	těsné	úzké	černá	rozšiřující se dolů
volné	těsné	široké	hnědá	rozšiřující se dolů
volné	velmi těsné	široké	černá	zužující se dolů
volné	těsné	široké	šedá	rozšiřující se dolů
volné	těsné	úzké	růžová	rozšiřující se dolů
volné	těsné	široké	šedá	rozšiřující se dolů
volné	těsné	široké	bílá	rozšiřující se dolů
volné	velmi těsné	široké	hnědá	zužující se dolů
volné	velmi těsné	široké	černá	zužující se dolů
volné	těsné	úzké	hnědá	rozšiřující se dolů
volné	těsné	široké	černá	rozšiřující se dolů

...vodorovné pokračovanie tabuľky...

ZakonceniNozky	PovrchNadPrstynkem	PovrchNozkyPod	BarvaNozkyNad	BarvaNozkyPod	TypZavoje
stejný, rovný	hladký	hladký	bílá	bílá	částečný
hůl, pálka	hladký	hladký	bílá	bílá	částečný
stejný, rovný	hladký	hladký	bílá	bílá	částečný
hůl, pálka	hladký	hladký	bílá	bílá	částečný
stejný, rovný	hladký	hladký	bílá	bílá	částečný
hůl, pálka	hladký	hladký	bílá	bílá	částečný
hůl, pálka	hladký	hladký	bílá	bílá	částečný
stejný, rovný	hladký	vláknitý	bílá	bílá	částečný
stejný, rovný	hladký	hladký	bílá	bílá	částečný
stejný, rovný	hladký	hladký	bílá	bílá	částečný
hůl, pálka	hladký	hladký	bílá	bílá	částečný

...vodorovné pokračovanie tabuľky...

BarvaZavoje	PocetPrstynku	TypPrstynku	BarvaVytrusu	Populace	Prostredi
bílá	jeden	přívěskovitý	černá	roztroušená	město
bílá	jeden	přívěskovitý	hnědá	početná	louka
bílá	jeden	rozplývající se	hnědá	velmi početná	tráva
bílá	jeden	přívěskovitý	černá	početná	louka
bílá	jeden	přívěskovitý	černá	několik hub	tráva
bílá	jeden	přívěskovitý	hnědá	početná	tráva
bílá	jeden	přívěskovitý	hnědá	roztroušená	tráva
bílá	jeden	rozplývající se	černá	velmi početná	tráva
bílá	jeden	rozplývající se	hnědá	velmi početná	tráva
bílá	jeden	přívěskovitý	hnědá	roztroušená	město
bílá	jeden	přívěskovitý	hnědá	roztroušená	louka

Tabuľka ***tblMushroomsCases*** obsahuje 4062 záznamov. Niekoľko tisíc údajov je už pomerne slušný balík na „vydolovanie„ súvislostí a v niektorých aplikáciách aj na predpovedanie budúceho trendu.

Vytvorenie modelu pre Data Mining

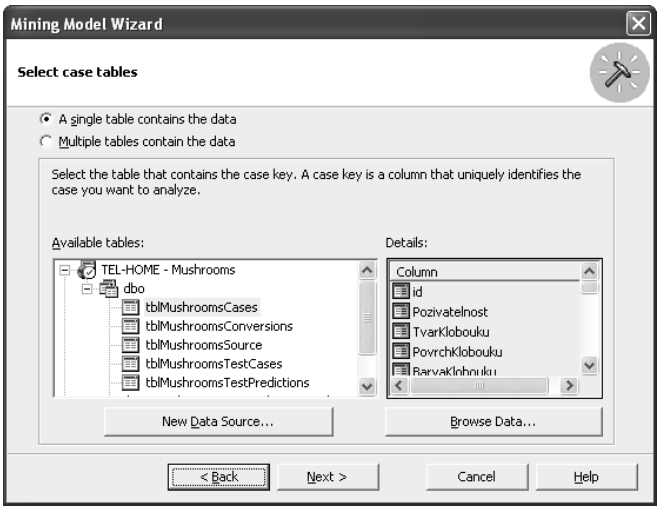
Prikročme pre ilustráciu k vlastnému dataminingu. Pomocou nástroja **Analysis Manager** (súčasť dodávky MS SQL Servera 2000) vytvorme nad údajmi v databáze „mining model" s názvom napríklad *JedleHuby*.

Obrázok 4.2 - Mining Model Wizard

Ťažiť údaje môžeme jednak z relačných databáz, ale aj z multidimenzionálnych dátových štruktúr OLAP. Preto sa od nás požaduje špecifikovať typ zdroja údajov. Naša databáza húb je samozrejme relačná, takže voľba v nasledujúcom dialógovom okne bude jednoznačná:

Obrázok 4.3 - Výber typu zdroja údajov pre DM

Nasleduje výber tabuľky, alebo viacerých tabuliek, ktoré sa stanú podkladom pre proces dataminingu.



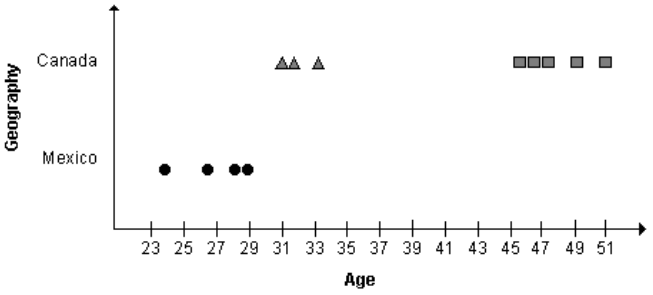
Obrázok 4.4 - Výber tabuliek

V nasledujúcom kroku je potrebné vybrať si typ algoritmu pre analýzu údajov a predikciu. Microsoft v analytických službách QL Serveru 2000 ponúka dve možnosti"

– **Microsott clustering** (Viacrozmerné zhlukové diagramy) algoritmus sa používa pre odhaľovanie zhlukov dát v multidimenzionálnych priestoroch

Ako príklad uvedieme výsledky analýzy, do ktorých susedných krajín najčastejšie cestujú ľudia z USA v určitých vekových kategóriach.

Obrázok 4.5 - príklad zhlukového diagramu

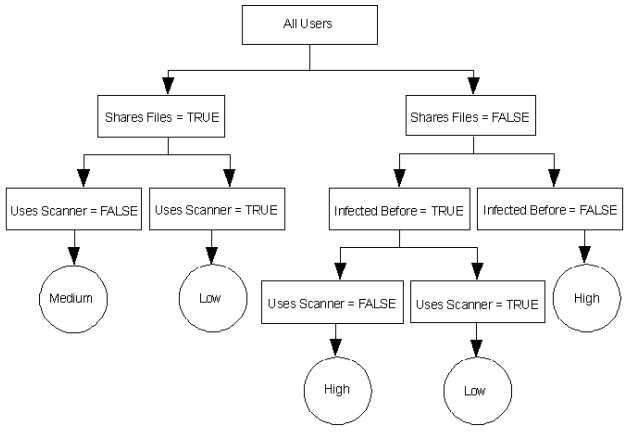


Vidíme, že do Mexika cestujú najviac ľudia vo veku od 23 do 29 rokov a do Kanady v dvoch vekových intervaloch 31-34 a 45-51 rokov.

