EF SKRIPTA

Jurij JAKLIČ

UPRAVLJANJE IN UPORABA PODATKOV

UNIVERZA V LJUBLJANI
Ekonomska fakulteta

S4A734

4 125

UNIVERZA V LJUBLJANI
Založništvo Ekonomske fakultete

Jurij JAKLIČ

UPRAVLJANJE IN UPORABA PODATKOV

LJUBLJANA, 2002

544734

544734

Dr. Jurij Jaklič: Upravljanje in uporaba podatkov
Ekonomska fakulteta – skripta; šifra: JJA02AS102
Založila: Ekonomska fakulteta v Ljubljani, Enota za založništvo
Za založnika: Prof.dr. Bogdan Lipičnik
Natisnil: COPIS, Dunajska 158, Ljubljana
Naklada: 400 izvodov, 1. natis
Recenzent: prof.dr. Andrej Kovačič
Oblikovanje: dr. Jurij Jaklič

CIP – Kataložni zapis o publikaciji
Narodna in univerzitetna knjižnica, Ljubljana

004.6(075.8)

JAKLIČ, Jurij - 1967
Upravljanje in uporaba podatkov / Jurij Jaklič. –1. izd. -
Ljubljana : Ekonomska fakulteta, 2002. – (EF. Skripta)

ISBN 961-6430-29-7

119208448

18 -09- 2002 0200213189

Vse pravice pridržane. Noben del tega gradiva se ne sme reproducirati ali kopirati v kakršnikoli obliki: grafično, elektronsko ali mehanično, kar vključuje (ne da bi bilo omejeno na) fotokopiranje, snemanje, skeniranje, tipkanje ali katerekoli druge oblike reproduciranja vsebine brez pisnega dovoljenja avtorja ali druge pravne ali fizične osebe, na katero bi avtor prenesel materialne avtorske pravice.

Kazalo

1. POGLAVJE

PODATKOVNI VIRI V ORGANIZACIJI 1

1.1 Učinkovita uporaba podatkov 2

Problemi upravljanja podatkov 4

Uporabniki 9

1.2 Podatkovna arhitektura 10

Operativni nivo 12

Zgodovinski podatki 13

Referenčni podatki 14

Zunanji podatkovni viri 15

Podatkovni viri za analitično obdelavo 16

Podatkovno skladišče 18

Področno podatkovno skladišče 20

Operativna shramba podatkov 21

Tokovi podatkov 21

Opisovanje podatkov 22

Vprašanja za preverjanje razumevanja 25

Ključni pojmi 26

2. POGLAVJE

UPRAVLJANJE PODATKOV 27

2.1 Življenjski cikel upravljanja podatkov 28

2.2 Aktivnosti upravljanja podatkov 30

Načrtovanje 30

Viri 31

Pridobivanje in vzdrževanje 31

Opredelitev in opis 32

Organizacija in dostopnost 32

Nadzor kakovosti in celovitosti 33

Varovanje in zaščita 34

Obračun uporabe 37

Obnavljanje in nadgradnja 37

Odločanje o obdržanju in odstranitvi 38

Izobraževanje in svetovanje za uporabo 39

2.3 Načrtovanje podatkovnih virov 39

Model entitet – povezav 41

Analiza povezav med entitetnimi tipi 49

Funkcionalna odvisnost 53

Podatkovno modeliranje 56

2.4 Upravljanje znanja 59

Vprašanja za preverjanje razumevanja 67

Ključni pojmi 68

3. POGLAVJE

PODATKOVNE BAZE 69

3.1 Podatkovne baze in njihove značilnosti 70

Značilnosti podatkovne baze 70

Uporaba in druge aktivnosti 71

Sistemi za upravljanje podatkovnih baz 75

Vloge 84

3.2 Načrtovanje podatkovnih baz 85

Podatkovni modeli 86

Hierarhični model 88

Mrežni model 92

Relacijski model podatkov 93

Objektni model 102

Večdimenzionalni model 105

Osnove dobrega načrtovanja podatkovne baze 110

Programska orodja za podporo razvoju baz podatkov 123

3.3 Uporaba podatkovnih baz 125

Kaj je SQL? 125

Zakaj SQL? 126

Jezik SQL 126

Izmenjava podatkov 144

Vprašanja za preverjanje razumevanja 146

Ključni pojmi 146

4. POGLAVJE

INFORMACIJSKA PODPORA MANAGEMENTU 147

Proces odločanja in analiza podatkov 150

4.1 Sistemi za podporo odločanju 159

4.2 Uporaba programskih paketov za delo s preglednicami 164

4.3 Poslovno obveščanje 177

Sprotna analitična obdelava podatkov 178

Rudarjenje v podatkih 184

4.5 Poslovni portali 198

Vprašanja za preverjanje razumevanja 201

Ključni pojmi 202

LITERATURA 203

STVARNO KAZALO 207

Predgovor

Pri nastanku pričujočega gradiva sem imel v mislih predvsem študente pri predmetu Informatika 2 na Ekonomski fakulteti v Ljubljani, v uporabi pa bo tudi pri nekaterih drugih predmetih, ki pokrivajo področje upravljanja podatkov, podatkovnih baz in njihove uporabe. Ker gradivo torej ni v prvi vrsti namenjeno specialistom na tem področju (informatikom), gre predvsem za pregled področja s poudarki na tistih delih, ki bi jih morali razumeti tudi uporabniki za svoje uspešno delo. Pri pisanju sem predvideval, da ima študent, ki bo to gradivo uporabljal, znanja, ki so v učnem načrtu pri predmetu Informatika 1 (ali Poslovna informatika). Prvi inačici gradiva je sledil dodatek, ki je pokrival manjkajoča poglavja iz osnovnega gradiva. Tokratna izdaja obsega celotno gradivo, ki je z nekaterimi vsebinami še dodatno dopolnjeno, nekatere obstoječe vsebine pa so skrajšane ali delno spremenjene glede na dosedanje izkušnje z uporabo.

V informatiki predstavlja zaradi hitrega razvoja področja večen problem slovensko izrazje. Trudil sem se, da bi uporabil čim več uveljavljenih slovenskih prevodov za angleške izraze. Zaradi razvoja so se tudi nekateri prevodi spremenili glede na prejšnjo izdajo. Na primer, podatkovna tržnica je zdaj področno podatkovno skladišče. Posebno težavo je predstavljala uporaba izraza za management, saj se pojavljajo kar trije različni izrazi: management (informacijska podpora managementu), uravnavanje (uravnavanje odnosov s kupci) in upravljanje (upravljanje podatkov, znanja, podatkovnih baz ...). Pri odločitvi je bilo potrebno upoštevati kriterije enovitosti, uveljavljenosti izraza in pomenom. Tokrat je pretehtala uveljavljenost.

Prof. dr. Andreju Kovačiču se zahvaljujem za pregled gradiva in opozorila na nekatere pomanjkljivosti in napake. Seveda je vse, kar je morebiti ostalo napak, izključno moja odgovornost. Zavedam se, da je gradivo še potrebno izboljšav, popravkov in dopolnitev, ki jih načrtujem v prihodnje. Pri tem želim uporabiti predvsem izkušnje z uporabo te izdaje. Zato študente vabim, da mi sporočite svoja opažanja. Za vsako opozorilo ali nasvet za izboljšavo vam bom hvaležen.

Še to: Dele vsebine, ki so namenjene zahtevnejšim študentom, sem posebej označil s prekinjeno črto ob besedilu, na enak način kot ta odstavek. Za uspešno opravljen izpit študij teh delov ni obvezen, vendar je za poglobljeno razumevanje ostalega besedila priporočljiv.

V Ljubljani, julija 2002

1. POGLAVJE

Podatkovni viri v organizaciji

Cilji poglavja

V tem poglavju boste spoznali:

- Da so vlaganja v informacijsko tehnologijo lahko donosna le, če je organizacija informacijsko usmerjena.
- Kateri so poglavitni elementi informacijske usmerjenosti in kako so med seboj povezani?
- Kateri so problemi upravljanja podatkov v organizacijah in kaj so razlogi zanje?
- Kdo so uporabniki podatkov v organizacijah in kakšne so njihove značilnosti?
- Kakšna je sodobno zasnovana podatkovna arhitektura?
- Katere so vrste, značilnosti in pomen različnih vrst podatkov?
- Zakaj so potrebni posebni podatkovni viri za analitične potrebe, kateri so ti viri in kakšne so njihove značilnosti?
- Zakaj je pomembno opisovanje podatkov in kaj obsega?

1.1 Učinkovita uporaba podatkov

Podatki in informacije, ki so bile nekoč redka dobrina, postajajo z razvojem informacijske in komunikacijske tehnologije (na primer Interneta) vsepovsod prisotne in dostopne. Kupci, zaposleni, dobavitelji in tekmeci imajo možnost dostopa po mnogih komunikacijskih poteh do istih poslovnih in gospodarskih podatkov (Marchand, Kettinger, Rollins, 2001). To zahteva od organizacij še večji trud, da bi dosegle in povečale **informacijsko asimetrijo** oziroma uporabile podatke in informacije, ki jih zbirajo in so dostopne, za doseganje konkurenčne prednosti.

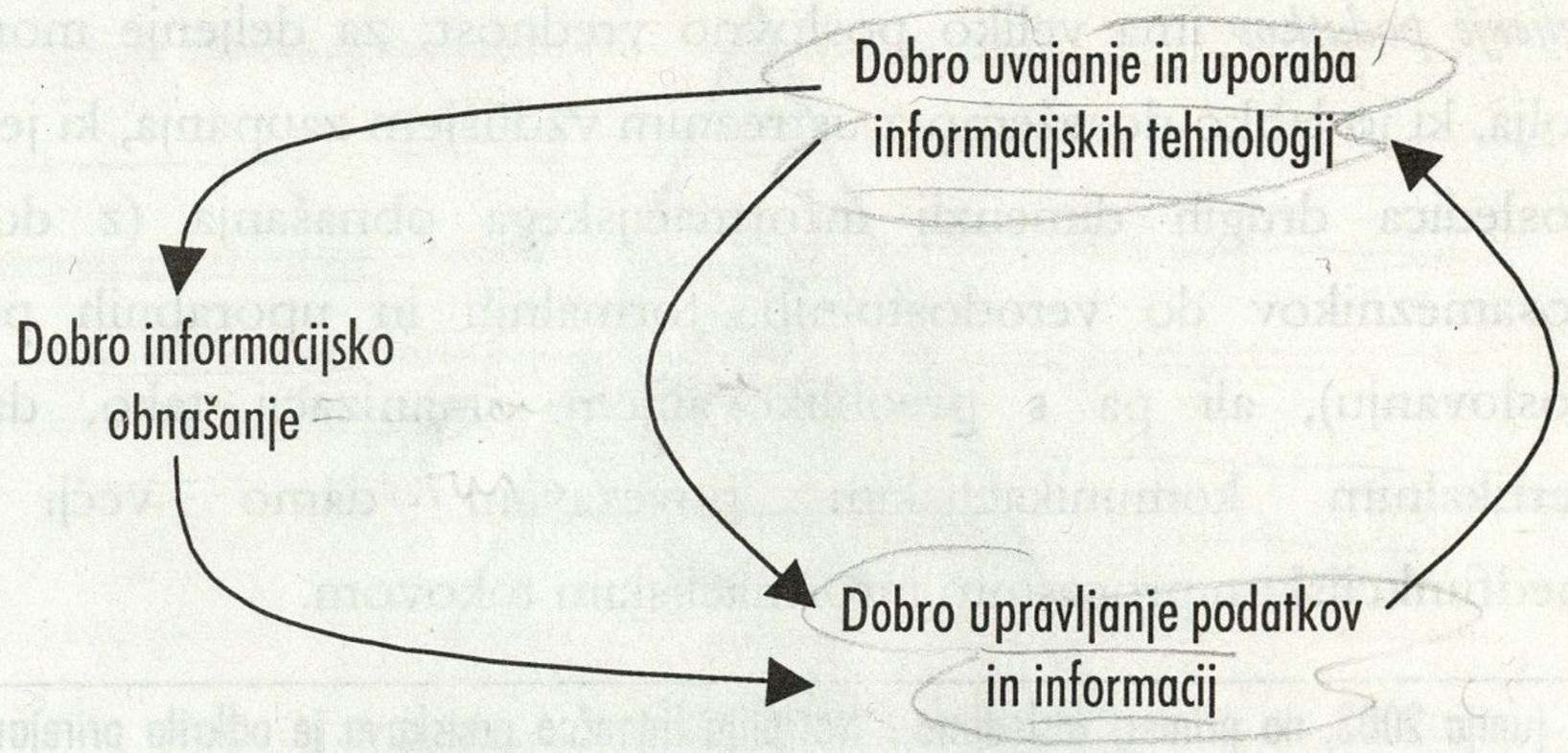
Razumevanje, da je osnova učinkovite uporabe podatkov zgolj uporaba sodobnih informacijskih tehnologij za pretvarjanje le-teh v informacije, se je v praksi izkazalo za napačno. Večina raziskav, v katerih so iskali povezave med vlaganji v informatiko (ali informacijsko tehnologijo) ter uspešnostjo podjetij, tovrstnih povezav ni odkrila ali pa je bila podpora povezavam zelo šibka[1,2]. To kaže na dejstvo, da zgolj uvajanje novih tehnologij ni dovolj, pač pa mora postati organizacija **informacijsko usmerjena** (Marchand, Kettinger, Rollins, 2001), kar vključuje poleg uporabe sodobnih *informacijskih tehnologij za podporo odločanju* in *reševanju problemov* tudi **upravljanje podatkov** in **informacijsko obnašanje** (informacijsko kulturo).

Slika 1 prikazuje medsebojno povezanost vseh elementov: dobro razvita informacijska kultura in dobro upravljanje podatkov povečujejo možnost učinkovite uporabe informacijske tehnologije za podporo odločanju, to pa spet povečuje informacijsko obnašanje in vzpodbuja izboljšanje upravljanja podatkov.

---

1 Lester (1998 v Marchand, Kettinger, Rollins, 2001) pravi: »Računalnike lahko najdete povsod, le v statistikah o produktivnosti ne.«

2 Velika vlaganja v informacijsko tehnologijo v ameriških podjetjih v 1980. letih, se niso odražala v povečani produktivnosti gospodarstva (Marchand, Kettinger, Rollins, 2001).



Slika 1: Spirala učinkovite uporabe podatkov (Marchand, Kettinger, Rollins, 2001)

Medtem ko sta področji upravljanja podatkov ter upravljanja in uporabe informacijske tehnologije relativno dobro razviti, pa področje informacijskega obnašanja šele v zadnjem času vzbuja več zanimanja in raziskovanja. Informacijsko obnašanje se je namreč v preteklosti obravnavalo kot del splošnega upravljanja (managementa).

Značilnost dobrega informacijskega obnašanja je **proaktivna uporaba podatkov** in informacij, ki pa je odvisno od drugih dimenzij informacijskega obnašanja:

- *podatki* morajo biti *celoviti*, to je *verodostojni*, *točni* in *nepristrani*, zaposleni morajo verjeti vanje,
- *uporaba formalnih virov podatkov* mora imeti prednost pred uporabo podatkov iz neformalnih virov, kar povečuje zanesljivost in kakovost uporabljenih podatkov,
- podatki (npr. računovodski) omogočajo *nadzor* nad delovanjem posameznika in organizacije kot celote in imajo s tem motivacijsko funkcijo,
- *transparentnost delovanja* zahteva pošteno obravnavanje odločitev, napak, zmot, problemov; iz podatkov so razvidni tudi neuspehi in problemi, ne le pozitivne plati poslovanja,

- *deljenje podatkov* ima veliko poslovno vrednost; za deljenje mora obstajati volja, ki jo lahko dosežemo z ustreznim vzdušjem zaupanja, ki je spet lahko posledica drugih dimenzij informacijskega obnašanja (z dostopnostjo posameznikov do verodostojnih, formalnih in uporabnih podatkov o poslovanju), ali pa s preoblikovanjem organizacij tako, da namesto vertikalnim komunikacijskim povezavam damo večji poudarek medfunkcijskim procesnim informacijskim tokovom.