– **Microsoft decision Trees** (Nevyvážený rozpadový strom) odhaľuje závislosti a vyhľadáva špecifické vlastnosti, ktoré potom poslúžia pre predikciu

Ako príklad takéhoto nevyváženého rozpadového stromu môžeme uviesť diagram závislosti, ktorý predikuje riziko zavlečenia počítačového vírusu do počítača v závislosti od toho, či využívame alebo nevyužívame zdieľané súbory a či používame, alebo nepoužívame antivírový program

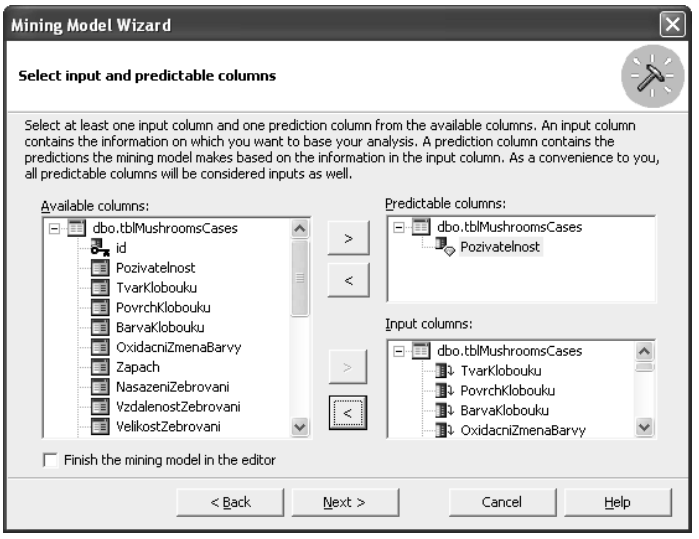
Obrázok 4.6 - Nevyvážený rozpadový strom



Skúsení hubári by určite dokázali takýto diagram zostaviť aj pre náš príklad, len **na základe svojich skúseností**. A sme vlastne doma. Hubár získal svoje poznatky na základe dlhoročných skúseností, preto sa na jeho odhad môžeme s vysokou mierou istoty spoľahnúť. Zaváňa to možno aj troškou čierneho humoru, ale táto miera istoty je o to väčšia, že hubári, ktorí sa dopustili závažného omylu už medzi nami nie sú. SQL Server 2000 na huby nechodí, a teda žiadne poznatky o nich nemá (rovnako ako nemá poznatky o vývoji dámskej módy, počasia a podobne). Na základe zaznamenaných údajov však tieto poznatky môže získať.

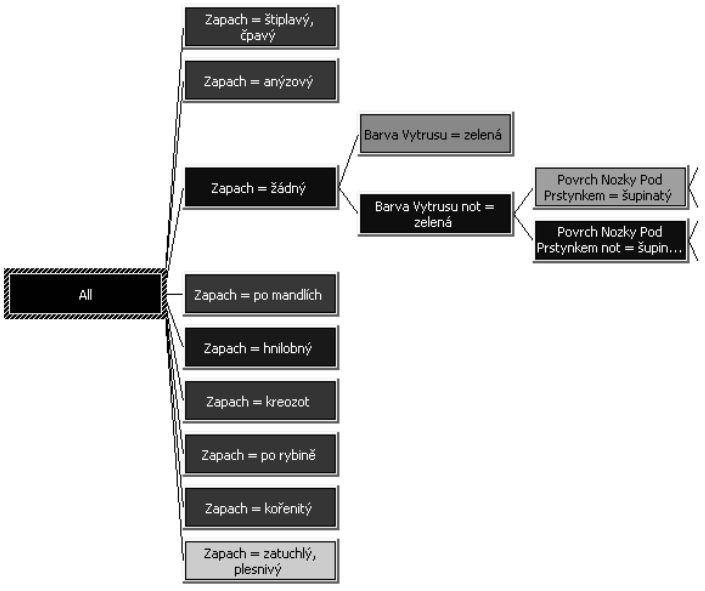
V nasledujúcom jednoduchom dialógu je potrebné vybrať kľúčový stĺpec (unikátny identifikátor pre každý záznam), v našom prípade to bude ID. Pokračujeme ďalším dialógom, najdôležitejším, v ktorom špecifikujeme stĺpce, ktoré chceme predikovať a stĺpce, na základe hodnôt ktorých chceme predikciu uskutočniť. V našom príklade budeme predikovať požívateľnosť na základe všetkých ostatných znakov.

Obrázok 4.7 - výber vstupných a predikovateľných stĺpcov.



Výsledkom bude prehľadný diagram pre všetky hodnoty stĺpca požívatelnosť, teda pre jedlé aj jedovaté huby. Jedným slovom pre všetky druhy húb. Čím je farba políčka tmavšia, tým je pravdepodobnosť výskytu tejto vlastnosti vyššia.

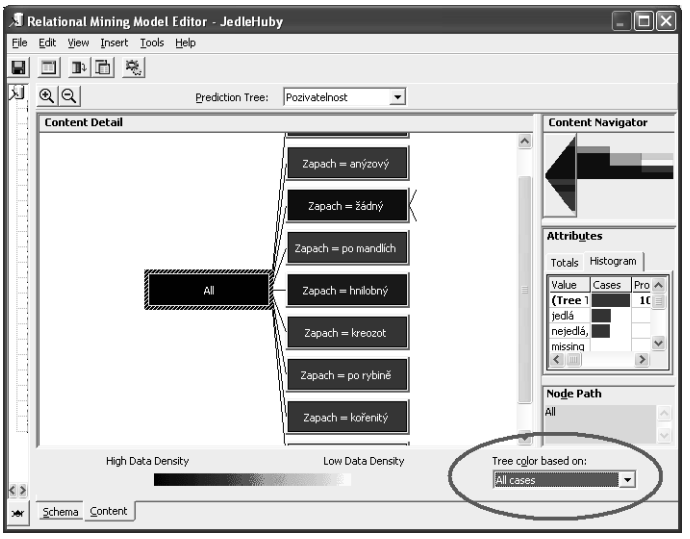
Obrázok 4.8 - odhalené závislosti pre všetky huby.



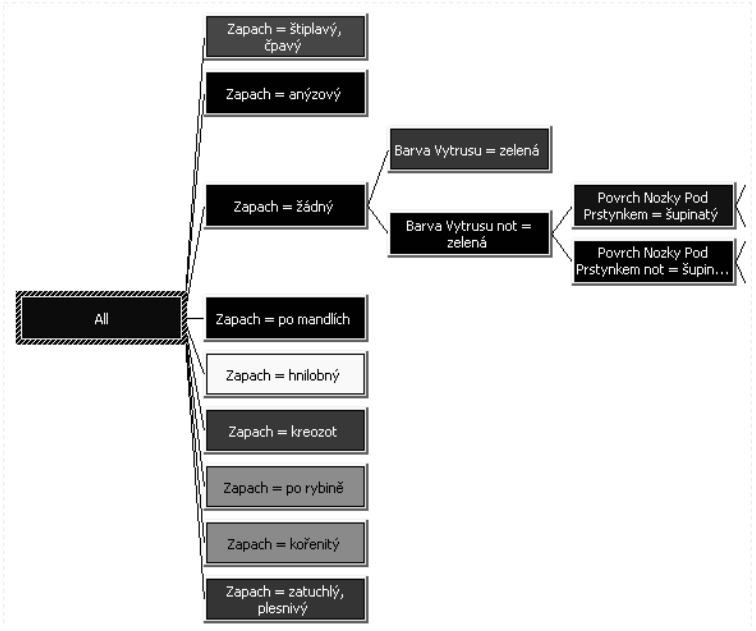
Najčastejšie sa teda na základe našich údajov v prírode vyskytujú huby, ktoré pod prstienkom nie sú šupinaté, nemajú farbu výtrusu zelenú a nijako nezapáchajú.

Na základe tohto diagramu samozrejme nemôžeme určiť vlastnosti pre huby jedlé a nejedlé. Všimnime si v ľavej dolnej časti okna aplikácie **Relational Mining Model editor** ovládací prvok s názvom **Tree color based on**. Pomocou tohto prvku môžeme nastaviť, k akej hodnote stĺpca požívateľnosť (jedlé, nejedlé) sa budú intenzity farieb v diagrame vzťahovať.

Obrázok 4.9 - Relational Mining model editor.



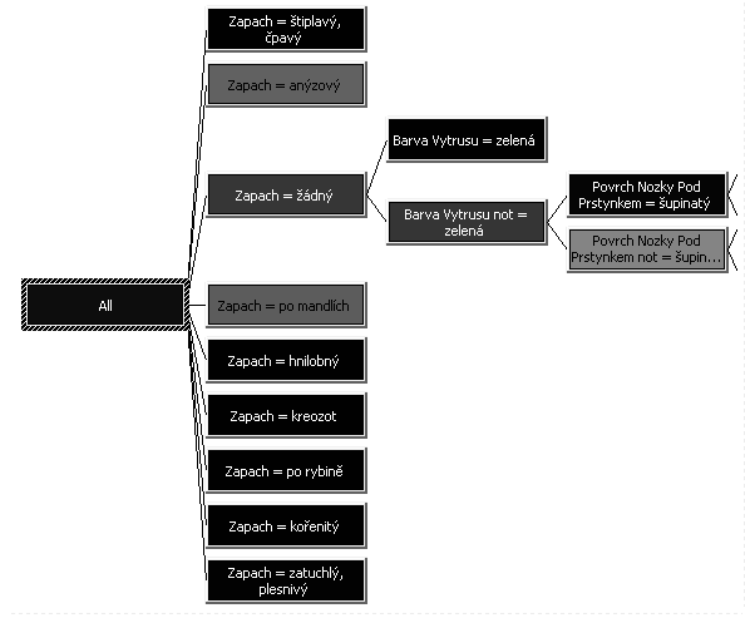
Pozrime sa podrobnejšie na diagram pre jedlé huby



Obrázok 4.10 - diagram pre jedlé huby.

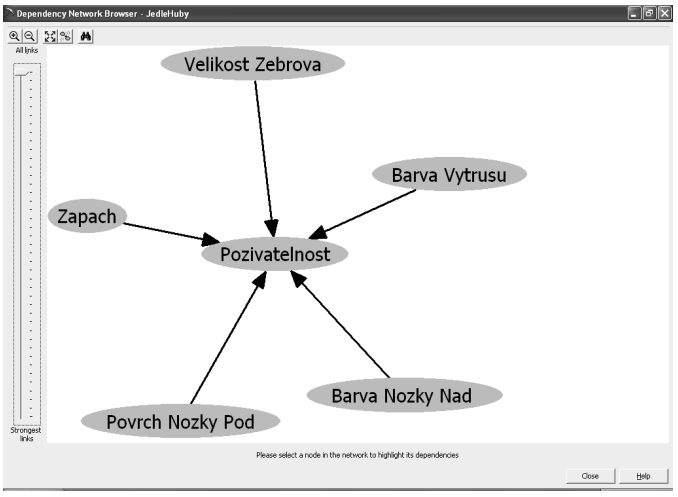
Jedným z charakteristických znakov jedlých húb je, že nemajú zelený výtrus, ale hlavným kritériom pre určenie konzumovateľnosti je ich zápach. Môžeme teda akceptovať huby bez zápachu, alebo také, čo sú cítiť po aníze, alebo po mandliach. A naopak vidíme, že huby, ktoré majú hnilobný zápach sa z najväčšou pravdepodobnosťou dajú zjesť len raz.

Keďže stĺpec poživateľnosť obsahuje len dve hodnoty, diagram pre nejedlé a jedovaté huby bude akýmsi približným negatívom húb jedlých. Ak má huba zelený výtrus, prípadne má štipľavý zápach, hnilobný zápach, alebo razí po kreozote (ropný produkt), rybách, prípadne má korenistý, alebo zatuchlý, plesnivý zápach, na konzumáciu sa určite nehodí.



Obrázok 4.11 - Diagram pre jedovaté huby.

Vo všeobecnosti sme odhalili, že konzumovateľnosť húb, teda či sú jedlé, prípadne nejedlé alebo jedovaté závisí v prvom rade od ich zápachu. Potvrdí nám to aj nástroj **Dependency Network Browser.**



Obrázok 4.12 - Dependency Network Browser.

Práca s týmto nástrojom je veľmi jednoduchá. Po jeho spustení sa vo forme prehľadného diagramu zobrazí od čoho a v akej miere závisí predikovateľná veličina. V našom prípade závisí požívateľnosť húb od týchto vlastností (zhora v smere hodinových ručičiek)

- veľkosť rebrovania
- farba výtrusu
- farba nožičky nad prstienkom
- povrch nožičky pod prstienkom
- zápach

Zatiaľ nám chýba informácia, ktorá vlastnosť najviac vplýva na konzumovateľnosť húb. Všimnime si však posuvný ovládací prvok v ľavej časti okna aplikácie. Ak ho budeme posúvať smerom nadol, postupne nám odpadnú menej významné vplyvy. Poradie vlastností, ktoré vplývajú na jedlosť húb bude teda v poradí od najcharakteristickejšej po najmenej charakteristickú vlastnosť nasledovné:

- **zápach**
- **farba výtrusu**
- **povrch nožičky pod prstienkom**
- **veľkosť rebrovania**
- **farba nožičky nad prstienkom**

Tento príklad je veľmi vhodný pre prvé pokusy s dataminingom. Súvislosti totiž odhalíme len tam, kde sa naozaj nejaké nachádzajú. Preto sa pre datamining hodia práve takéto príklady z reálneho života. Z umelo vytvorených tabuliek toho pravdepodobne veľa nevydolujeme.

Analógia cvičného príkladu s inými oblasťami - napríklad stanovenie úverového rizika.