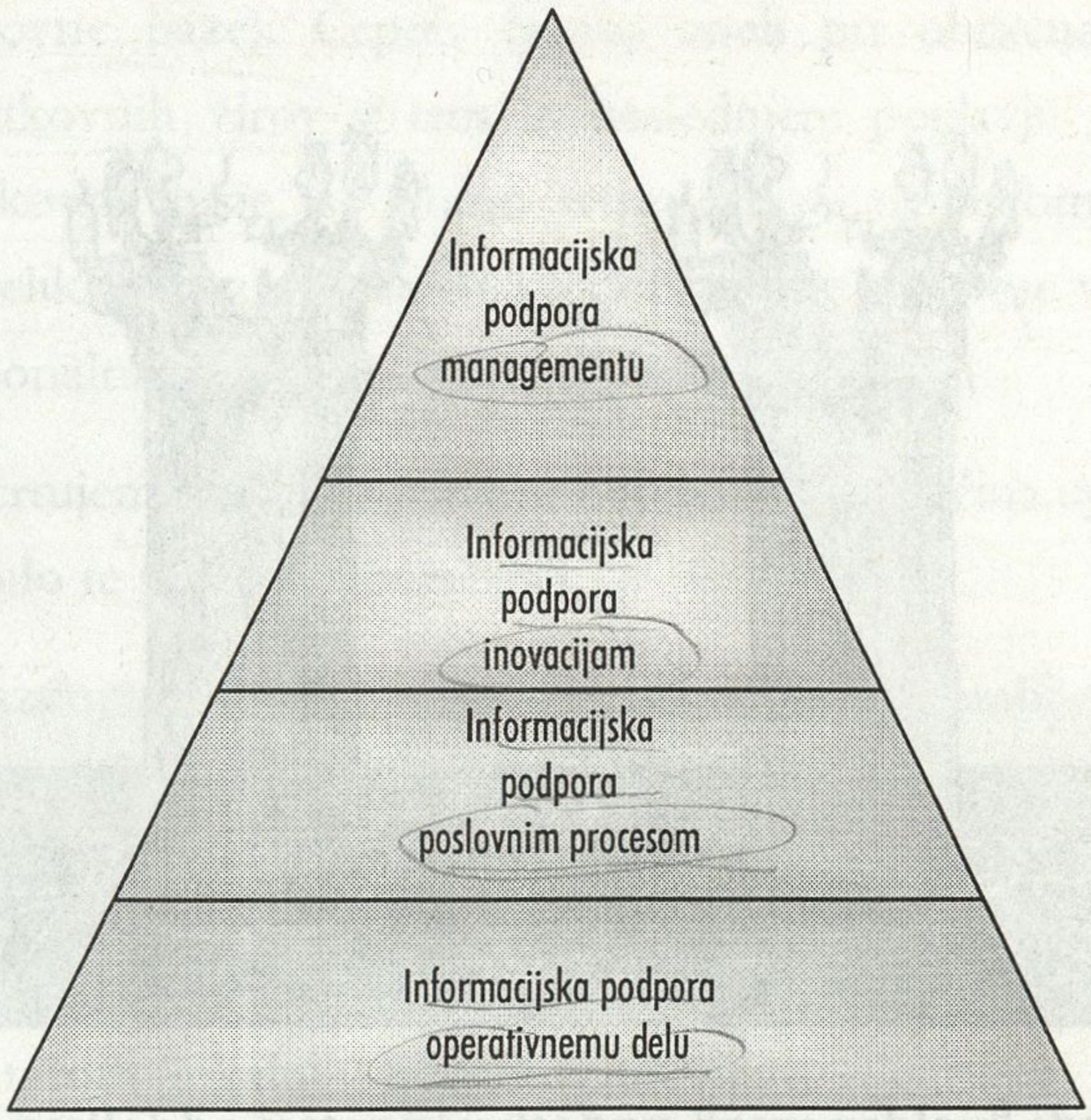
V Delu 28. junija 2002, na primer, zasledimo: "Notranja finančna preiskava je odkrila prirejanje podatkov o denarnih tokovih ... S pravilno vodenimi računovodskimi izkazi bi imelo podjetje izgubo ... Worldcom pa je imel s prirejeno bilanco lani za 35,2 milijarde dolarjev prodajnih prihodkov, prijavil je 1,4 milijarde dolarjev dobička, ..., saj je zdaj že nekdanji finančni direktor knjižil operacijske stroške, denimo vzdrževanja opreme, kot kapitalska vlaganja ..."

Praktične izkušnje kažejo, da je prav transparentnost tisti element, ki pogosto povzroča zastoje pri projektih informatizacije poslovanja. Dogaja se, da si vplivni zaposleni (kljub drugačnim besedam in deklarativni podpori) možnosti, da bi imeli ostali vpogled v dejansko stanje, ne želijo.

Problemi upravljanja podatkov

Podatki predstavljajo enega *najpomembnejših virov* v vsaki organizaciji, saj večina opravil na vseh nivojih organizacije (glej sliko 2) danes temelji na informacijah. To pomeni, da moramo podatke obravnavati in upravljati na enak način kot vse druge vire: stroje, človeške vire, kapital … Uspešno upravljanje podatkov pomeni, da morajo biti podatki dostopni takrat, ko jih potrebujemo, in v obliki, v kakršni jih potrebujemo. Poglavitni podatkovni problemi, ki jih opažamo v praksi so (Turban, Aronson, 2001): podatki niso pravilni, podatki niso pravočasno na voljo ali pa potrebni podatki sploh ne obstajajo.

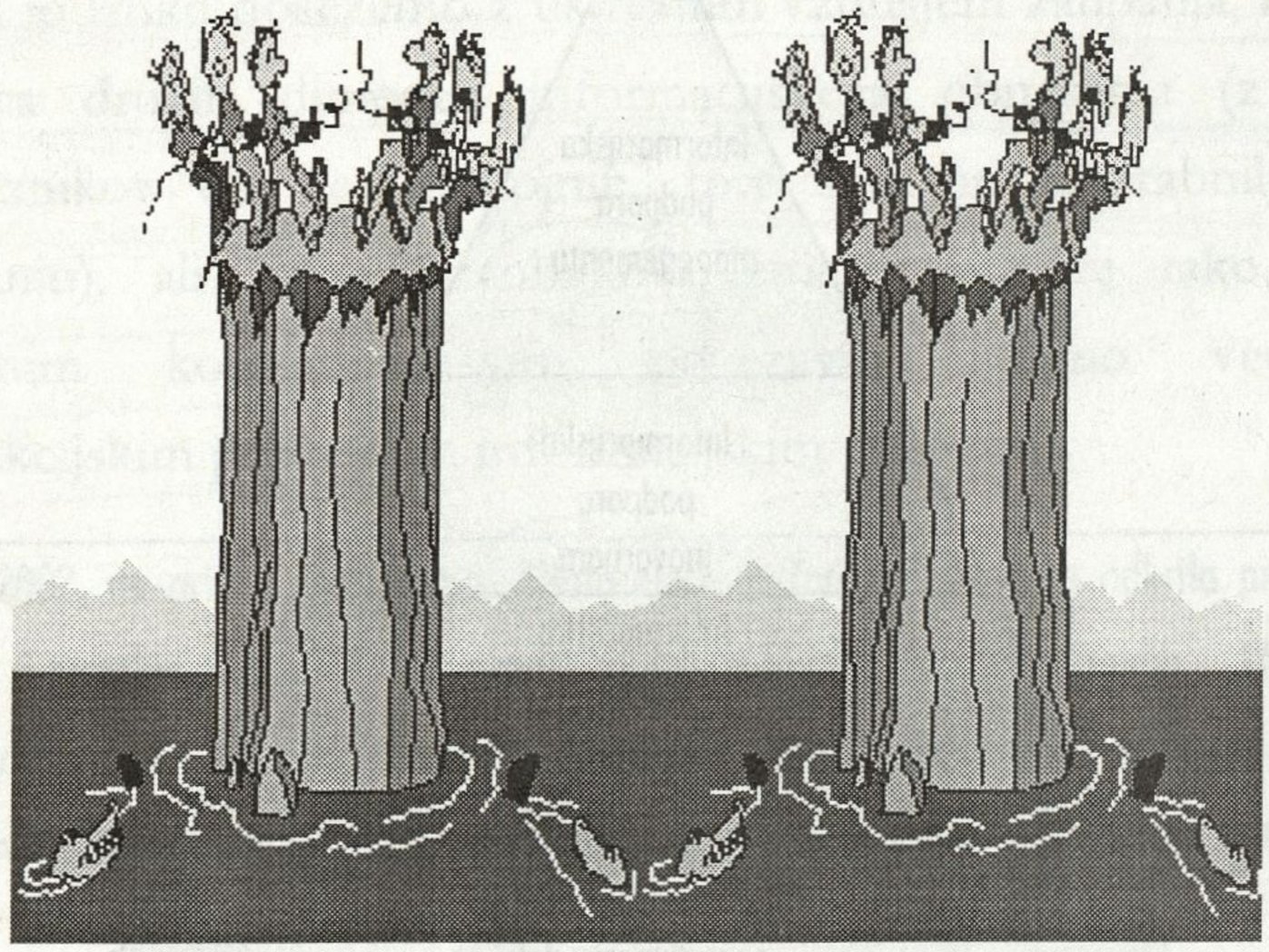
Upravljanje podatkov je torej odgovornost vsakega managerja, tako kot je to upravljanje financ, zaposlenih, opreme … Aktivnosti upravljanja podatkov, kot so načrtovanje, nadzor kakovosti, vodenje evidence, identifikacija kakovostnih virov, načrtovanje zaščite in podobno so tipično najprej poslovna vprašanja in šele nato tehnična vprašanja. Šele ko se odločimo, kdo ima pravico dostopa do katerega podatka, lahko skrbnik podatkovne baze (glej poglavje Podatkovne baze) poskrbi za ustrezne tehnološke rešitve, ki omogočajo dostop samo pooblaščenim.



Slika 2: Piramida informacijske podpore

Pri upravljanju podatkov pa v organizacijah tradicionalno opažamo probleme, ki izhajajo iz zastarele miselnosti in načina obdelave podatkov in so znak slabo razvitega informacijskega obnašanja. Posamezni oddelki ali managerji si namreč lastijo podatke, ki jih uporabljajo. Posledično si podatkov ne delijo, izmenjava podatkov je otežkočena ali nemogoča, za odločanje ni na voljo informacij, ki jih potrebujemo, podatki pa se v veliki meri podvajajo. Vzroki za tako "zaklepanje" podatkov izhaja iz dejstva, da podatki pomenijo moč, ki je pogosto povezana z netransparentnostjo, neverodostojnim in pristranim prikazom podatkov navzven in podobno. To pa je s stališča organizacije kot celote nesprejemljivo.

Tak način upravljanja podatkov je bil še podkrepljen z informacijsko tehnologijo preteklosti, ko je vsak program (npr. računovodski, kadrovska evidenca, obračun plač) uporabljal svoje podatke. Ker programi niso uporabljali skupnih podatkov, je bilo nujno, da so bili isti podatki shranjeni večkrat v različnih oblikah in pod različnimi imeni.



Slika 3: **Zaradi "zaklepanja" podatkov znotraj oddelkov prihaja v organizaciji do otokov informacij**

Za skrbnike podatkovnih virov je to predstavljalo "nočne more" pri vzdrževanju, po drugi strani pa je prihajalo do nekonsistentnih podatkov, saj isti podatki niso bili spremenjeni hkrati na vseh mestih. Na primer, naslov zaposlenega, ki se je pred kratkim preselil, je bil v enem programu[3] že spremenjen, v drugem pa ne.

Podatki morajo biti torej zbrani v skupnem viru podatkov, brez njihovega nepotrebnega podvajanja. Hkrati to pomeni, da mora biti načrtovanje in upravljanje podatkov koordinirano.

Idealna rešitev za tak skupni vir podatkov je **celovita** (integrirana) računalniško podprta **podatkovna baza**[4], za katero je značilno prav to, da ne vsebuje nepotrebnega podvajanja podatkov, dostop do podatkov pa imajo vsi uporabniki, ki te podatke potrebujejo. Ker je danes podatkovna baza pomemben del informacijskega sistema večine organizacij, ji namenjamo posebno poglavje (glej

---

3 Tako izražanje ni povsem natančno, saj so bili podatki shranjeni na datotekah in ne neposredno kodirani v programih. Ker pa so bili podatki tesno povezani s programi, moremo in želimo s takim "pogovornim" izražanjem to povezavo še posebej poudariti.

4 Podatkovna baza je, kot bomo podrobneje videli v nadaljevanju, neka urejena, računalniško podprta zbirka podatkov.

poglavje Podatkovne baze). Čeprav bomo imeli pri obravnavi opisovanja in upravljanja podatkovnih virov v tem in naslednjem poglavju pogosto v mislih predvsem podatkovne baze, se bomo temu izrazu namenoma, kjer se le da, izogibali, saj se velik del razmisleka enako dobro nanaša na vse vrste podatkovnih virov, tudi tradicionalnih kot so, na primer, papirni arhivi.

Podatke torej načrtujemo in organiziramo neodvisno od načina uporabe, na primer programov, ki bodo te podatke uporabljali.

Žal pa je bil razvoj v mnogih organizacijah kljub uporabi podatkovnih baz drugačen:

- Zaradi potreb po hitrem razvoju uporabniških rešitev se je marsikje zanemarjalo celovito načrtovanje informatike, kar je privedlo ponovno do uporabe internih (lokalnih) podatkovnih virov.
- Marsikje obstoječi sistemi delujejo dobro in bi jih bilo nesmotrno zavreči, zato so podatkovni viri integrirani le deloma v novejših delih informacijskega sistema.
- Celovite rešitve so lahko izjemno drage, česar si nekatere organizacije ne morejo privoščiti.
- Zaradi razvoja osebnih računalnikov ter uporabniško prijazne informacijske tehnologije in s tem "demokratizacije" informatike ter neučinkovitih služb za informatiko se je mnogo uporabnikov lotilo izdelave lastnih podatkovnih virov (največkrat v obliki preglednic, tabel v urejevalnikih besedil, pa tudi preprostih podatkovnih baz). Uporabniki so s tem tudi prevzeli nadzor nad lastnimi podatkovnimi viri, kar je v nasprotju z željo po integraciji podatkov.
- V nekaterih organizacijah se še vedno lotevajo informatizacije tako, da informatizirajo oddelke in ne procese, kar vodi v nepovezanost delov informacijskega sistema.

Zaradi teh in drugih razlogov se v večini organizacij danes srečujejo s stanjem, ko imajo množico podatkovnih virov, v zelo različnih oblikah, šibko povezanih, z močno prisotnim podvajanjem podatkov, z nekonsistentnimi podatki. Slabosti takih sistemov se pokažejo predvsem na višjih nivojih odločanja, ko pri analizah

potrebujemo množice raznovrstnih podatkov, ki se nahajajo v različnih podatkovnih virih. Ko nam le uspe identificirati podatke, jih zbrati, prečistiti ... pa ugotovimo, da podatkovni viri niso konsistentni[5].

Takšnemu okolju množice nekoordinirano nastalih podatkovnih virov pravimo **okolje pajkove mreže** (angl. spider web environment). Čeprav si takega stanja ne želimo, pa ga moramo pri sodobnem načrtovanju sistema podatkovnih virov v organizaciji upoštevati (glej podpoglavje Podatkovna arhitektura).

Poudarimo pa, da lahko razpršeni sistemi, to so sistemi, v katerih imamo več podatkovnih virov, prav tako dobro služijo svojemu namenu, če le upoštevamo načela, ki veljajo za izgradnjo integrirane podatkovne baze, in načela dobrega upravljanja s podatki.

Kadar imamo v organizaciji razpršene podatkovne vire, moramo skrbeti za dovolj pogost[6] (in po možnosti avtomatiziran) prenos podatkov med podatkovnimi viri, ki vsebujejo iste podatke. V idealnem primeru vsak podatek zajamemo samo enkrat, nato pa ga posredujemo v vse podatkovne vire, ki ta podatek ali od njega odvisen podatek vsebujejo. Na primer, spremembo naslova kupca, ki jo zajamemo v računovodstvu, shranimo v računovodsko podatkovno bazo, nato pa posredujemo v trženjsko podatkovno bazo. Iz podatkov o prodaji moramo vsaj enkrat mesečno izračunati skupno prodajo in ta sumarni podatek posredovati v bazo, ki se uporablja za napoved prodaje.

---

5 To pomeni, da npr. če pri analizi uporabimo istovrstne podatke, ki se nahajajo v dveh podatkovnih virih, dobimo različne rezultate. Neredko se zgodi, da bosta dala dva oddelka popolnoma različni poročili o isti dejavnosti. Za širšo obravnavo razlogov za takšne dogodke glej (Inmon, 1996).