Niekto by sa mohol opýtať: A načo je to dobré? Ľudia, ktorí na tému „otrava hubami" už ležali v nemocnici by určite vedeli. Prienik množín hubárov a analytikov používajúcich SQL Server asi nebude veľký. SQL Server, a hlavne jeho analytické služby skutočne viac používajú finanční a marketingoví odborníci a iní špecialisti prevažne z tejto brandže. Náš príklad s určením konzumovateľnosti húb má veľmi dobrú analógiu k iným veličinám, ktoré na základe doteraz nazbieraných údajov potrebujeme predikovať. Je to napríklad stanovenie úverového rizika pri poskytovaní úveru fyzickým osobám. Podobne ako o hubách s ktorými sme doteraz mali skúsenosti sme dokázali zhromaždiť rôzne údaje, napríklad:

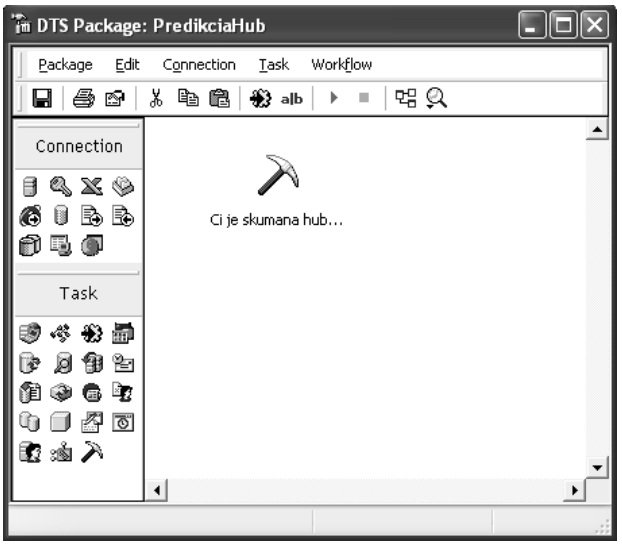
Tvar Klobouku,
Povrch Klobouku,
Barva Klobouku,
Oxidacni Zmena Barvy,
Zapach
,Nasazeni Zebrovani,
Vzdalenost Zebrovani,
Velikost Zebrovani,
Tvar Nozky,
Zakonceni Nozky,
Povrch Nozky Pod Prstynkem,
Barva Nozky Nad Prstynkem,
Barva Nozky Pod Prstynkem,
Typ Zavoje,
Barva Zavoje,
Pocet Prstynku,
Typ Prstynku,
Barva Vytrusu,
Populace,
Prostredi,
Pozivatelnost

pričom nás v konečnom dôsledku ako správnych konzumentov zaujíma hlavne konzumovateľnosť, aj o zákazníkoch ktorým sme doteraz poskytli úver, alebo kreditnú kartu máme zhromaždené pomerne veľa údajov, len stačí nájsť súvislosti medzi ostatnými vlastnosťami a mierou úverového rizika.

Dokonca by sme sa mohli s určitou dávkou humorného nadhľadu zamyslieť aj nad analógiou výsledkov analýzy konzumovateľnosti húb a mierou úverového rizika. U húb bol určujúcim kritériom zápach. Asi je každému jasné, aká je miera úverového rizika u zákazníka, ktorý vonia drahou kolínskou a u zákazníka, z ktorého je dominantne cítiť jablčné víno ☺ .

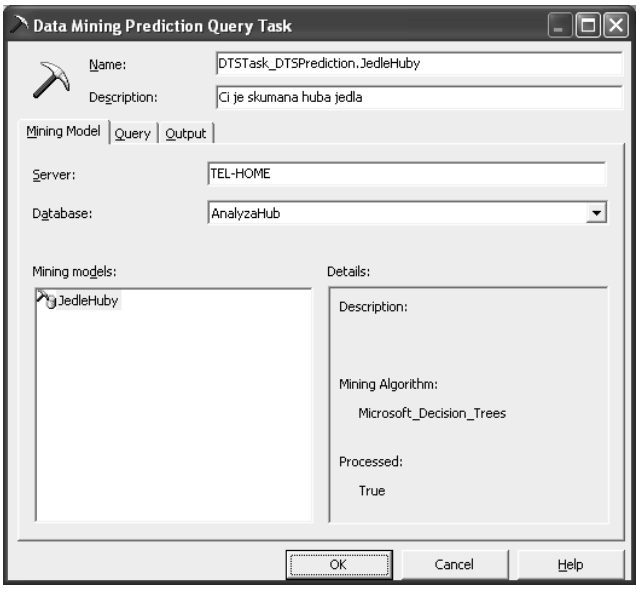
Príklad predikcie na základe výsledkov analýzy

Na záver príkladu si ukážeme ešte príklad predikcie nad databázou húb. Táto databáza totiž obsahuje dve tabuľky, ktoré vznikli rozdelením jednej väčšej tabuľky na dve časti. Tabuľku ***tblMushroomsCases*** už poznáme, nad ňou sme robili analýzu. Predikciu vykonáme nad tabuľkou ***tblMushroomsTestCases***, ktorá má presne rovnakú štruktúru a obsahuje druhú polovicu údajov z pôvodnej tabuľky. Algoritmus pre rozdelenie tabuľky bol veľmi jednoduchý. Záznamy s párnym ID poputovali do jednej tabuľky a záznamy s nepárnym ID do druhej. Ako nástroj použijeme **Data Mining Prediction Quey Task**, čo je vlastne job v DTS. Ak si pozrieme obrázok aplikácie **DTS Package**, je to posledná ikona v okne tasks (symbol murárskeho kladiva)



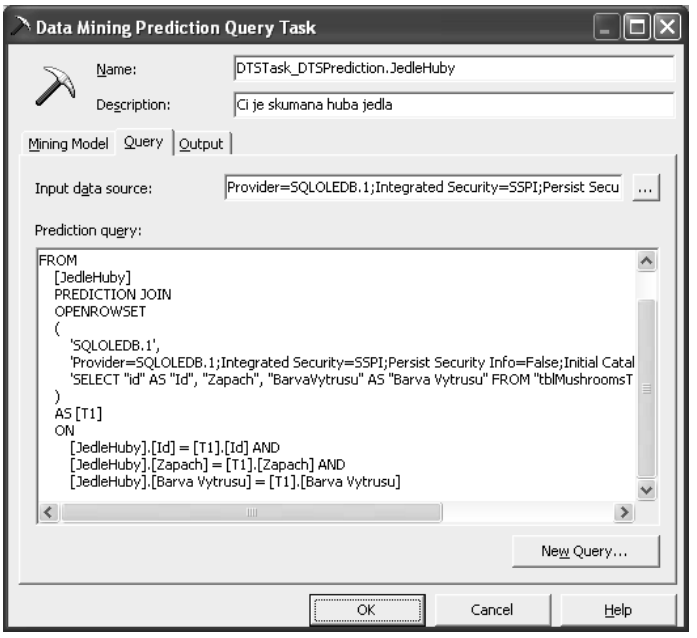
Obrázok 4.13 - DTS Package.

Obrázok 4.14 - Prediction Query Builder.