6 Natančneje pogostosti ne moremo opredeliti, saj je le-ta odvisna od potreb organizacije.

Hkrati z razvojem pajkovih mrež se je povečevala pri uporabnikih na vseh nivojih odločanja tudi "lakota" po informacijah. Uporaba kakovostnih, to je *pravočasnih*, *relevantnih*, *konsistentnih* ter *popolnih* podatkov je za poslovno odločanje postala nepogrešljiva. Podatki, ki jih dobimo iz operativnih podatkovnih virov, ne zadoščajo za učinkovito podporo odločanju. Zato je razumljivo prišlo do razvoja novih vrst podatkovnih virov, to je podatkovnih baz, ki so namenjene prav podpori odločanju: operativnih shramb podatkov, podatkovnih skladišč in področnih podatkovnih skladišč. Ena od njihovih najpomembnejših značilnost je, da vsebujejo integrirane podatke iz okolja pajkove mreže, to je iz operativnih podatkovnih virov.

Uporabniki

Za učinkovito informacijsko podporo je pomembno tudi razumevanje potreb uporabnikov. Čeprav so te potrebe zelo raznolike, lahko uporabnike podatkov[7] v grobem razdelimo v dve veliki skupini:

- operativni uporabniki ter
- uporabniki podatkov za podporo odločanju (analitični nivo).

Operativni uporabniki se ukvarjajo s trenutnimi in zelo neposrednimi odločitvami, na primer:

- Koliko pošiljk moramo pripraviti danes?
- Kje je ta trenutek pošiljka?
- Kakšen je skupni znesek računa?
- Koliko izdelkov imamo na zalogi?

7 Tu niso mišljeni le tisti uporabniki, ki uporabljajo podatke (do njih dostopajo), pač pa širše vsi, ki imajo opravka s podatki (vnos, ažuriranje in podobno).

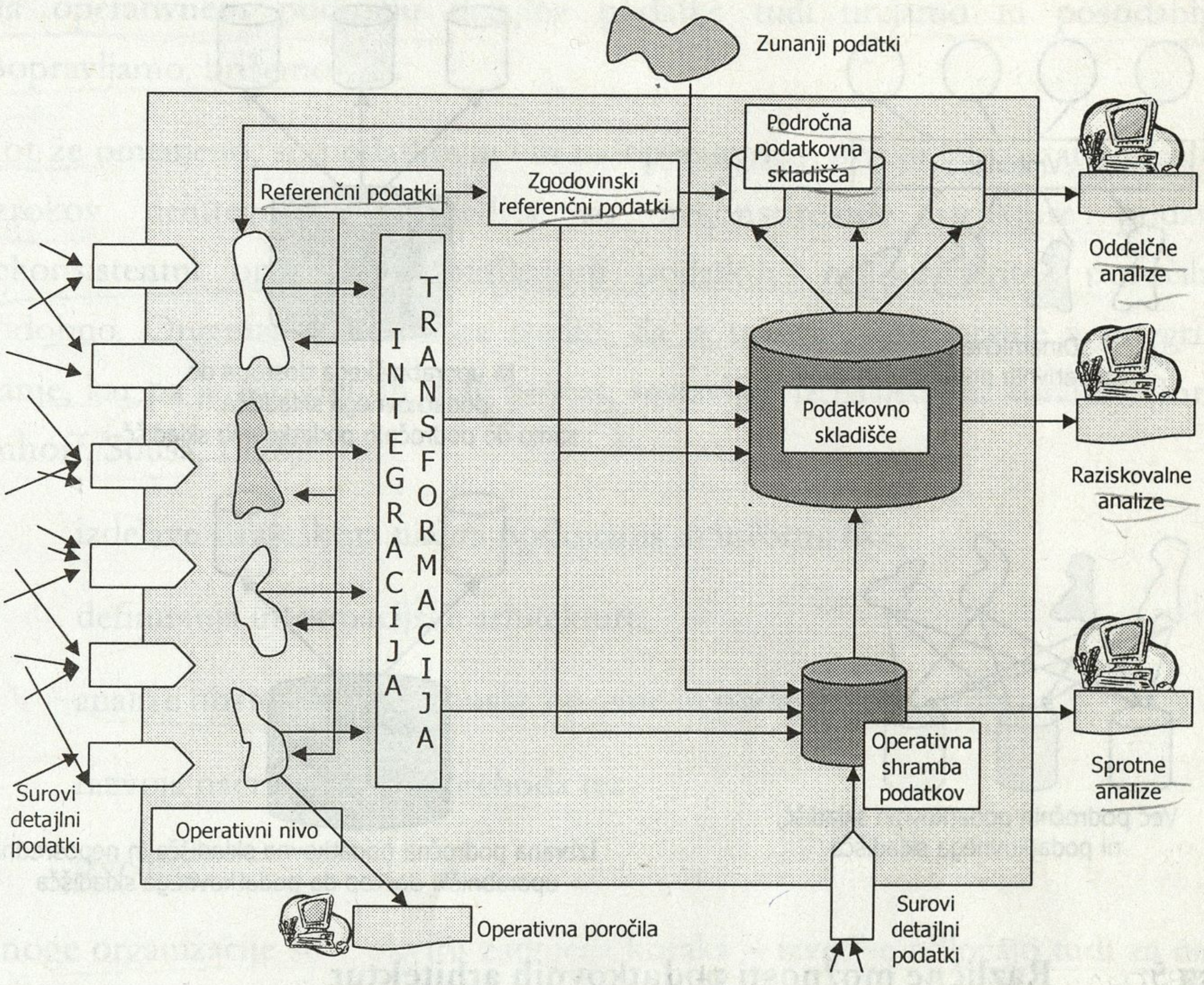
Uporabnik podatkov za podporo odločanju (managerski in strateški nivo) pa se ukvarja z mnogo širšimi in bolj dolgoročnimi vprašanji. Analitične uporabnike zanima, na primer:

- Primerjava prodaje določenega izdelka v prvem trimesečju lani in letos?
- Primerjava prodaje po regijah; za vse izdelke in za vsak izdelek posebej?
- Vpliv postavitve izdelka v trgovini na njegovo prodajo?
- Kakšne bi bile posledice 5% povečanja cen na prodajo (elastičnost povpraševanja)?

Analitični uporabnik ima torej povsem drugačen pogled na podatke; širši, integrativen, zanimajo ga podatki daljšega časovnega obdobja in ne le trenutne vrednosti.

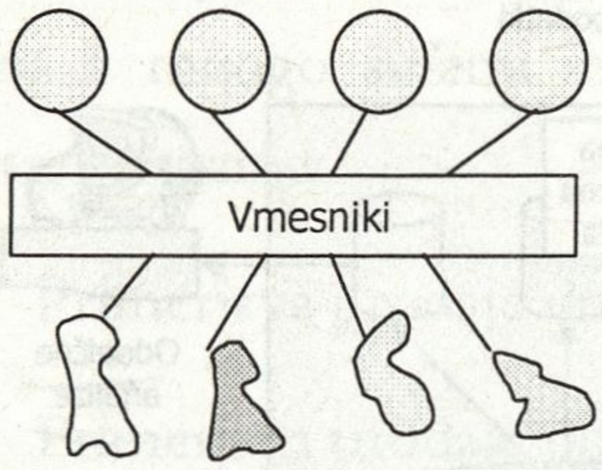
1.2 Podatkovna arhitektura

Sodobno zasnovana **podatkovna arhitektura** mora izhajati iz prej opisane problematike in zagotavljati povezano ter uravnoteženo informacijsko okolje. Sistem podatkovnih virov in drugih komponent (programska oprema, komunikacijska tehnologija ...) mora biti *prilagodljiv* in se spreminjati v skladu s spremembami v okolju.

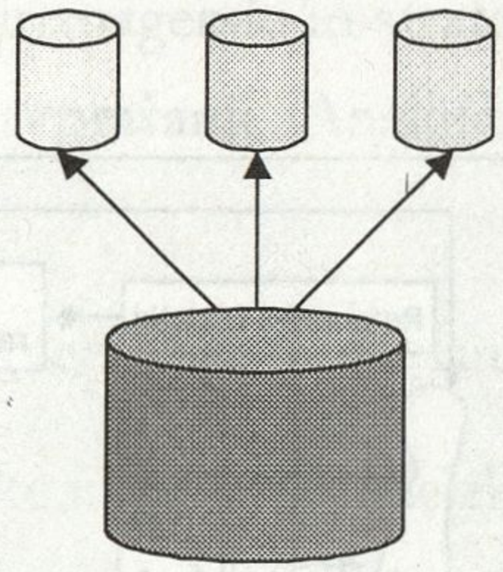


Slika 4: Primer podatkovne arhitekture (Inmon, Imhoff, Sousa, 1998)

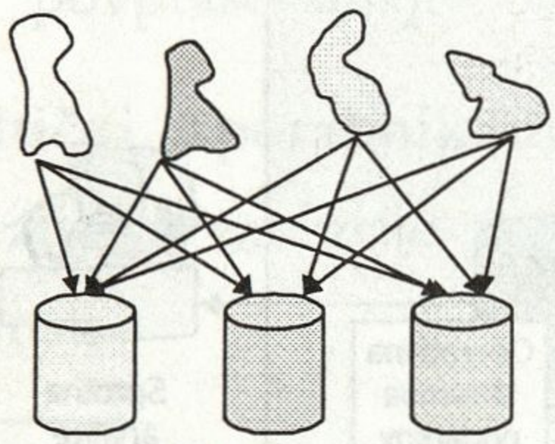
Podatkovna arhitektura, prikazana na sliki 4, vsebuje vse bistvene komponente takega sistema podatkovnih virov, vsaka organizacija pa bo glede na lastne informacijske potrebe izdelala ustrezno podatkovno arhitekturo, ki bo vsebovala vse ali samo nekatere od teh komponent. Na sliki 5 so prikazane nekateri primeri možnih podatkovnih arhitektur.



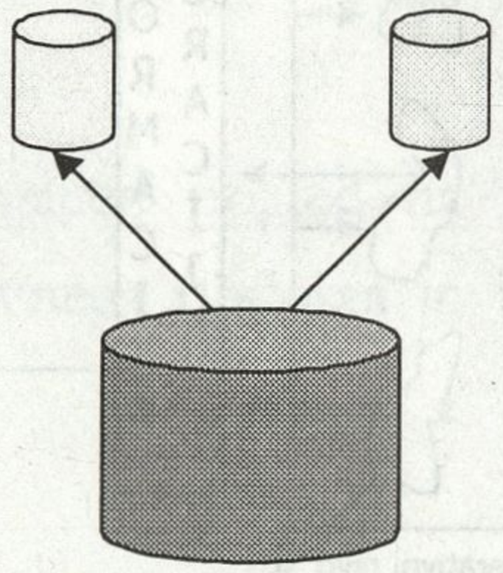
Dinamične poizvedbe
operativnih podatkovnih virov



Ni uporabniškega dostopa do
podatkovnega skladišča,
samo do področnih podatkovnih skladišč



Več področnih podatkovnih skladišč,
ni podatkovnega skladišča



Izbrana področna podatkovna skladišča in neposredni
uporabniški dostop do podatkovnega skladišča

Slika 5: Različne možnosti podatkovnih arhitektur (Gartner Gorup, 1999)

Operativni nivo

Surovi detajlni podatki so podatki o vseh objektih in dogodkih, ki so povezani z vsakodnevnim poslovanjem organizacije, na primer, podatki o kupcih, dobaviteljih, naročilih, dobavah, prodajah, plačilih računov, notranjih delovnih naročilih, projektih, zaposlenih …

Surove detajlne podatke v splošnem zajamemo in uporabljamo na **operativnem nivoju**. Z ustrezno programsko opremo jih shranimo v operativne podatkovne baze posameznega področja (lokalne podatkovne vire), od koder jih preko transformacijskega in integracijskega sloja prenesemo v integrirane podatkovne vire, namenjene odločanju (operativno shrambo podatkov, podatkovno skladišče in področna podatkovna skladišča). Le redko podatke zajemamo tudi neposredno v operativno shrambo podatkov, kadar potrebnih podatkov, ne zajemamo z nobeno programsko rešitvijo na operativnem nivoju.

Na operativnem področju detajlne podatke tudi urejamo in posodabljamo (popravljamo, brišemo ...).

Kot že omenjeno, so podatkovni viri na operativnem nivoju lahko zaradi različnih vzrokov neintegrirani. Posledica so nekonsistentne strukture podatkov, nekonsistentni opisi in identifikatorji podatkov, nekonsistentna poročila in podobno. Organizacija se mora truditi, da iz takega stanja preide v integrirano stanje, kar pa je zapleten in drag proces, sestavljen iz naslednjih korakov (Inmon, Imhoff, Sousa, 1998):

- izdelave strateškega načrta poslovanja in informatike,
- definiranja informacijske arhitekture,
- analize obstoječe programske opreme in podatkovnih virov,
- razvoja načrta projekta prehoda ter
- izvedbe.

Mnoge organizacije se v okviru zadnjega koraka – izvedbe odločajo tudi za nakup že izdelanih celovitih programskih rešitev, kot so SAP, Baan, Oracle ... Več o informatizaciji lahko bralec najde v (Kovačič, 1998).

Zgodovinski podatki

Nekateri podatkovni viri (predvsem operativni) so namenjeni zlasti hranjenju trenutnih podatkov, oziroma podatkov iz bližnje preteklosti. Podatkovni viri, ki so usmerjeni v uporabo pri podpori odločanju, pa morajo vsebovati tudi t.i. **zgodovinske podatke**, to je podatke iz daljšega časovnega obdobja.

Kdaj postanejo podatki zgodovinski? Pravzaprav lahko rečemo, da so vsi podatki, takoj ko vstopijo v sistem, zgodovinski (razen napovedi). Torej vprašanje ni, ali je podatek zgodovinski, pač pa, koliko je zgodovinski.

Pri zgodovinskih podatkih se nam zastavlja vprašanje, kako dolgo naj jih hranimo in v kakšni obliki (detajlni ali sumarni)? Na odločitev vplivajo elementi:

- *Količina podatkov.* S časom se nabere ogromna količina podatkov, ki jih več ne obvladujemo.
- *Poslovna uporabnost.* Za podatke iz bližnje preteklosti je večja verjetnost, da bodo relevantni za poslovne odločitve.
- *Stopnja agregacije.* Podatki iz bližnje preteklosti bodo verjetneje uporabljeni na detajlnem nivoju, starejši podatki pa verjetneje na sumarnem nivoju.

Več o vprašanju odločanja o obdržanju ali odstranitvi podatkov najde bralec še v poglavju Upravljanje podatkov.

Referenčni podatki



Referenčni podatki so operativni podatki, ki povezujejo raznolike uporabnike in omogočajo standardizacijo procesov med različnimi oddelki. To so torej podatki, ki so skupni za več uporabniških področij (programskih rešitev) in so torej last organizacije kot celote in ne posameznih skupin uporabnikov (oddelkov in podobno). Tipični primeri referenčnih podatkov so podatki o izdelkih, podatki o kupcih, podatki o poštnih številkah.

Referenčni podatki zagotavljajo določeno mero integracije operativnih podatkovnih virov. Na primer, z referenčnimi podatki o izdelkih bosta dva oddelka enako razvrstila izdelke v kategorije pri pripravi poročil o prodaji izdelkov.

Praviloma se spreminjajo redko in so stabilni, zato se z njimi (strukturiranjem in urejanjem) ne ukvarjamo dosti. Seveda pa je zaradi pomembnosti referenčnih podatkov za podjetje nujna stalna skrb za njihovo ažurnost. Referenčni podatki namreč odražajo trenutno stanje – so ažurni v trenutku uporabe. Pri prehodu v druge podatkovne vire, kjer je pomembno tudi hranjenje starih vrednosti, npr. opis izdelka pred spremembo, pa se spremenijo v **zgodovinske referenčne podatke**.

Pogosto so viri referenčnih podatkov zunanji, na primer podatki o poštnih številkah ali devizni tečaji.

Zunanji podatkovni viri

Zunanji podatki niso generirani znotraj organizacije, jih tam ne zajemamo na detajlnem nivoju. Predstavljajo dogodke in objekte izven organizacije, ki so za organizacijo zanimivi. Uporabljamo jih lahko povsod: na operativnem nivoju, lahko jih vključimo v podatkovno skladišče, področna podatkovna skladišča ali v operativno shrambo podatkov.

Ker jih zajemamo iz zunanjih podatkovnih virov, nimamo vpliva na njihovo obliko: lahko so kakršnegakoli tipa in velikosti, strukturirani ali nestrukturirani, detajlni ali sumarni, zato jih na vhodu običajno obdelamo tako, da ustrezajo našim potrebam.

Želeno je, da tudi zunanje podatke zajemamo avtomatizirano. Internet je v zadnjem času eden pomembnejših medijev za dostop do zunanjih podatkovnih virov. Omogoča dostop do bolj ali manj urejenih podatkovnih baz (npr. statističnih podatkov) ali kar neposredno do podatkov na spletnih straneh[8]. Dostopamo lahko do podatkov o strankah, kupcih, konkurentih … Čeprav je preko Interneta dostopnih ogromno podatkov, pa je uporabo bolj kakovostnih podatkovnih virov ponavadi potrebno plačati.

Podatki lahko prihajajo v podatkovne vire organizacije tudi skozi sistem računalniške izmenjave podatkov – RIP (angl. Electronic Data Interchange – EDI), kjer gre za povezavo informacijskih sistemov dveh ali več organizacij. V zadnjem času se za povezovanje čedalje bolj pogosto uporabljajo **ekstraneti**, kjer gre za skupne informacijske sisteme več organizacij, ki temeljijo na tehnologijah Interneta, predvsem spletnih tehnologijah in standardu za opis podatkov XML (glej podpoglavje Opisovanje podatkov v nadaljevanju). Tako lahko, na primer, dobavitelj preko (ustrezno zaščitenih) spletnih strani svojega kupca pregleda stanje zalog in ugotovi, katere izdelke mora dobaviti. Na ta način podjetja ne le povezujejo informacijske sisteme, pač pa integrirajo tudi poslovne procese.

---

8 Sodobna orodja omogočajo kar poizvedovanje po podatkih na spletnih straneh. Na primer, eno od najpogosteje uporabljenih analitičnih orodij MS Excel ima že vgrajeno funkcijo Spletna poizvedba (angl. Web Query), ki omogoča poizvedovanje po tabelah na spletnih straneh na podoben način kot po ostalih strukturiranih podatkovnih virih.

Procese, ki potekajo med podjetji, obravnavajo kot celoto, jih skušajo optimirati kot celoto in ne le znotraj posameznega podjetja.

Podatkovni viri za analitično obdelavo

Kot smo že ugotovili, večino podatkov zajamemo v operativne podatkovne baze. Torej bi lahko uporabniki analitična orodja uporabljali kar neposredno nad podatki v operativnih podatkovnih bazah. Zato se na tem mestu, preden se lotimo obravnave analitičnih podatkovnih virov (podatkovna skladišča in področna podatkovna skladišča), vprašajmo, zakaj so pravzaprav potrebni posebni podatkovni viri, namenjeni analitičnim obdelavam, to je podpori odločanju.

Enega izmed razlogov lahko razberemo iz že povedanega. Podatki v operativnih podatkovnih virih so neintegrirani, zato se prekrivajo, opisi niso enaki, identifikatorji[9] se razlikujejo in podobno. Zato je potrebno za analize, za katere je običajno potrebnih veliko podatkov iz različnih operativnih podatkovnih virov, podatke predhodno integrirati in prečistiti (odpraviti podvajanja, poenotiti obliko in podobno).

Potrebe po podatkih na operativnem nivoju in na analitičnem nivoju se povsem razlikujejo. Tako je za uporabo podatkov na operativnem nivoju značilno:

- da se podatki ne bi smeli podvajati, s čemer se izognemo težavam pri njihovem posodabljanju,
- uporabnike zanimajo predvsem trenutne vrednosti podatkov,
- dostop do podatkov je pogost (časovno enakomerna obremenjenost podatkovnih virov), količina podatkov, s katerimi delamo pa ponavadi majhna (podatki o enem nakupu, enem kupcu in le nekaj izdelkih),

9 Identifikator (tudi oznaka ali ključ) je podatek, ki nam omogoča razlikovanje istovrstnih objektov, o katerih hranimo podatke. Na primer, zaposlene lahko identificiramo z EMŠO, šifro zaposlenega in podobno. Več o tem v podpoglavju Načrtovanje podatkovnih virov.

- način uporabe podatkov se le redko spreminja (ponavljajoč vzorec uporabe),
- zanesljivost mora biti visoka, čas dostopa pa majhen.

Po drugi strani pa za uporabo na analitičnem nivoju velja:

- da podatkov večinoma ne posodabljamo, zato podvajanje ne predstavlja problema,
- uporabnike zanimajo tudi zgodovinski podatki (npr. gibanje prodaje v preteklih letih),
- dostop do podatkov je neenakomeren (morda nekaj analiz dnevno), količina podatkov, s katerimi delamo, pa ponavadi velika, čeprav analitik dobi kot rezultat le nekaj sumarnih vrednosti,
- način uporabe podatkov zelo raznolik,
- zanesljivost ni tako pomembna kot na operativnem nivoju, za zahtevnejše analize pa smo pripravljeni počakati nekaj sekund ali minut, morda celo ur.

Ker je za operativni nivo pomembno, da se podatki ne podvajajo, je struktura podatkov v operativnih bazah zapletena. Uporabniki te zapletene strukture večinoma ne opazijo, saj zaradi ponavljajočih se vzorcev uporabe do podatkov dostopajo skozi vnaprej pripravljene programe, ki strukturo podatkov skrijejo. Način uporabe analitikov pa je zelo raznolik; vseh analiz, ki jih potrebujejo, ni mogoče vnaprej predvideti. Zato tudi ni mogoče vnaprej pripraviti vseh mogočih poročil oziroma pregledov podatkov. **Sprotna analitična obdelava** (angl. On-Line Analytical Processing – OLAP) omogoča uporabniku izdelavo poljubnih analiz nad podatki. Zato potrebujemo:

- orodja, ki tako analizo omogočajo in so dovolj preprosta za uporabo, ter
- *preprosto strukturirane podatke*, saj mora analitik, ki z orodji obdeluje podatke, razumeti njihovo strukturo.

Zato so analitični podatkovni viri ponavadi strukturirani preprosto (večdimenzionalni model, o katerem pišemo v poglavju Podatkovne baze) tudi na škodo podvajanja podatkov.

Če bi izvajali analitično obdelavo kar nad operativnimi podatkovnimi viri, bi to pomenilo tudi njihovo dodatno obremenitev. Te obremenitve so, kot vidimo iz zgornje primerjave potreb po podatkih, neenakomerne, vendar velike, saj analitik ponavadi hkrati obdeluje veliko množico podatkov. Tipičen primer poizvedbe na analitičnem nivoju je primerjava prodaje nekega izdelka v preteklem in tem četrtletju. Čeprav bo rezultat predstavljen le z dvema številoma, je potrebno pregledati vse prodaje v zadnjega pol leta, kar pa je običajno ogromno podatkov. Izvajanje take poizvedbe bi lahko povsem obremenilo operativni podatkovni vir. Nekoliko nerodno bi bilo stranki v trgovini reči: "Oprostite. Ne morem vam računati blaga. Počakati bo potrebno kakih petnajst minut, saj naš direktor ravno analizira prodajo izdelkov, da bo ugotovil, kako bi privabili več kupcev."

Operativne podatkovne baze tudi nima smisla obremenjevati z zgodovinskimi podatki, ki na tem nivoju običajno niso potrebni. Podatkov o prodaji v enem letu je ogromno, vendar jih na blagajni trgovine ne potrebujejo. Zato je smotrno operativno bazo razbremeniti in po določenem času operativne podatke prenesti v podatkovno skladišče. Tudi s finančnega vidika je nesmotrno imeti dovolj zmogljivo strojno in programsko opremo, ki bi uspešno obvladovala velike množice podatkov, na operativnem nivoju.

Podatkovno skladišče

Podatkovno skladišče torej ni, kot bi utegnili sklepati iz imena, neko odlagališče nepotrebnih podatkov, temveč rešuje problem, ki ga ima danes mnogo organizacij: *gore podatkov, ki jih pa ne moremo uporabiti.*

Podatkovno skladišče (angl. Data Warehouse) je podatkovni vir, ki je:

- integriran – vsebuje podatke o vseh vidikih dejavnosti organizacije,
- organiziran po poslovnih področjih, to je okrog glavnih entitet podjetja,
- vsebuje zgodovinske podatke, ki so pomembni za poslovne analize, zato ima skladišče tudi časovno dimenzijo (podatki so točni glede na časovni trenutek, zato ponavadi vsebujejo zaznamek časa),

- nespremenljiv (podatkov v glavnem ne posodabljamo),
- vsebuje detajlne (podrobne) in sumarne (zbirne) podatke.

Podatkovno skladišče je torej namenjeno podpori odločanju. Ker podatki v skladišču "pokrivajo" celotno poslovanje, ga uporabljamo predvsem za odločanje na strateškem nivoju. Tudi struktura (organizacija) podatkov je taka, da ni prirejena za delo posameznega oddelka. Kot bomo videli v nadaljevanju, običajno za posamezna poslovna področja razvijemo področna podatkovna skladišča, ki so podmnožice skladišča. To pa tudi pomeni, da je podatkovno skladišče "last" celotne organizacije in ne posameznega oddelka ali skupine uporabnikov. Zagotavlja torej dostop do organizacijskih podatkov neposredno ali preko področnih podatkovnih skladišč.

Ker podatkovno skladišče vsebuje podatke iz mnogih raznolikih notranjih in zunanjih podatkovnih virov, moramo pri prenosu podatkov v skladišče poskrbeti za integracijo in transformacijo podatkov, na primer:

- Poenotenje identifikatorjev. Na primer, na enem oddelku (operativni bazi) na drugačen način dodeljujejo šifre kupcem kot na drugem. Zgodi se lahko celo, da imata različna kupca enako šifro ali obratno.
- Poenotenje oblik podatkov. V organizaciji datume hranimo v "evropskem formatu" (24.8.2000), uporabljamo pa tudi podatke iz podatkovne baze, dostopne preko Interneta, kjer so datumi zapisani v "ameriškem formatu" (8/24/00).
- Za isti podatek imamo več virov (podvajanje podatkov v operativnih podatkovnih virih), v katerih pa se lahko vrednost podatka razlikuje. Ker transformacija podatkov v podatkovno skladišče poteka avtomatizirano, moramo definirati pravila, po katerih bo izbran boljši vir za ta podatek.

Proces prenosa podatkov v skladišče se ponavadi izvaja enkrat dnevno (ponoči), včasih pa še bolj redko, saj za analize ni potreben dostop do trenutnih podatkov. Na primer, podatek o trenutni prodaji ni bistven za analizo prodaje nekega izdelka.

Problem podatkovnih skladišč je količina podatkov, predvsem zaradi potrebe hranjenja zgodovinskih podatkov. Tako lahko podatkovno skladišče neke trgovske

družbe s podatki o prodaji v vseh njenih trgovinah obsega nekaj sto milijonov zapisov, kar ustreza dvajsetim giga bytom[10]. To pa seveda pomeni, da je za uporabo podatkovnih skladišč potrebna dovolj zmogljiva programska in strojna oprema.

Področno podatkovno skladišče

Zaradi velike količine podatkov in njihove še vedno precej zapletene strukture je skladišče za neposredno uporabo analitikov manj primerno. **Področno podatkovno skladišče** (angl. Data Mart) je podatkovni vir prirejen za uporabo v sistemih za podporo odločanju za posamezna poslovna področja (finance, prodaja, trženje ...). Oddelki so potemtakem "lastniki" področnih podatkovnih skladišč in jih povsem nadzorujejo.

Edini legitimni vir podatkov za področno podatkovno skladišče je podatkovno skladišče, saj so podatki tam že integrirani in prečiščeni. Je torej podmnožica skladišča podatkov, vendar so podatki nekoliko prirejeni; ker analitiki pogosto dostopajo neposredno do področnega podatkovnega skladišča, mora le-ta:

- imeti strukturo enostavno za razumevanje; praviloma so podatki organizirani večdimenzionalno, kar omogoča poljubne poglede na podatke,
- vsebovati že izračunane sumarne podatke, kar omogoča hitrejše poizvedovanje.

Po (Gartner Group, 1999) imajo področna podatkovna skladišča za razliko od podatkovnega skladišča tudi nekatere pomanjkljivosti, ki jih poznamo že iz preteklih obdobij, saj so zaradi prilagojenosti določeni uporabi manj fleksibilne. Hkrati so stroški razvoja področnih podatkovnih skladišč visoki (Gartner Group, 1999), zato je pred začetkom razvoja nujno izdelati natančno analizo potreb po področnih podatkovnih skladiščih.

---

10 Izračun je narejen (Kimball, 1996) na predpostavkah, da ima podjetje 300 trgovin, ki poročajo dnevno o prodaji posameznega izdelka. V vsaki trgovini prodajo dnevno 3.000 *različnih* izdelkov.

Operativna shramba podatkov

Čeprav imata podatkovno skladišče in operativna shramba podatkov skupen cilj, to je integracijo podatkov, se po nekaterih drugih značilnostih ta dva podatkovna vira med seboj razlikujeta. Operativna shramba podatkov (angl. Operational Data Store – ODS) pokriva potrebe po integriranem podatkovnem viru na operativnem nivoju. Pogosto je namreč tudi za odločanje na operativnem nivoju potreben integriran podatkovni vir. Te potrebe ponavadi nastanejo, ko se sprejmejo strateške odločitve z uporabo podatkovnih skladišč in področnih podatkovnih skladišč in je potrebno na njihovi podlagi začeti izvajati aktivnosti.

Bistvene razlike med operativno shrambo podatkov in podatkovnim skladiščem so (Inmon, Zachman, Geiger, 1997):

- V operativni shrambi podatkov so shranjeni trenutni in časovno bližnji podatki, v podatkovnem skladišču pa podatki iz daljšega časovnega obdobja.
- Operativna shramba podatkov vsebuje pretežno detajlne podatke, podatkovno skladišče pa je usmerjeno bolj v sumarne podatke.
- Skladišče običajno ne dovoljuje posodabljanja, shramba pa običajno ponuja omejene možnosti posodabljanja podatkov.
- Podatki se iz operativnih podatkovnih baz prenašajo v skladišče dnevno ali redkeje, v operativno shrambo podatkov pa zelo pogosto, mnogokrat kar v realnem času, to je sproti ob spremembah v operativnih podatkovnih bazah.

Operativna shramba podatkov ima torej dvojno naravo, saj združuje operativne potrebe in potrebe podpore odločanju. Zaradi tega je najbolj kompleksen tip podatkovnega vira.

To vrsto podatkovnega vira srečamo najbolj redko. Zelo koristen je v primeru slabe integriranosti operativnih podatkovnih virov.

Tokovi podatkov

S slike 4 je razvidno, da so za sistem podatkovnih virov značilni tudi številni tokovi podatkov. Podatke bolj ali manj pogosto prenašamo iz enega vira v drugega. Pri

tem je pomembno, da prenos podatkov *dodaja vrednost* in ni samemu sebi namen. Vsak premik in transformacija mora pomeniti kvalitativne spremembe v podatkih.

Tako gredo, na primer, pri prenosu iz operativnih podatkovnih virov v podatkovno skladišče skozi **transformacijski in integracijski sloj**. To je kompleksna programska oprema, ki podatkom dodaja vrednost, tako da jih:

- integrira,
- prečisti in poenoti obliko,
- transformira,
- analizira, t.j. ugotavlja neveljavne vrednosti,
- preverja kakovost podatkov (pravilnost in popolnost),
- izračuna nove (predvsem sumarne) podatke,
- drugače strukturira …

Tudi pri prehodu podatkov iz skladišča v področna podatkovna skladišča se le-ti preoblikujejo: spremeni se struktura, ki postane bolj preprosta (ponavadi večdimenzionalna), izračunajo se novi sumarni podatki …

Pri zajemu podatkov iz zunanjih virov jim spremenimo strukturo in prilagodimo naši notranji strukturi podatkov, spremenimo obliko in podobno.

Če pri toku podatkov ne ugotovimo dodane vrednosti, je verjetno eden od obeh podatkovnih virov, med katerima prenašamo podatke, odveč.