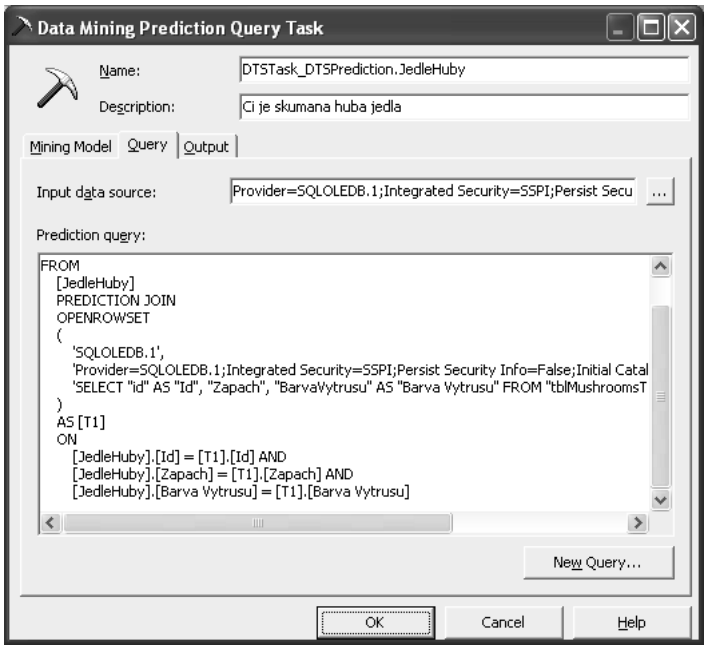
Parametre pre tento nástroj nastavíme v troch záložkách. V prvej z nich sa pripojíme na príslušnú databázu analytického servera. V druhej záložke špecifikujeme, ktorú veličinu predikujeme na základe vstupných veličín.

Obrázok 4.15 - Prediction Query Builder.



Nezľaknime sa pomerne zložitého príkazu v tomto okne:

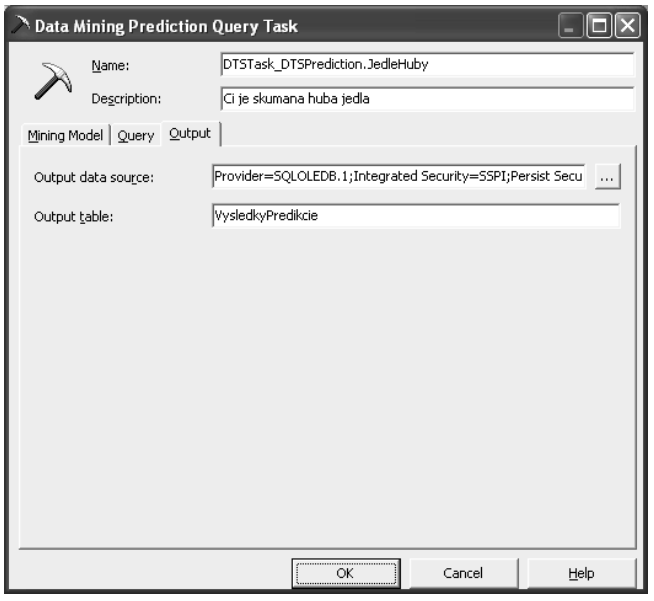
```
SELECT FLATTENED
    [T1].[Id], [T1].[Zapach], [T1].[Barva Vytrusu], [JedleHuby].[Pozivatelnost]
FROM
    [JedleHuby]
    PREDICTION JOIN
    OPENROWSET
    (
        'SQLOLEDB.1',
        'Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial
Catalog=Mushrooms;Data Source=TEL-HOME',
        'SELECT "id" AS "Id", "Zapach", "BarvaVytrusu" AS "Barva Vytrusu" FROM "tblMushro-
omsTestCases" ORDER BY "id"'
    )
    AS [T1]
    ON
        [JedleHuby].[Id] = [T1].[Id] AND
        [JedleHuby].[Zapach] = [T1].[Zapach] AND
        [JedleHuby].[Barva Vytrusu] = [T1].[Barva Vytrusu]
```



nemusíme ho totiž písať sami, za tlačidlom New Query sa skrýva sprievodca s názvom **Prediction Query Builder**. Zadáme tabuľku, ktorú budeme na základe vybraného modelu skúmať ***tblMushroomsTestCases***, vyberieme príslušné stĺpce, a zvyšok urobí tento nástroj sám. Vieme, že hlavný vplyv na konzumovateľnosť húb má ich zápach a farba výtrusu, preto budeme predikovať konzumovateľnosť húb z druhej tabuľky len na základe týchto dvoch vlastností.

Obrázok 4.16 - Prediction Query Builder.

V poslednej záložke určíme, kam sa uložia výsledky predikcie. V našom príklade sme zvolili tabuľku **VysledkyPredikcie** v tej istej databáze.



Obrázok 4.17 - výsledná tabuľka.

Výsledkom bude jednoduchá tabuľka:

```
T1_Id       T1_Zapach        T1_Barva Vytrusu   JedleHuby_Pozivatelnost
----------- ---------------- ----------------- ----------------------
2           po mandlích      hnedá              jedlá
4           štiplavý, cpavý  cerná              nejedlá, jedovatá
6           po mandlích      cerná              jedlá
8           anýzový          hnedá              jedlá
10          po mandlích      cerná              jedlá
12          po mandlích      cerná              jedlá
14          štiplavý, cpavý  hnedá              nejedlá, jedovatá
16          žádný            hnedá              jedlá
18          štiplavý, cpavý  cerná              nejedlá, jedovatá
...
```

Výsledky predikcie sú v poslednom stĺpci `JedleHuby_Pozivatelnost`.

Logicky nás zaujíma, nakoľko je táto predikcia úspešná. V tomto prípade to môžeme jednoducho zistiť. V stĺpci **Pozivatelnost** máme totiž uloženú informáciu o tom, či daná huba v skutočnosti je, alebo nie je konzumovateľná. Vytvoríme preto novú tabuľku `PorovnaniePredikcie`, ktorá bude mať len tri stĺpce:

- ID
- Predikciu (teda predpoveď vykonanú pomocou Data Mining Prediction Task)
- Realitu (skutočný údaj o tom či daná huba je, alebo nie je konzumovateľná)

```
CREATE TABLE PorovnaniePredikcie
(
    ID INT,
    Predikcia VARCHAR (255),
    Realita VARCHAR (255)
);
```

Tabuľku naplníme "zosypaním" údajov z obidvoch tabuliek:

```
INSERT INTO PorovnaniePredikcie (ID, Predikcia, Realita)
       SELECT   VysledkyPredikcie.T1_Id,
                VysledkyPredikcie.JedleHuby_Pozivatelnost,
                tblMushroomsTestCases.Pozivatelnost
       FROM tblMushroomsTestCases, VysledkyPredikcie
       WHERE VysledkyPredikcie.T1_Id = tblMushroomsTestCases.ID;
```

Výsledná tabuľka bude potom obsahovať údaje v takomto tvare:

```
ID          Predikcia                Realita
----------- ------------------------ ------------------------
2           jedlá                    jedlá
4           nejedlá, jedovatá        nejedlá, jedovatá
6           jedlá                    jedlá
8           jedlá                    jedlá
10          jedlá                    jedlá
12          jedlá                    jedlá
14          nejedlá, jedovatá        nejedlá, jedovatá
16          jedlá                    jedlá
18          nejedlá, jedovatá        nejedlá, jedovatá
20          nejedlá, jedovatá        nejedlá, jedovatá
```

V prvých desiatich prípadoch sa teda náš predikčný mechanizmus nepomýlil ani raz. Samozrejme, že nás zaujíma ako to dopadlo v celej tabuľke, ktorá obsahuje 4062 záznamov.

Zistiť to nebude žiadny problém, a aspoň si trochu zopakujeme SQL príkazy. Napíšeme jednoduchý skript:

```
SELECT count(*) AS [Celkovy pocet] FROM PorovnaniePredikcie;

SELECT count(*) AS [Chybna predikcia] FROM PorovnaniePredikcie
WHERE Predikcia <> Realita;
```

Výsledok potvrdzuje takmer úplnú presnosť odhadov

```
Celkovy pocet
-------------
4062

Chybna predikcia
----------------
25
```

Percentuálne to znamená (v článku o SQL serveri predsa nebudeme počítať percentá na kalkulačke)

```
SELECT 100* CONVERT(float, (SELECT COUNT(*) FROM PorovnaniePredikcie
                            WHERE predikcia <>Realita)) /
            CONVERT(float, COUNT(*)) AS Percento
FROM    PorovnaniePredikcie;

Percento
------------------
0.6154603643525357
```

Vidíme, že z celkového počtu 4062 húb bolo nesprávne určených 25, čo je len zanedbateľných **0.62%**. V tomto prípade nie je však omyl ako omyl. Ak necháme jedlú hubu, ktorú algoritmus nesprávne určil ako jedovatú v lese, nestane sa nič. Opačný prípad, kedy algoritmus určí jedovatú hubu ako jedlú však môže mať tragické následky. Overme si teda, aký je pomer spomínaných prípadov, ktoré môžu nastať pri nesprávnom určení huby:

```
SELECT count(*) AS [Jedovata urcena ako jedla] FROM PorovnaniePredikcie
WHERE Predikcia ='jedlá' AND Predikcia <> Realita;

SELECT count(*) AS [Jedla urcena ako jedovata] FROM PorovnaniePredikcie
WHERE Predikcia ='nejedlá, jedovatá' AND Predikcia <> Realita;
```

```
Výsledok nás v tomto prípade jemne zamrazí.

Jedovata urcena ako jedla
-------------------------
25
Jedla urcena ako jedovata
-------------------------
0
```

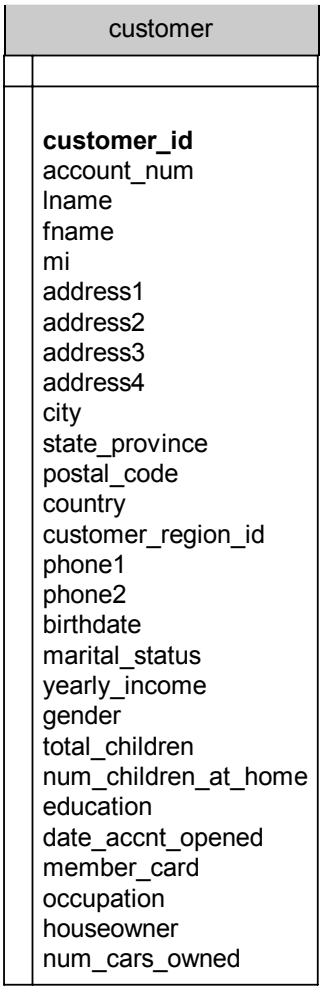
Potvrdzuje to platnosť Murphyho zákonov. (Ak sa už dá niečo pokaziť, tak sa to urobí dôsledne). Určité riziko 0.62% tu zostáva. Spomeňme si, že náš model nebol až taký presný, ako by teoreticky mohol byť. Na základe diagramu, ktorý bol výsledkom dataminingu sme vedeli, že hlavný vplyv na konzumovateľnosť húb má ich zápach a farba výtrusu, preto sme predikovali len na základe týchto dvoch vlastností. Ak budeme predikovať na základe všetkých vlastností, získame len 9 nesprávne určených húb, čo predstavuje **0.22%**. Presnosť sa teda zvýšila trojnásobne, ale ak by sme brali do úvahy absolútne čísla, bol aj predošlý model úplne vyhovujúci. Bohužiaľ aj teraz sú nesprávne určené huby tým horším spôsobom, teda jedovaté sú určené ako jedlé.

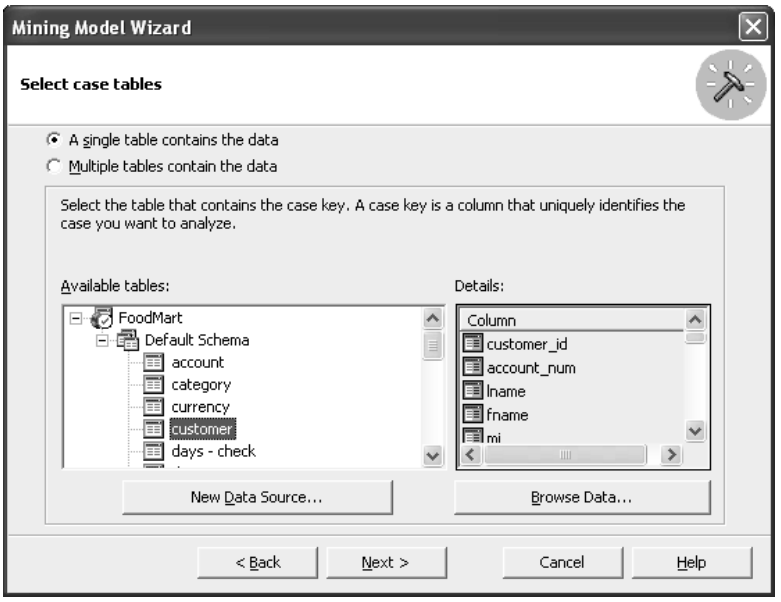
Príklad dataminingu z cvičnej databázy FoodMart

Predchádzajúci príklad bol veľmi vďačný z dôvodu jednoduchosti a názornosti, ovšem zohnať takúto vynikajúcu databázu je niekedy problém, takže sa pokúsime predsa len ukázať aj analýzu údajov z cvičnej databázy (v našom prípade databázy FoodMart, ktorá sa dodáva z MS SQL Serverom 2000) Ukážeme si výsledok analýzy, ktorá skúma, ako závisí druh kreditnej karty (zlatá, strieborná...) od ročného príjmu jej majiteľa, počtu detí, stavu (ženatý, slobodný) a podobne. Postup už poznáme, preto len stručne v bodoch:

– vyberieme ako typ zdroja relačnú databázu a ako zdroj údajov tabuľku **Customer**