Opisovanje podatkov

Ker se razumevanje pomena podatkov znotraj organizacije, še zlasti pa v medorganizacijski izmenjavi podatkov, lahko bistveno razlikuje, je pomembno, da vsak podatek ustrezno opišemo. Opis naj obsega najmanj naslednje elemente:

- **Poimenovanje**. Vsak tip podatka naj ima enolično ime (oznako). Pri tem je s tipom podatka mišljena množica istovrstnih podatkov, na primer EMŠO (vsi državljani imajo to lastnost, vendar se vrednosti razlikujejo), ime zaposlenega, plača … Pogosta napaka, ki jo srečujemo v praksi, so enaka

imena za različne tipe podatkov. Če nek tip podatka poimenujemo šifra, potem ni jasno, ali gre za šifro delavca, šifro izdelka, ali morda neko drugo šifro.

- **Oznaka**. Vsaka poslovna entiteta (objekt), ki je predmet našega zanimanja (o katerem hranimo podatke) naj ima enolično oznako, ki ga razlikuje od ostalih istovrstnih entitet (objektov). Na primer, avtomobile v Sloveniji lahko med seboj zanesljivo ločimo po registrski oznaki; brez le-te bi težko med seboj razlikovali dva avta istega modela, iz iste serije in enake barve.

- **Opredelitev**. Poleg enoličnega poimenovanja je pomemben tudi nedvoumen in podrobnejši opis vsake vrste podatkov, saj za razumevanje in obravnavo podatkov pogosto samo ime vrste podatka ne zadostuje.

- **Pravila celovitosti**. Za vsak podatek moramo opredeliti, kakšne so dopustne vrednosti. Na primer, dopustni vrednosti za spol sta "moški" in "ženski", dopuste vrednosti za vpisne številke na Ekonomski fakulteti v Ljubljani so osemmestna pozitivna cela števila, katerih prvi dve števki (cifri) sta 19 …

- **Pravice uporabe**. Določiti moramo tudi pravice dostopa do posameznega podatka: kdo sme dostopati, kaj sme s podatkom početi (ga prebrati, spremeniti, dodati …) in pod kakšnimi pogoji.

Z naraščanjem medorganizacijskega elektronskega poslovanja pa so se pojavili tudi problemi opredelitve strukture in opisa podatkov, ki si jih podjetja izmenjujejo na elektronski način, saj mora program, s katerim v enem podjetju zajemamo podatke, ki smo jih dobili od drugega podjetja, te podatke "razumeti". Na primer, če dobimo naročilo, moramo vedeti, kateri podatek pomeni naziv podjetja, kateri šifro izdelka, kateri količino … Razlog za opisani problem je seveda ta, da podjetja nimajo skupnega pogleda na podatke, enake organizacije podatkov.

Že pred leti so zato nastali standardi EDI (angl. Electronic Data Interchange), ki natančno določajo strukturo posameznih vrst dokumentov, ki si jih podjetja med seboj izmenjujejo. V zadnjih letih, ko pa se v medorganizacijsko elektronsko poslovanje vključuje čedalje več podjetij, ko se potrebe po vrstah podatkov, ki se izmenjujejo, čedalje večje in ko so sistemi, ki v tem poslovanju sodelujejo, čedalje

bolj heterogeni, so postali standardi EDI preveč omejujoči in nefleksibilni, tehnologija, ki se v tem okviru uporablja, pa je bila predraga in dostopna le velikim podjetjem. Zato se v zadnjem času vedno bolj širi uporaba standarda XML (angl. Extensible Markup Language), ki omogoča ustrezno označevanje poljubnih podatkov in njihovo izmenjavo preko Interneta. Več o izmenjavi podatkov med sistemi pišemo v poglavju Upravljanje podatkov.

Vprašanja za preverjanje razumevanja

- V dnevnem časopisu skušajte najti prispevke o primerih prikrivanja pravih podatkov v organizacijah in razmislite, kakšni bi lahko bili razlogi za to.
- Navedite primer neformalnega podatkovnega vira.
- Razmislite, ali ste že naleteli na primer, da ste morali v isti organizaciji posredovati isti podatek o sebi dvakrat (ali večkrat). Zakaj?
- V podjetju, ki ga poznate, opazujte zaposlene pri delu in skušajte ugotoviti, kdo je operativni uporabnik podatkov in kdo jih uporablja za podporo odločanju.
- Katere so bistvene razlike med uporabo podatkov na operativnem in na analitičnem nivoju?
- Kateri so problemi upravljanja podatkov in kateri so bistveni razlogi zanje?
- Kaj je okolje pajkove mreže? Bi lahko ocenili, da v podjetju, ki ga poznate, obstaja tako okolje?
- Kateri so poglavitni elementi (komponente) sodobno zasnovanega sistema podatkovnih virov?
- Kateri podatki so zgodovinski?
- Kakšen je pomen referenčnih podatkov?
- V okviru slovenskih spletnih strani skušajte najti čimveč podatkovnih virov, za katere menite, da bi jih v podjetju lahko uporabili za podporo poslovnemu odločanju. Na katerih področjih?
- Kaj so razlogi za uvedbo podatkovnega skladišča? Kakšen je namen podatkovnega skladišča?
- Kaj je področno podatkovno skladišče in kakšne so njegove značilnosti?
- Kaj je pomembno pri prenosu podatkov iz enega vira v drugega?
- Kaj so elementi opisa podatkov in zakaj je opis potreben?

Ključni pojmi

analitična obdelava
analitik
celovitost
deljenje podatkov
informacijska usmerjenost
informacijsko obnašanje
integracija
okolje pajkove mreže
operativna shramba podatkov
operativni nivo
operativni uporabnik
opisovanje podatkov
oznaka
podatkovna arhitektura
podatkovni vir
podatkovno skladišče
področno podatkovno skladišče
pravica uporabe
referenčni podatki
surovi detajlni podatki
tokovi podatkov
upravljanje podatkov
zgodovinski podatki
zunanji podatki

2. POGLAVJE

Upravljanje podatkov

Cilji poglavja

V tem poglavju boste spoznali:

- Da moramo vedeti, katere podatke zbiramo in zakaj, ter jih vzdrževati in organizirati tako, da podpiramo deljenje in ponovno uporabo.
- Da so aktivnosti upravljanja podatkov enake kot aktivnosti, povezane z upravljanjem drugih virov organizacije.
- Da je upravljanje podatkovnih virov dolžnost vsakega managerja. Pri tem so le redka aktivnosti upravljanja podatkov take, ki zahtevajo poglobljeno poznavanje tehnologije.
- Katere so aktivnosti upravljanja podatkov?
- Zakaj je pomembno načrtovanje podatkovnih virov?
- Kaj je podatkovno modeliranje, čemu služi in kako poteka?
- Kaj je model entitet-povezav in zakaj je njegova uporaba tako razširjena?
- Kaj je znanje in kakšni so problemi njegovega upravljanja?
- Kako lahko uporabimo informacijsko tehnologijo pri upravljanju znanja?

2.1 Življenjski cikel upravljanja podatkov

Upravljanje podatkov daje večji poudarek kot tehnologiji uporabi, relevantnosti in kakovosti podatkov. Podatkov ne obravnavamo zgolj kot množico objektov, pač pa skozi proces, ki predstavlja **življenjski cikel upravljanja podatkov**. Namen tega procesa je izboljšanje uporabe podatkov pri poslovnem odločanju, ugotavljanje informacijskih potreb, zmanjšanje zbiranja nerelevantnih podatkov, izogibanje preobremenjenosti s podatki in ponovnemu zbiranju podatkov ter z vsem tem varčevanju s časom in viri. Faze življenjskega cikla upravljanja podatkov so prikazane na sliki 6.



Slika 6: Življenjski cikel upravljanja podatkov

Zaznavanje pomeni ugotavljanje oziroma odkrivanje dogodkov in stanj v okolju organizacije, ki bi bili za organizacijo zanimivi. Spremljati moramo dogodke, trende, spremembe v pogojih poslovanja, ki bi utegnile odločilno vplivati na poslovne odločitve. Zaznati moramo gospodarske, socialne, politične spremembe, opazovati tekmece, ugotavljati spremembe na trgu, zahteve po novih izdelkih, predvidevati težave pri odnosih z dobavitelji in podobno.

Zaznavanje je podlaga za definiranje novih podatkovnih potreb, kar ima za posledico sistematično **zbiranje podatkov**. Pri zbiranju moramo biti pozorni na nekatera temeljna vprašanja:

- *Kdo potrebuje podatke*? Razumeti moramo odgovornosti zaposlenih na različnih nivojih in pri različnih aktivnostih.
- *Katere vrste podatkov potrebuje in koliko*? Pogosto v organizacijah zbirajo precej več podatkov, kot jih potrebujejo, kar ima za posledico podatkovno preobremenjenost. Razloge gre iskati v razvoju informacijske tehnologije, slabo opredeljenih poslovnih procesih in slabih formalnih informacijskih sistemih, ki ne zagotavljajo potrebnih informacij, kadar jih zaposleni potrebujejo za sprejemanje odločitev.
- *Kje so podatki*? Pogosto so podatki, ki jih zbiramo, že na voljo nekje v organizaciji, vendar tega ne vemo ali pa ne vemo, kje in kako jih lahko dobimo. Zato je potreben "podatkovni katalog" (glej Definiranje in opis v nadaljevanju).
- *Kdo in zakaj zbira podatke*? Zaposleni morajo razumeti, zakaj zbirajo podatke, ki jih ne bodo neposredno potrebovali, saj jih s tem motiviramo, da to počno natančno in popolno. Vedo naj, kdo je neposredno odvisen od podatkov, ki jih zbirajo. Pogosto se namreč dogaja, da tisti, ki bi morali zbirati podatke, tega ne počno natančno, ker ne razumejo, čemu je to potrebno. Vsak bi moral vedeti, kdo je od njega informacijsko odvisen in kako z dobrim zbiranjem podatkov prispeva k uspešnemu opravljanju nalog. Pri procesni organiziranosti, kjer zaposleni bolje razumejo celoten proces, v katerem sodelujejo s svojimi aktivnostmi, je tovrstnih težav praviloma manj.

Z **organiziranjem podatkov** poskrbimo, da so podatki dosegljivi čez meje posameznih oddelkov skozi celotne poslovne procese in z uvajanjem elektronskega poslovanja še dlje do dobaviteljev in kupcev. Pogoj za to je, da vsi enako razumemo pojme kot so kupec, naročilo, plačilo, sicer so podatki neuporabni. Organizacija podatkov naj bo torej taka, da vzpodbuja deljenje podatkov.

Z **obdelavo podatkov** razumemo doseganje in analiziranje podatkov, da dobimo uporabne informacije za podporo poslovnim odločitvam. Več o tem v poglavju Informacijska podpora managementu.

Vzdrževanje podatkov pa podpira ponovno uporabo obstoječih podatkov in izogibanje ponovnemu zbiranju. Podatki morajo biti zato redno posodabljani, ustrezno organizirani in dostopni, izgrajena pa mora biti tudi kultura deljenja podatkov, saj se ljudje pogosto izogibajo uporabi podatkov, ki jih niso zbrali sami.

2.2 Aktivnosti upravljanja podatkov

Aktivnosti upravljanja podatkov so enake kot aktivnosti, povezane z upravljanjem katerihkoli drugih virov organizacije, seveda pa je njihovo izvajanje glede na značilnosti podatkov kot vira lahko drugačno:

- načrtovanje,
- določitev vira,
- pridobivanje in vzdrževanje,
- opredelitev in opis (vodenje evidence),
- organizacija in dostopnost,
- nadzor kakovosti in celovitosti,
- varovanje in ščitenje,
- obračun uporabe,
- obnavljanje in nadgradnja,
- odločanje o obdržanju in odstranitvi ter
- izobraževanje in svetovanje za učinkovito uporabo.

Načrtovanje

Pri načrtovanju podatkov izhajamo iz sedanjih in bodočih potreb organizacije po podatkih, njihovi medsebojni povezanosti, načinov njihove uporabe, količine … Kot za vsako načrtovanje, tudi tu v večini primerov (razen v najbolj preprostih) izdelamo načrte na različnih nivojih podrobnosti. Na nivoju organizacije najprej izdelamo globalni podatkovni načrt, ki vsebuje najpomembnejše entitete in

povezave med njimi (glej poglavje Načrtovanje podatkovnih virov), nato pa nadaljujemo z načrtovanjem do vseh podrobnosti. Podatkovno načrtovanje je seveda soodvisno od načrtovanja drugih delov informacijskega sistema, na primer, načrtovanja zmogljivosti računalniškega sistema, načrtovanje programske opreme in podobno.

Viri

Za vsako vrsto podatkov moramo določiti njihov vir. V nekaterih primerih imamo več možnosti, kje bomo zajemali podatke. Seveda bomo v tem primeru izbrali le en vir, saj bomo vsak podatek zajeli le enkrat. Pri izbiri moramo upoštevati kakovost vira (npr. povprečno število napak, ki se pojavi pri zajemu), možnost pravočasnega zagotavljanja podatkov, ceno vira, potrebo po izobraževanju, natančnost, nivo podrobnosti … Če imamo možnost, izberemo za podatek njegov primarni vir.

Pridobivanje in vzdrževanje

Ko smo identificirali vire podatkov, moramo zgraditi sisteme za zajemanje podatkov, ki so lahko v svoji pojavni obliki raznoliki. Lahko gre za avtomatsko zajemanje podatkov, prebiranje črtne kode, zaslonske maske (forme) za vnos podatkov preko tipkovnice, prenos iz drugih podatkovnih virov …

Vse prepogosto zanemarjamo vzdrževanje podatkov in podatkovnih virov, kar se odraža na uspešnosti iskanja podatkov, ko jih potrebujemo. Neurejenost, prevelik obseg nepotrebnih podatkov, isti podatkovni element, ki ima dve različni vrednosti hkrati in podobno so lahko pokazatelji pomanjkljivega vzdrževanja.

V okviru te aktivnosti moramo zagotavljati tudi, da je sprememba v podatku takoj vidna vsem uporabnikom. To je pomembno predvsem takrat, ko nimamo integrirane podatkovne baze, oziroma kadar imamo iz kakršnegakoli razloga podvojene podatke (vrednost istega podatkovnega elementa shranjena večkrat).

Opredelitev in opis

Da bi vedeli, s kakšnim podatkom imamo opravka in kaj lahko z njim počnemo, je izredno pomembno, da je vsak podatek dobro opredeljen (definiran) in opisan. Opredelitve in opis podatkov imenujemo s skupnim imenom **podatkovni slovar**. V podatkovnem slovarju hranimo

- imena in opise podatkovnih elementov,
- organizacijo podatkov,
- obliko podatkov (tekstovni, numerični, slika …),
- omejitve, na primer, kakšne so dopustne vrednosti za posamezni podatkovni element, kako so podatki med seboj odvisni in podobno,
- pravice dostopa do podatkov,
- količine podatkov …

Opisi, ki jih hranimo v podatkovnem slovarju, so seveda tudi podatki. Ker pa ti podatki opisujejo tiste "prave" podatke, ki so predmet našega zanimanja, jih imenujemo tudi podatki o podatkih ali **metapodatki**.

Organizacija in dostopnost

Podatki morajo biti organizirani tako, da bo dostop do njih učinkovit. Z drugimi besedami, uporabnik mora imeti takrat, ko podatke potrebuje, možnost priti do njih in to hitro ter v pravi obliki. Učinkovita organizacija podatkov se razlikuje v primeru, kadar imamo ali načrtujemo računalniško podprt sistem, od primera, ko podatke obdelujemo "ročno in papirno". Razlikujeta se tudi primer, ko imamo opravka s tradicionalnimi oblikami podatkov (tekstovni in numerični), in primer, ko delamo z večpredstavnimi podatki (slike, zvok, video …).

Večina dela v okviru te aktivnosti je precej *tehnične narave*, zato le v redkih preprostejših primerih organizacijo podatkov načrtuje uporabnik. Vendarle, organizacija je bistveno odvisna od pomena podatkov in njihove uporabe, zato *ne more biti dobro načrtovana brez sodelovanja tistega, ki pozna obravnavano poslovno področje.*

V nadaljevanju (glej poglavje Podatkovne baze) prikazujemo osnovna pravila dobre organizacije podatkov v računalniško podprti podatkovni bazi. Poznavanje teh pravil omogoča uporabniku samostojno izdelavo dobrih preprostih rešitev, hkrati pa omogoča razumevanje problematike in boljše sodelovanje s strokovnjakom pri skupnem načrtovanju organizacije podatkov.