Obrázok 4.18 - Tabuľka Customer



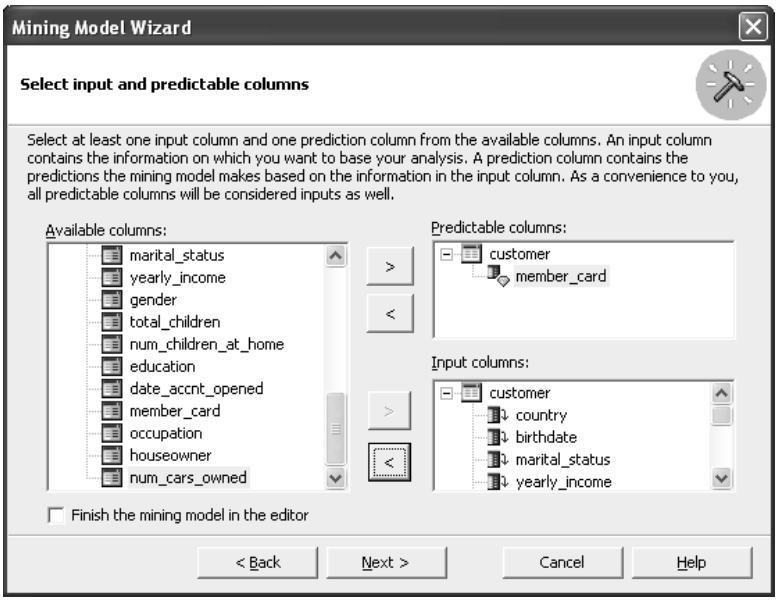


Obrázok 4.19 - Výber tabuliek

Vyberieme typ algoritmu **Microsoft decision Trees** (Nevyvážený rozpadový strom) a unikátny identifikátor každého záznamu **customer_id**.

Predikujeme stĺpec **[member_card]**. Tabuľka sa skladá z pomerne veľa stĺpcov, pričom vieme, že veľa z nich na to, akú úverovú (zákaznícku) kartu zákazník vlastní veľký vplyv nemá. Ak tieto stĺpce vynecháme, zostanú nám tieto:

```
[country]
[birthdate]
[marital_status]
[yearly_income]
[gender]
[total_children]
[num_children_at_home]
[education]
[occupation]
[houseowner]
[num_cars_owned]
```



Obrázok 4.20 - Výber vstupných a predikovateľných stĺpcov

Výsledkom bude aj v tomto prípade prehľadný diagram pre všetky hodnoty stĺpca **member_card**, teda pre zlaté, strieborné, bronzové, normálne karty a aj pre prípad, že zákazník kartu nevlastní Čím je farba políčka tmavšia, tým je pravdepodobnosť výskytu tejto vlastnosti u držiteľa príslušnej karty vyššia. Všimnime si aj percentuálne rozloženie jednotlivých druhov kariet medzi zákazníkmi.

Obrázok 4.21 - percentuálne zastúpenie jednotlivých druhov kariet medzi zákazníkmi

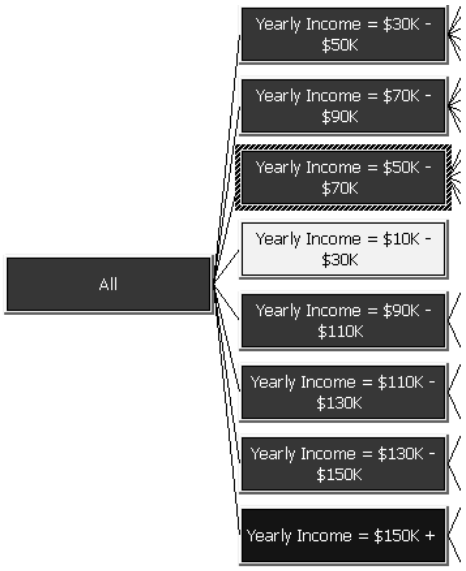
Attributes

Totals | Histogram

Value	Cases	Probability
(Node Total)		**100.00%**
Bronze		72,00%
Golden		13,14%
Normal		4,38%
Silver		10,43%
missing		0,05%

Zlaté karty

Najviac majiteľov zlatých kariet sa grupuje zo skupiny zákazníkov s ročným príjmom nad 150 tisíc USD. Zákazníci (okrem pár výnimiek, ktorí možno niečo zdedili) s príjmom 10 - 30 tisíc USD na vydanie takejto karty pravdepodobne nemajú ani nárok.



Obrázok 4.22 - Diagram pre zlaté karty

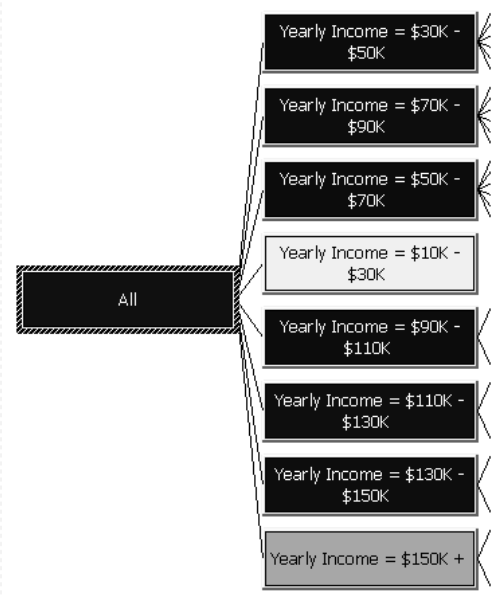
Strieborné karty

V tomto segmente je situácia prakticky rovnaká ako u zlatých kariet. Najviac majiteľov strieborných kariet sa aj tu grupuje zo skupiny zákazníkov s ročným príjmom nad 150 tisíc USD. Zákazníci s príjmom 10 - 30 tisíc USD sú aj tu prakticky bez šance.

Bronzové karty

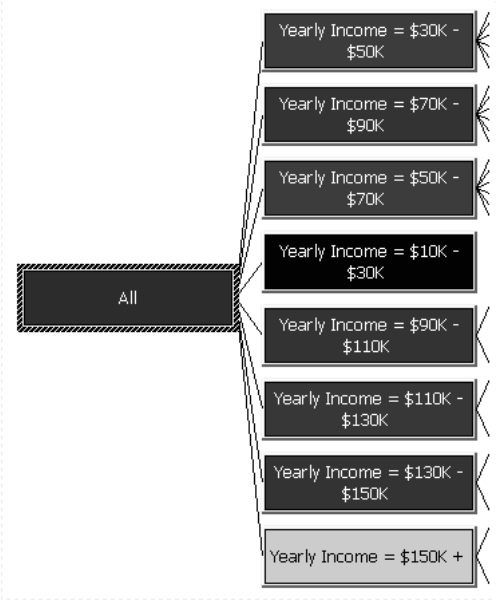
V tomto segmente je situácia úplne opačná. K slovu sa dostávajú ostatné príjmové skupiny s výnimkou zákazníkov s príjmom 10 - 30 tisíc USD. Zákazníci s ročným príjmom nad 150 tisíc USD na druhej strane o takéto karty už záujem nemajú.

Obrázok 4.23 - Diagram pre bronzové karty



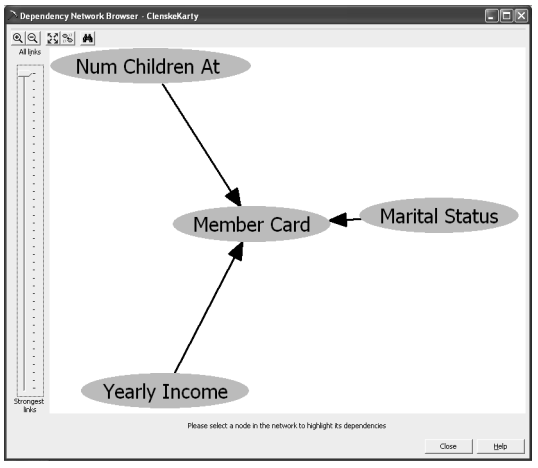
Štandardné karty

Tieto karty sú prakticky doménou zákazníkov s príjmom 10 - 30 tisíc USD.



Obrázok 4.24 - Diagram pre štandardné karty

Vyskúšajme ešte silu vplyvu jednotlivých atribútov pomocou **Dependency Network Browser**. To že na druh karty má najväčší vplyv ročný príjem sme určite vedeli, na to datamining nepotrebujeme, ale čo ostatné vplyvy?

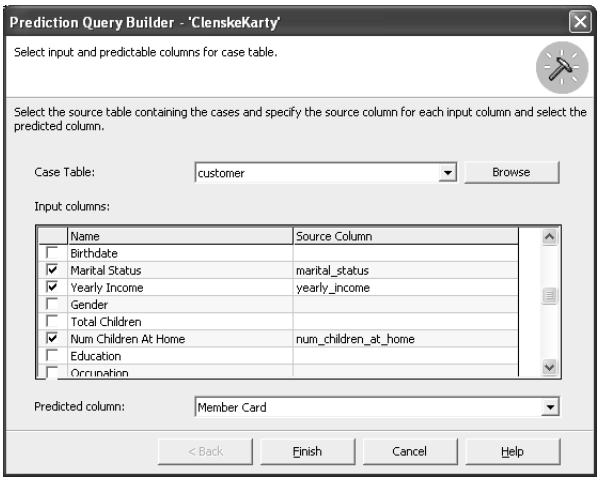


Obrázok 4.25 - vplyv atribútov

V našom prípade závisí druh kreditnej karty od ročného príjmu, počtu detí v domácnosti a od toho či je zákazník (zákazníčka) slobodný, alebo v „stave manželskom" (ženatí určite vedia o čom teraz hovoríme ☺).

Predikcia na základe výsledkov analýzy

Na záver príkladu si znova ukážeme predikciu. Aplikujeme ju na tabuľku customer (tú istú, ktorá bola predmetom analýzy) Ako nástroj použijeme **Data Mining Prediction Quey Task**, ako job v DTS.



Obrázok 4.26 - výber vstupných stĺpcov a predikovaného stĺpca

Výsledkom je pomocná tabuľka **[Vysledky predikcie]**.

```
T1_Customer Id T1_Marital Status  T1_Yearly Income  T1_Num Children At Home ClenskeKarty
-------------- ----------------- ----------------- ----------------------- -------------
1              M                 $30K - $50K       2                       Bronze
2              S                 $70K - $90K       0                       Bronze
3              M                 $50K - $70K       1                       Bronze
4              M                 $10K - $30K       4                       Normal
5              S                 $30K - $50K       0                       Bronze
6              S                 $70K - $90K       0                       Bronze
7              M                 $30K - $50K       1                       Bronze
8              M                 $50K - $70K       2                       Bronze
9              M                 $10K - $30K       3                       Normal
10             S                 $30K - $50K       0                       Bronze
```

Aj v tomto prípade nás bude zaujímať absolútna a aj percentuálna úspešnosť predikcie. Vytvoríme preto novú tabuľku **PorovnaniePredikcie**, ktorá bude mať tieto stĺpce:

- ID
- Predikciu (teda predpoveď vykonanú pomocou Data Mining Prediction Task)
- Realitu (skutočný údaj o tom akú kreditnú kartu má daný zákazník)

```
CREATE TABLE PorovnaniePredikcie
(
     ID INT,
     Predikcia VARCHAR (255),
     Realita VARCHAR (255)
);
```

Tabuľku naplníme „zosypaním" údajov z obidvoch tabuliek:

```
INSERT INTO PorovnaniePredikcie (ID, Predikcia, Realita)
       SELECT   [Vysledky predikcie].[T1_Customer Id],
                [Vysledky predikcie].[ClenskeKarty_Member Card],
                customer.[member_card]
       FROM customer, [Vysledky predikcie]
       WHERE [Vysledky predikcie].[T1_Customer Id] = customer.customer_ID;
```

Výsledná tabuľka bude potom obsahovať údaje v takomto tvare:

```
ID         Predikcia             Realita
---------- --------------------- ------------------
1          Bronze                Bronze
2          Bronze                Bronze
3          Bronze                Bronze
4          Normal                Normal
5          Bronze                Silver
6          Bronze                Bronze
7          Bronze                Bronze
8          Bronze                Bronze
9          Normal                Normal
10         Bronze                Golden
11         Bronze                Bronze
12         Bronze                Bronze
13         Bronze                Bronze
14         Bronze                Bronze
15         Bronze                Bronze
16         Bronze                Bronze
17         Bronze                Bronze
18         Bronze                Bronze
19         Bronze                Bronze
20         Normal                Normal
```

V prvých dvadsiatich prípadoch sa náš predikčný mechanizmus pomýlil dva krát. Nebude to teda takmer absolútna presnosť predikcie ako v predchádzajúcom príklade (však v tomto prípade nejde o život ale len o biznis). Samozrejme, že aj v tomto prípade nás zaujíma ako to dopadlo v celej tabuľke, ktorá obsahuje 10 281 záznamov.

```
SELECT count(*) AS [Celkovy pocet] FROM PorovnaniePredikcie;

SELECT count(*) AS [Chybna predikcia] FROM PorovnaniePredikcie
WHERE Predikcia <> Realita;

SELECT 100* CONVERT(float, (SELECT COUNT(*) FROM PorovnaniePredikcie
                            WHERE predikcia <>Realita)) /
           CONVERT(float, COUNT(*)) AS Percento
FROM    PorovnaniePredikcie;
```

Výsledok je v globále uspokojivý.

```
Celkovy pocet
-------------
10281


Chybna predikcia
----------------
1769


Percento
--------------------
17.206497422429724
```

Sedemnásť percent bolo určené chybne, ale **82.7 % bolo predikovaných správne**

Záver

Z doterajšieho prehľadu by sa zdalo, že na základe vhodných údajov môžeme skúmať a odhaľovať prakticky akúkoľvek, dokonca aj nezmyselnú závislosť, napríklad vplyv polárnej žiary na panenstvo. Je to pravda, všetko samozrejme závisí od správneho nastavenia parametrov modelu pre datamining.

Literatúra

[1] Lacko, L: Web a databáze, Computerpress 2001
[2] Lacko, L: Budovanie dátových skladov, PC REVUE -seriál 10, 11, 12 2000, 1.2001
[3] Lacko, L: Databázy pre každého, PC SPACE - 8/2001
[4] Chon S. Chua, Richard Green: Data Warehouse Method - štúdijná literatúra firmy ORACLE, Oracle University 1999
[5] Vieira, R: SQL Server 2000, Programujeme profesionálne, Computerpress 2001

WWW.microsoft.com
www.oracle.com

Vydal:
Microsoft, s. r. o., Novodvorská 1010/14B, 140 00 Praha 4, tel.: +420 (2) 61 19 71 11, fax: +420 (2) 60 19 71 00
http://www.microsoft.com/cze, http://www.microsoft.com/slovakia