Nadzor kakovosti in celovitosti

Kakovost podatkov lahko obravnavamo skozi naslednje štiri dimenzije (Turban, Aronson, 2001):

- kontekstna kakovost: relevantnost, popolnost, količina, časovni vidik ...,
- notranja kakovost: točnost, objektivnost, zaupanje v podatke ...,
- kakovost dostopnosti: dostopnost in varnost dostopa ter
- kakovost predstavitve: enostavno razumevanje, zgoščena predstavitev, konsistentna predstavitev ...

Ker podatkovne vire upravljamo neodvisno od njihove uporabe, je razumljivo, da mora biti nadzor kakovosti in celovitosti podatkov vgrajen v okvir njihovega upravljanja. Navadno nadzor opravimo že na vhodu, torej podatek, ki ne ustreza kriterijem, zavrnemo, še preden bi ga utegnili shraniti. Kljub temu je priporočljivo tudi občasno preverjanje celotnega podatkovnega vira.

V okviru te aktivnost ugotavljamo, na primer, ali

- je podatek pravilne oblike (poskus vnosa tekstovne vrednosti za numerični podatke),
- je morda tekstovni podatek predolg, numerični premalo natančen (premalo decimalnih mest),
- je zvokovna datoteka pokvarjena,
- vrednost podatka ne ustreza množici dopustnih vrednosti (poštna številka 123456, spol srednji, teža -30 kg in podobno),

- smo poskušali vnesti podatke o prodaji izdelka (šifro izdelka), ki ga sploh ne prodajamo; v tem primeru gre za t.i. referenčno celovitost – zapis o prodaji ne more vsebovati šifre izdelka, če nimamo med zapisi o izdelkih takega, ki bi imel to šifro.

Varovanje in zaščita

Z odpiranjem informacijskih sistemov (npr. povezovanjem v Internet) in s povečevanjem pomena podatkov za uspešno poslovanje, se je povečal pomen in tudi problemi povezani z ustrezno zaščito podatkov pred nepooblaščenim dostopom[11]. Varovanje podatkov ima dva vidika:

- poslovni in
- tehnološki.

Strokovnjaki lahko učinkovito uporabijo ustrezno tehnologijo varovanja, ki je na voljo, le, če ima organizacija ustrezno varnostno politiko. To pa pomeni, da je potrebno za vsak podatek določiti, kdo in kakšne pravice dostopa naj ima, potrebno stopnjo zaščite … Pomembno je, da vse to razmislimo že med načrtovanjem podatkovnih virov. Glede na politiko zaščite lahko razdelimo sisteme (organizacije) na štiri skupine:

- Uporabnik dobi pravico dostopa do podatkov, ko le-te potrebuje (**po potrebi**). Pravica je omejena na natanko tiste podatke, ki jih potrebuje.
- Podatki so v **največji možni meri dostopni uporabnikom**. To ne pomeni, da imajo vsi uporabniki pravico dostopa do vseh podatkov, vendar so posameznemu uporabniku nedostopni le tisti, ki res morajo biti nedostopni.
- **Odprti pristop zaščite podatkov**. Praviloma imajo uporabniki pravico dostopa do vseh podatkov, z izjemo tistih, ki so eksplicitno zaščiteni.

---

11 Čeprav so po nekaterih raziskavah (v ZDA) za večino zlorab podatkov odgovorne pooblaščene osebe znotraj organizacije, ki imajo pravico dostopa do teh podatkov.

- **Zaprti pristop zaščite podatkov**. Praviloma uporabniki nimajo pristopa do nobenega podatka, z izjemo tistih, za katere dobijo eksplicitno pravico. Očitno omogoča ta pristop boljšo zaščito, saj pri odprtem pristopu lahko pozabimo zaščititi določen del podatkovnega vira in s tem lahko vsi uporabniki dostopajo do teh podatkov.

Večinoma so pravice dostopa do podatkov vezane na posameznega uporabnika. Zaradi enostavnejšega vzdrževanja lahko uporabnike, ki imajo enake pravice, združimo v skupine in potem vežemo pravice na skupino.

Uporabnik ima lahko do posameznega podatka različne pravice dostopa in sicer gre za poljubno kombinacijo naslednjih vrst dostopa:

- definiranje,
- doseganje,
- vnos,
- brisanje ter
- spremembe.

Glede na pogoje, pod katerimi sme uporabnik dostopati do podatkov, ločimo naslednje vrste zaščite:

- vsebinsko odvisna zaščita,
- vsebinsko neodvisna zaščita,
- kontekstno odvisna zaščita in
- zaščita statističnih podatkovnih baz.

Pri **vsebinsko odvisni zaščiti** podatkov je dovoljenje za doseganje dela podatkovnega vira odvisno od vsebine (podatkov). Na primer, uporabnik Janez sme spreminjati vrednost atributa PLAČA, če je ta manjša od 50.000 denarnih enot.

Pri **vsebinsko neodvisni zaščiti** so pravice dostopa vnaprej določene (ne glede na vrednost) za vsakega uporabnika in vsak podatkovni element. Na primer,

uporabnik Janez sme brati vrednosti podatke o plačah. Ali, uporabnica Metka sme brati in spreminjati podatke o zaposlenih.

Pri **kontekstno odvisni zaščiti** je isti podatkovni element v določenem kontekstu dostopen posameznemu uporabniku, v drugem kontekstu pa (istemu uporabniku) ne. Na primer, prodajalec lahko spreminja podatke o prodaji le v času, ko je trgovina odprta. Dostopnost torej ni odvisna od vsebine (npr. vrednosti podatkov o prodaji ali imen izdelkov), temveč od ostalih, zunanjih glede na vir, okoliščin (časa).

V kontekstu varnosti in zaščite podatkov govorimo tudi o **statističnih podatkovnih bazah**. Statistične podatkovne baze omogočajo uporabniku, da izvaja na podatkih o posameznih objektih le statistične operacije, kot na primer vsota, povprečje. Z drugimi besedami, uporabnik lahko dobi le sumarne podatke, ne pa tudi individualnih podatkov, to je podatkov o posameznikih. Na primer, raziskovalec v zdravstveni organizaciji ne more izvedeti podatkov o boleznih za posameznega pacienta. Omogočene so mu samo statistične raziskave. Seveda lahko na isto podatkovno bazo gledamo tudi kot na ne-statistično, saj lahko osebni zdravnik pacientov pregleduje njihove individualne podatke.

Žal pa ni dovolj, da uporabniku ne dovolimo doseganja podatkov o posameznikih, saj je mogoče iz več sumarnih podatkov sklepati na podatke o posameznikih.

Uporabnik Miha raziskuje gibanje plač v podjetju. Ker ne želimo, da bi uporabnik Miha poznal podatke o plačah posameznih zaposlenih, smo mu dodelili le pravico za statistične poizvedbe na podatkih o delavcih. Uporabnik Miha lahko, na primer, dobi odgovore na naslednje poizvedbe:

Povprečna plača zaposlenih, ki delajo na oddelku trženje.

Vsota plač zaposlenih, ki niso vodje oddelkov.

Ne more pa izvedeti:

Plača zaposlenega, ki ima šifro 3322.

Vendar lahko uporabnik Miha zastavi več zaporednih statističnih poizvedb in iz rezultatov sklepa na plačo posameznika. Na primer, iz odgovorov na naslednji poizvedbi:

> Vsota plač vseh zaposlenih.
>
> Vsota plač vseh zaposlenih, ki niso generalni direktorji.

z odštevanjem takoj lahko izračuna plačo generalnega direktorja podjetja.

Poznamo več algoritmov[12], ki omogočajo učinkovito zaščito statističnih podatkovnih baz pred pridobivanjem informacij s sklepanjem iz rezultatov dovoljenih statističnih poizvedb.

Obračun uporabe

Aktivnosti kot so načrtovanje, zajemanje, vzdrževanje podatkov, izdelava poročil so lahko povezane s precejšnimi stroški, ki morajo biti identificirani. Zaradi značilnost podatkov, da se z uporabo ne izrabljajo, je obračun uporabe posebej težek računovodski problem. Najmanj, kar lahko ugotovimo, so stroški povezani s hranjenjem in obdelavo podatkov.

Obnavljanje in nadgradnja

Zaradi različnih razlogov se nam lahko zgodi, da se podatkovni vir poškoduje. Magnetni trak se lahko razmagneti, računalnik, na katerem je shranjena podatkovna baza, uniči strela, arhiv z računi zgori v požaru … Za take primere moramo zagotoviti način, da lahko podatkovni vir obnovimo. Za računalniško podprto podatkovno bazo je to največkrat občasna izdelava **rezervne kopije** (angl. backup), hkrati pa (elektronsko) beležimo vsako aktivnost, ki je bila narejena na podatkovni bazi od prejšnje izdelave rezervne kopije. Na podlagi obojega lahko v primeru nesreče – poškodovanja baze le-to vrnemo v stanje pred nesrečo. Pogostost izdelovanja rezervne kopije je odvisna od varnostne politike organizacije in ne more biti prepuščena skrbniku podatkovne baze.

---

12 Njihov opis presega namen tega dela; pregledno so predstavljeni v (Adam, Wortmann, 1989).

Poleg tega, da vemo, kako bomo podatke rešili v primeru nesreče, moramo imeti tudi načrt, kako bomo ukrepali v času, ko podatki ne bodo dostopni, torej do končanja obnove vira. Poslovanje mnogih organizacij je neposredno odvisno od dostopa do podatkov. Lep primer je poslovanje bank in trgovine z računalniškimi blagajnami. V skrajnih primerih imajo nekatere organizacije ves čas dve enaki kopiji istega podatkovnega vira.

Nadzor celovitosti, varnost ter obnavljanje in nadgradnjo celostno obravnavamo kot zaščito podatkovnih virov. Pri tem z nadzorom celovitosti zaščitimo vir pred tem, da bi ga uporabnik nenamerno privedel v nekonsistentno stanje. Varnost zagotavlja zaščito pred namernimi poskusi oškodovanja. Z obnavljanjem in nadgradnjo pa odpravljamo posledice nesreč.

Odločanje o obdržanju in odstranitvi

S časom količina podatkov narašča. Po eni plati podatki ne bodo uporabni za vedno. Po drugi plati pa nam zasedajo in obremenjujejo zmogljivosti, na primer, arhivske sobe so prepolne, disk je prepoln, iskanje se zaradi količine podatkov podaljša … Zato se moramo odločiti, kako bomo ravnali z zgodovinskimi podatki.

Na voljo imamo različne možnosti:

- Podatke lahko arhiviramo v drugačni obliki. Papirne dokumente shranimo na mikrofilm ali poskeniramo in shranimo v digitalni obliki. Zgodovinske podatke iz računalniške podatkovne baze prenesemo na magnetni trak, kjer so podatki še vedno dostopni, v primeru, če jih bomo kdaj vendarle potrebovali. Seveda bo čas dostopa večji.
- Shranimo lahko samo sumarne podatke, detajlne podatke pa arhiviramo ali zavržemo. Na primer, po določenem času nas ne bodo zanimali več podatki o vsaki posamezni prodaji, pač pa le še podatki o skupni prodaji po dnevih, kasneje pa morda le še po mesecih.
- Nazadnje pa bomo podatke morda celo zavrgli.

Odločitev o tem, kdaj je čas za prehod iz ene v drugo obliko, ali za to, da podatke zavržemo, je seveda poslovna odločitev.

Hranjenje starih podatkov pa lahko pomeni poleg obremenjevanja zmogljivosti tudi nekakovostne podatke za analize – poslovno odločanje.

Izobraževanje in svetovanje za uporabo

Sam obstoj kakovostnih podatkov še ne zagotavlja tudi njihove učinkovite uporabe. Zato moramo uporabnike izobraževati, da bodo vedeli, kateri podatki so na voljo, kako lahko pridejo do njih, kako naj uporabljajo informacijsko tehnologijo za delo s podatki …

V skupnem načrtovanju stroškov, ki so vezani na podatkovni vir, se stroški izobraževanja vse prepogosto podcenjujejo, kar ima za posledico premalo izobraževanja in svetovanja uporabnikom in v nadaljevanju neučinkovito uporabo virov. S tem pa lahko postane vprašljiva tudi upravičenost vloženih sredstev v kakovostni podatkovni vir.

2.3 Načrtovanje podatkovnih virov

Med izgradnjo podatkovnega vira in izgradnjo hiše lahko naredimo lepo primerjavo.

Vsak, ki je že imel priložnost videti hišo, v kateri moramo skozi spalnico, če želimo priti v kuhinjo[13], se bo strinjal, da je pred začetkom gradnje potrebno izdelati načrt. Sicer lahko naredimo tudi hišo brez načrta, ki bo imela vse potrebne elemente (stene, tla, vodovodne napeljave, električne napeljave, luči …), a v večini primerov ne bo taka, kot bi morala biti. Morda nam bo v kuhinji zmanjkalo samo 10 centimetrov za postavitev hladilnika, ki sicer po dimenzijah ustreza standardu.

Za preprostejše hiše si lahko investitor sam izdela načrt, ki bo povsem zadovoljiv. V večini primerov pa za izdelavo projektov najamemo arhitekta. Čeprav je arhitekt strokovnjak prav za izdelavo projektov, le-teh ne more izdelati brez sodelovanja investitorja. Ta mu mora povedati želje in možnosti, ki jih ima. Ali želi imeti tri otroške sobe? Morda bazen na vrtu? Naj bo hiša moderno opremljena ali v rustikalnem stilu? Koliko sredstev ima investitor na voljo? Na taka in podobna vprašanja mora arhitekt dobiti odgovor, preden začne načrtovati. In na podlagi odgovorov bo izdelal projekte, ki željam investitorja najbolj ustrezajo, hkrati pa ustrezajo zahtevam stroke (funkcionalnost, estetika …).

---

13 Avtorjeva osebna izkušnja.

Povsem enako kot v primeru hiše velja tudi za podatkovni vir. Pred gradnjo podatkovnega vira morajo strokovnjaki – informatiki izdelati načrt, ki pa ne more nastati brez predhodne analize uporabnikovih potreb, saj le uporabnik dovolj dobro pozna problemsko področje, ki ga želimo predstaviti s podatki v podatkovnem viru. Povezave in odvisnosti med objekti in podatki, ki te objekte opisujejo, odražajo poslovna pravila organizacije, ki jih najbolje pozna prav uporabnik.

Pri razvoju podatkovnega vira (na primer, podatkovne baze) torej fazi analize problemskega področja sledi faza načrtovanja. V skladu s tema fazama poznamo tudi dve *specializaciji dela informatikov*:

- **analitik** spoznava problemsko področje, ugotavlja, kateri podatki nas zanimajo, povezave in druge odnose med podatki …
- **načrtovalec** na podlagi ugotovitev analitika izdela načrt podatkovnega vira.

V splošnem v fazi analize sistema analitik ugotavlja:

- o katerih stvareh (objektih, dogodkih …) nameravamo hraniti podatke v podatkovnem viru;
- kako so te stvari med seboj povezane, saj moramo tudi informacijo o povezavi opisati v podatkovnem viru;
- katere podatke o teh rečeh želimo hraniti v podatkovni bazi;
- kako so ti podatki med seboj povezani (odvisnost med podatki), kar ima vpliv na organizacijo podatkov;
- poslovna pravila, ki vežejo (omejujejo) podatke.

Najpomembnejši aktivnosti analitika v fazi analize sta zbiranje obstoječe dokumentacije ter pogovori z uporabniki (intervjuji). Dokumentacija, ki jo potrebuje, med drugim obsega:

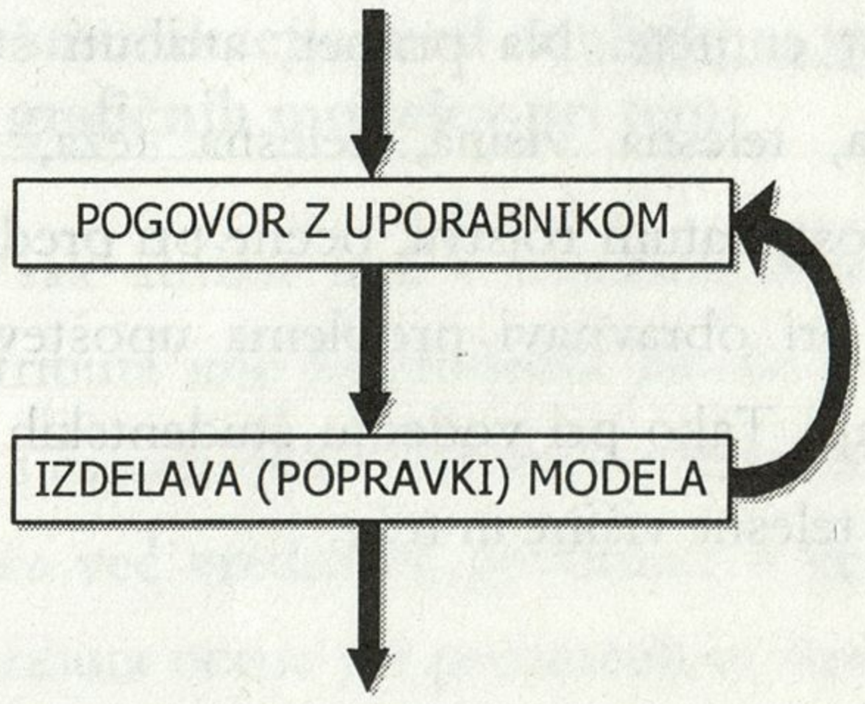
- dokumente, na katerih smo do sedaj hranili podatke, ter
- načrt morebitnega dosedanjega vira podatkov (podatkovne baze).

Od uporabnikov bo skušal zbrati vse informacije, ki jih bo potreboval načrtovalec za izdelavo načrta podatkovnega vira. Rezultat pogovorov je v največ primerih besedni opis problemskega področja.

Besedni opisi pa imajo nekatere slabosti, ki povzročajo težave pri načrtovanju:

- pogosto so premalo natančni, ne opredeljujejo vseh podrobnosti,
- mnogokrat vsebujejo opise, ki niso pomembni za izdelavo podatkovnega vira,
- opisi različnih uporabnikov si med seboj nasprotujejo…

Zato analitik izdela formalni zapis ugotovitev – model problemskega področja, v katerem ugotovi morebitne pomanjkljivosti zbranih opisov, morebitna protislovja … Ta formalni zapis nato prouči skupaj z uporabniki in ga po potrebi popravi in dopolni. V večini primerov izvedemo več iteracij postopka, kar je prikazano na sliki 7.



Slika 7: Analiza je iterativen postopek

Ker model proučujeta skupaj analitik in uporabnik, mora biti način prikaza enostaven in razumljiv tudi neinformatiku. Zato na tem mestu uporabljamo tehnike, ki so relativno preproste za razumevanje in uporabljajo grafično predstavitev.

Model entitet-povezav

Eden izmed v praksi najpogosteje uporabljanih tehnik, s katerimi grafično prikažemo podatkovni pogled na obravnavani del realnega sveta, je zagotovo

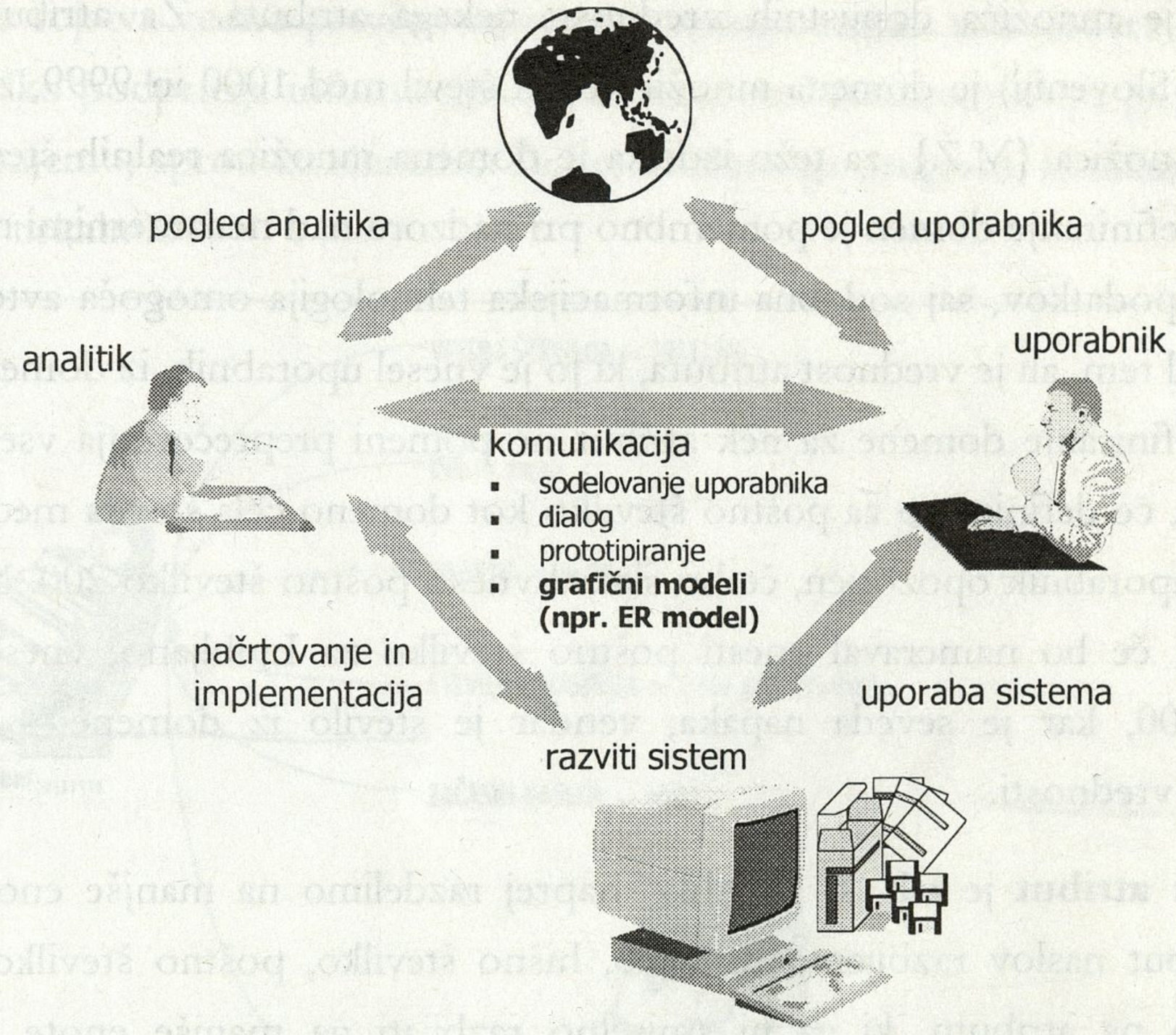
model entitet-povezav (angl. Entity Relationship model). Ker v praksi največkrat srečamo kratico, ki izhaja iz angleškega imena, bomo tudi mi uporabljali krajši izraz ER model.

Ker je ER model relativno enostavno razumeti, hkrati pa nudi dovolj široko paleto možnosti formalnega (nedvoumnega) izražanja strukture podatkov in poslovnih pravil, je razumljiva njegova pogosta uporaba. Zaradi omenjenih lastnosti je posebej pomemben pri komunikaciji med analitikom in uporabnikom, saj na ta način zbližamo pogleda uporabnika in analitika na obravnavano problemsko področje.

Poglejmo si torej, kako z ER modelom prikažemo podatkovni vidik obravnavanega dela realnega sveta.

Entiteta je predmet ali dogodek, ki ga obravnavamo. Primeri entitet so delavka Ana z EMŠO 0101900500000, študent Janez z vpisno številko 19111111, izpitni rok pri predmetu Informatika 2 dne 1.1.2002, naročilo številka 324 …

Atribut je neka lastnost entitete. Na primer, atributi študenta Janeza so ime, priimek, vpisna številka, telesna višina, telesna teža, naslov, slika, poročen (vrednost da ali ne), starost, datum rojstva, ocene pri predmetih, številka mobitela … Opozoriti velja, da pri obravnavi problema upoštevamo le atribute, ki so predmet našega zanimanja. Tako pri vodenju študentskih evidenc na univerzi ne upoštevamo kot atributa telesne višine in teže.



Slika 8: Pomen komunikacije med analitikom in uporabnikom in uporaba grafičnih modelov pri tem

Vrednost atributa. Vsak atribut ima v *določenem trenutku* nobeno, eno ali več vrednosti. Vrednost atributa ime za študenta Janeza je seveda 'Janez'. Vrednost atributa telesna teža se s časom lahko bistveno spreminja (praviloma z leti narašča). Kadar ima atribut *lahko* več vrednosti, govorimo o **večvrednostnem atributu**. Taka sta, na primer, atributa ocene pri predmetih in številka mobitela. V primeru, ko v določenem trenutku nek atribut nima vrednosti (ali je ne poznamo), to označimo z vrednostjo **NULL**[14]. Vrednosti NULL pa ne smemo zamenjevati z vrednostjo 0 (v primeru številskih podatkov) ali vrednostjo prazen niz '' (v primeru tekstovnih podatkov), saj v tem primeru ne gre za manjkajoče vrednosti.

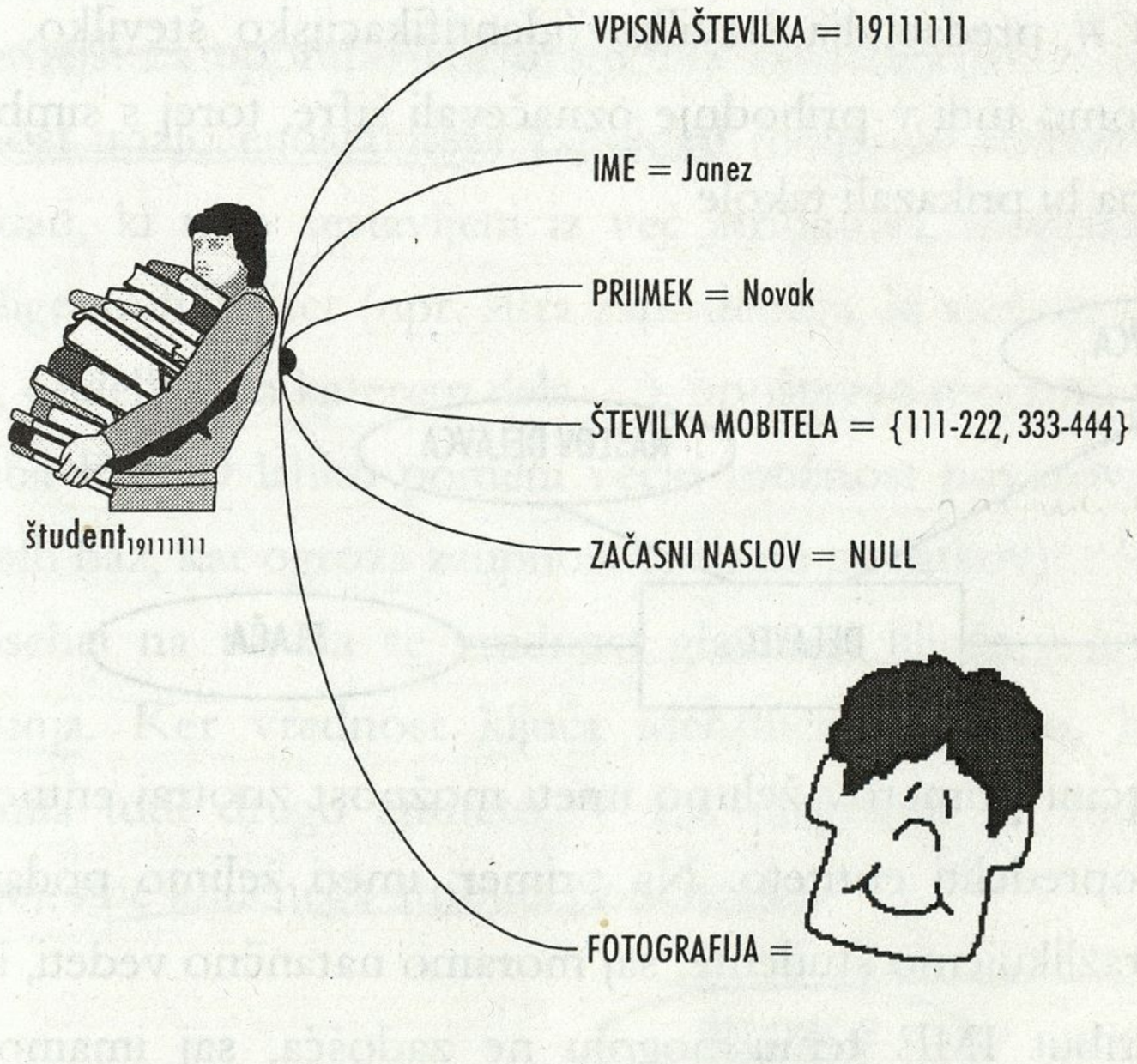
14 V programskih paketih za statistično analizo je vrednost, ki predstavlja manjkajočo vrednost, pogosto označena z MISSING VALUE.

Domena je množica dopustnih vrednosti nekega atributa. Za atribut poštna številka (v Sloveniji) je domena množica celih števil med 1000 in 9999, za spol je domena množica {M,Ž}, za težo izdelka je domena množica realnih števil večjih od 0 … Definiranje domen je pomembno pri nadzoru nad nenamernimi napakami pri vnosu podatkov, saj sodobna informacijska tehnologija omogoča avtomatičen nadzor nad tem, ali je vrednost atributa, ki jo je vnesel uporabnik, iz domene ali ne. Vendar definiranje domene za nek atribut ne pomeni preprečevanja vseh napak. Na primer, če definiramo za poštno številko kot domeno cela števila med 1000 in 9999, bo uporabnik opozorjen, če bo skušal vnesti poštno številko 200. Ne bo pa opozorjen, če bo nameraval vnesti poštno številko za Ljubljano, vnesel pa bo število 2000, kar je seveda napaka, vendar je število iz domene – množice dopustnih vrednosti.

Sestavljen atribut je tak, ki ga lahko naprej razdelimo na manjše enote. Tako lahko atribut naslov razbijemo na ulico, hišno številko, poštno številko in kraj. Nasprotno pa atributu, ki ga ni smiselno razbijati na manjše enote, pravimo **elementarni atribut**. Poštne številke ne moremo razbijati na manjše enote, zato je le-ta elementarni atribut. V različnih primerih je lahko isti atribut enkrat elementarni atribut, drugič pa ne. Tako lahko ime in priimek včasih obravnavamo kot enovit podatek (elementarni atribut), drugič pa kot dva atributa. Odločitev o načinu obravnave lahko posledično pomeni razlike pri kasnejšem iskanju po podatkih in vzdrževanju podatkov.

Izpeljani atribut je tisti, katerega vrednosti lahko izpeljemo (izračunamo) iz ostalih atributov. Na primer, povprečno oceno študenta lahko izračunamo iz njegovih ocen, starost lahko izračunamo iz datuma rojstva … Ker jih lahko v vsakem trenutku izračunamo (in računalnik to praviloma počne hitro), jih ne rabimo shranjevati neposredno. Načeloma želimo vsako dejstvo shraniti na en sam način. Če shranimo ocene študenta in še njegovo povprečno oceno, je povprečna ocena shranjena na dva načina. V primeru starosti in datuma rojstva sta oba atributa enakovredna, saj iz enega lahko izračunamo drugega in obratno (če je le starost shranjena dovolj natančno). Vendar se bomo odločili za hranjenje datuma rojstva, saj je starost podatek, ki ga moramo neprestano spreminjati, ali pa postane neažuren. Če pa hranimo datum rojstva, dobimo z odštevanjem trenutnega datuma

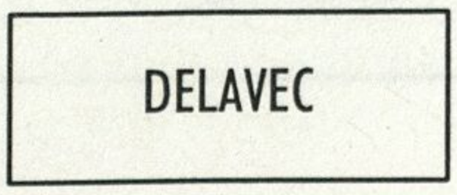
in datuma rojstva vedno pravilno starost. Skupnega zneska računa torej v primeru računalniško podprtega informacijskega sistema ne hranimo, pač pa ga vedno, ko ga potrebujemo, sproti izračunamo. Bolje, izračuna ga program, ki račun izpiše na zaslon ali tiskalnik.



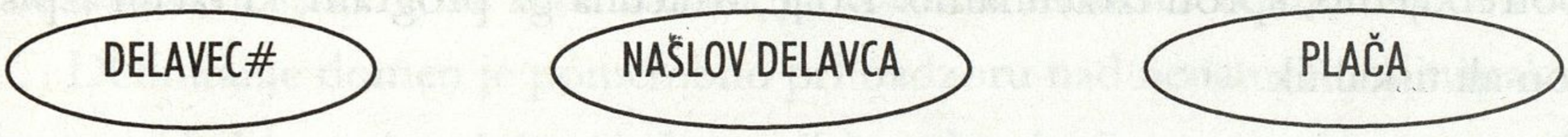
Slika 9: Primer entitete, atributov in njihovih vrednosti

Entitetni tip predstavlja vzorec za entitete z enakimi (ali vsaj podobnimi) atributi. Vsi študenti so entitetnega tipa ŠTUDENT. Vsa naročila pa lahko predstavimo z entitetnim tipom NAROČILO. Pri načrtovanju podatkov večinoma ne obravnavamo vsake entitete posebej, saj bi to pomenilo preveliko kompleksnost, hkrati pa tudi ni potrebno, pač pa proučujemo entitetne tipe in povezave med njimi. Kadar ne more priti do dvoumnosti, v praksi pogosto uporabljamo za entitetni tip kar krajši izraz entiteta.

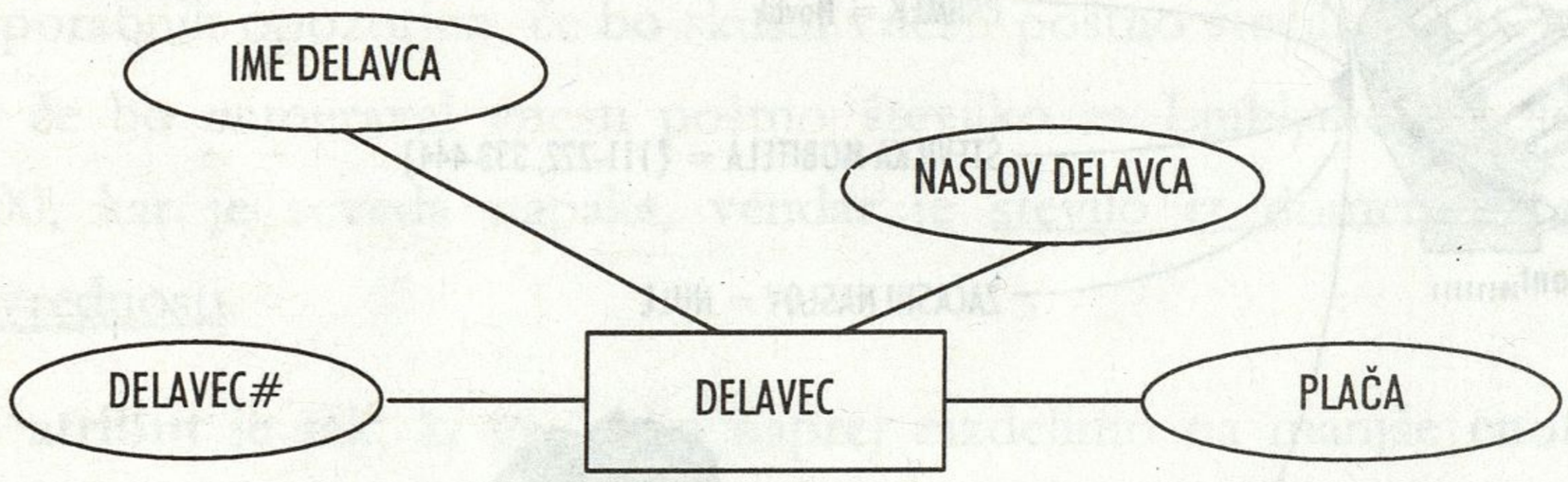
Entitetni tip grafično prikažemo s pravokotnikom, v katerega vpišemo ime le-tega, na primer



Za grafični prikaz atributov entitetnega tipa (skupnih atributov vseh entitet tega tipa) uporabljamo elipse, v katerih je zapisano ime atributa, na primer:



kjer postavka DELAVEC# predstavlja številko (identifikacijsko številko, šifro) delavca. Na enak način bomo tudi v prihodnje označevali šifre, torej s simbolom #. Entitetni tip z atributi pa bi prikazali takole



Kandidat za ključ. V večini primerov želimo imeti možnost znotraj entitetnega tipa s podatki enolično opredeliti entiteto. Na primer, imeti želimo podatek, s katerim lahko med seboj razlikujemo študente, saj moramo natančno vedeti, kdo je dobil kakšno oceno. Atribut IME temu pogoju ne zadošča, saj imamo zelo verjetno veliko študentov z enakimi imeni. Prav tako pogoju ne zadoščata skupaj ime in priimek, saj se neredko zgodi, da imamo dva študenta z enakim imenom in priimkom. *Atribut (ali več atributov skupaj), za katerega ne moreta imeti v nobenem trenutku nobeni dve entiteti istega entitetnega tipa enake vrednosti, imenujemo kandidat za ključ.* Kandidat za ključ enolično identificira entiteto znotraj entitetnega tipa. Ker je namenjen identifikaciji entitet, ne sme imeti vrednosti NULL. Pri zaposlenih v podjetju so kandidati za ključ, na primer, šifra delavca, EMŠO, davčna številka[15], morda pa tudi *kombinacija* imena, priimka in datuma rojstva. Opozorimo, da je EMŠO kandidat za ključ tudi znotraj entitetnega tipa vseh državljanov Slovenije, kar za šifro delavca ne velja, saj lahko v različnih podjetjih na enak način šifrirajo zaposlene. Šifre so umetni atributi, ki jih večinoma vpeljemo prav zato, da si zagotovimo obstoj vsaj enega kandidata za glavni ključ.

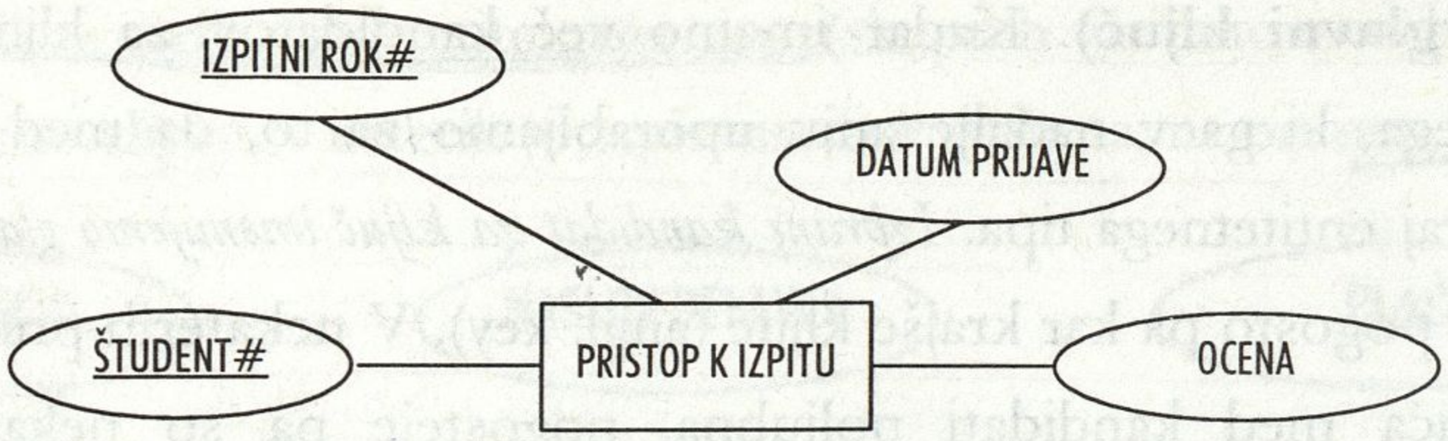
Ključ (tudi glavni ključ). Kadar imamo več kandidatov za ključ, med njimi izberemo enega, ki ga v nadaljevanju uporabljamo za to, da med seboj ločimo entitete znotraj entitetnega tipa. *Izbrani kandidat za ključ imenujemo glavni ključ* (angl. primary key), pogosto pa kar krajše ključ (angl. key). V nekaterih primerih je izbira glavnega ključa med kandidati poljubna, pogosteje pa so nekateri kandidati primernejši za uporabo kot drugi. Na odločitev lahko vplivajo različni parametri: prednost imajo enostavnejši kandidati (boljši so atributi s krajšimi vrednostmi in kandidati, ki niso sestavljeni iz več atributov), izogibamo se uporabi govorečih ("inteligentnih") šifer (npr. šifra zaposlenega, ki vsebuje podatek o datumu rojstva, spolu, oddelku, na katerem dela …), upoštevati moramo varnostne vidike (pogosta uporaba EMŠO lahko pomeni večjo možnost povezovanja osebnih podatkov iz različnih baz, kar ogroža zaupnost osebnih podatkov) … Pri izbiri bodimo pozorni še posebej na to, da se vrednost glavnega ključa v življenjski dobi entitete ne spreminja. Ker vrednost ključa identificira entiteto, bi spremenjena vrednost pomenila tudi drugo entiteto. V ER diagramu (in tudi sicer, kot bomo videli kasneje), ime ključnega atributa podčrtamo.



Sestavljeni ključ. Nekaterih entitet ni mogoče enolično opredeliti z enim samim atributom oziroma z glavnim ključem, ki je en sam atribut. Takrat uporabimo glavni ključ, ki je sestavljen iz dveh ali več atributov. Takšen ključ imenujemo sestavljeni ključ (angl. concatenated key). Če imamo, na primer, entitetni tip PRISTOP K IZPITU, katerega elementi so dogodki, ki ponazarjajo pristope študentov na izpitni rok, potem bi lahko imeli sestavljen glavni ključ, ki ga sestavljata atributa ŠTUDENT# in IZPITNI ROK#. Nobeden od obeh atributov sam zase ne definira enolično pristopa k izpitu, saj študent pristopi na več izpitnih rokov, na en izpitni rok pa več študentov. V ER diagramu lahko prikažemo sestavljeni ključ tako da podčrtamo imena vseh atributov, ki sestavljajo glavni ključ, na primer:

---

15 Zanimiva kandidata za ključ sta tudi fotografija delavca in pristni odtis, vendar ju le redko uporabljamo zaradi zapletenih postopkov za njihovo prepoznavanje. Z razvojem informacijske tehnologije se bo njihova uporaba za identifikacijo in razlikovanje prav gotovo razširila.

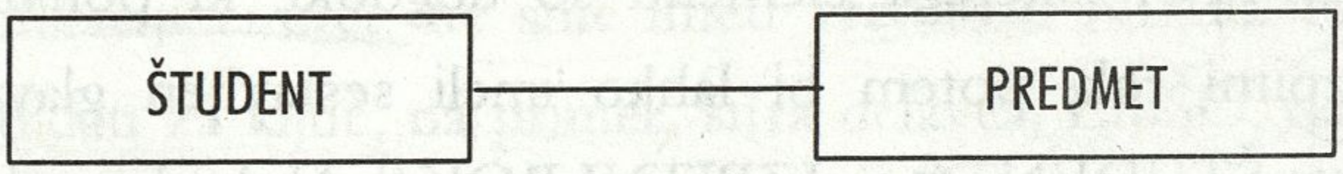


ali tako, da v eni elipsi zapišemo imena vseh sestavnih delov ključa in jih seveda prav tako podčrtamo:



Povezava. Za analizo problemskega področja z vidika podatkov je pomembno tudi ugotavljanje povezav med entitetami. Če je delavka Ana z EMŠO 0101900500000 zaposlena na oddelku Trženje, sta ti dve entiteti povezani, nista pa povezani entiteti delavka Ana z EMŠO 0101900500000 in oddelek Proizvodnja 1. Mogoče so tudi povezave med več entitetami. Na primer, kupec številka 123 je pri nakupu številka 456 kupil izdelek s šifro 789.

Tip povezave. Tako kot v primeru entitet pri načrtovanju ne modeliramo posamezne entitete, tudi tu ne analiziramo povezav med posameznimi entitetami, temveč povezave med entitetnimi tipi. Tudi tu govorimo o tipu povezave med entitetami dveh (ali več) entitetnih tipov, kar bomo pogosto poenostavljeno imenovali kar povezava. V grafični obliki tipe povezav prikazujemo s črtami med pravokotniki, ki prikazujejo entitetne tipe. Na primer, iz modela



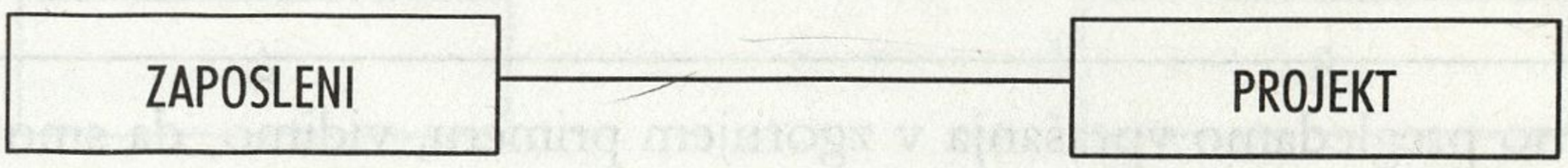
je razvidno, da sta lahko nek študent in nek predmet povezana. Seveda pa ni nujno vsak študent povezan z vsakim predmetom. S tem pa prehajamo na še en pomemben del analize, kjer se odražajo poslovna pravila, to je kardinalnost povezav.

Analiza povezav med entitetnimi tipi

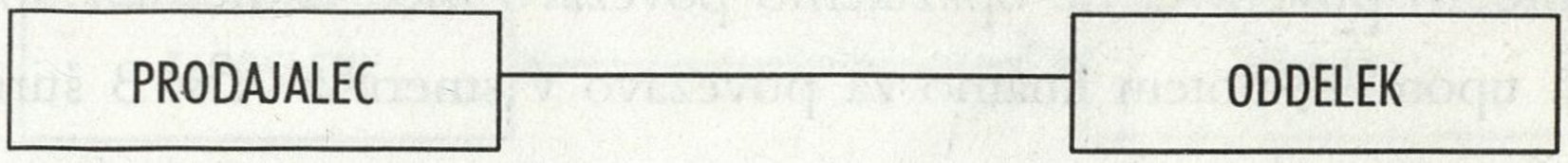
Poznavanje in razumevanje povezav med entitetnimi tipi je potrebno načrtovalcem informacijskih sistemov. Ker pa se v povezavah odražajo poslovna pravila, jih lahko analizira le s pomočjo uporabnika, ki dobro pozna problemsko področje.

Poglejmo si na dveh primerih, kaj nas zanima pri povezavah med entitetnimi tipi.

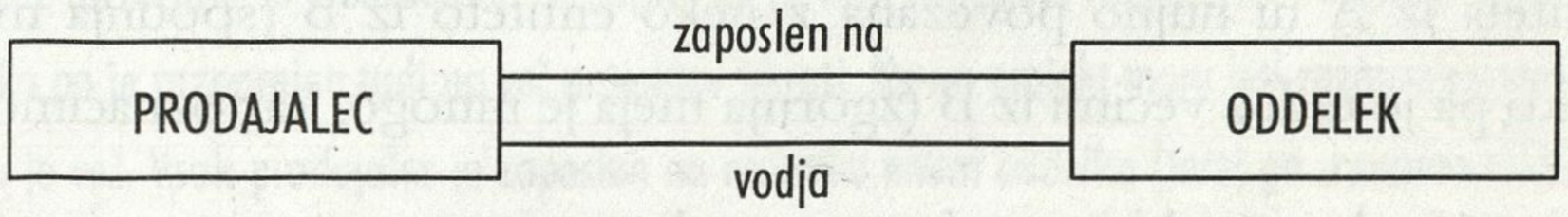
V prvem primeru imamo entitetna tipa ZAPOSLENI in PROJEKTI in med njima seveda obstaja povezava, saj ne nekem projektu delajo nekateri zaposleni v podjetju. Seveda ni nujno, da vsak zaposleni del na vsakem projektu.



V drugem primeru opazujemo blagovnico z oddelki, pri čemer so prodajalci razporejeni na določene oddelke.



Vendar lahko med prodajalci in oddelki obstaja še en tip povezave. Namreč, nekateri prodajalci imajo lahko tudi vlogo vodje oddelka, na katerem delajo. Kadar obstaja med entitetnimi tipi več tipov povezav (pa tudi sicer), lahko povezave med njimi označimo še z opisom.



Vprašanja, ki se nam ob analizi povezav zastavljajo, so, na primer:

- Ali mora vsak zaposleni v vsakem trenutku delati na vsaj enem projektu?
- Ali lahko zaposleni dela na več projektih hkrati?
- Ali mora biti na vsak projekt razporejen vsaj en zaposleni?
- Ali je lahko na projekt razporejenih več delavcev?
- Ali mora biti prodajalec v vsakem trenutku razporejen na vsaj en oddelek?
- Je lahko zaposlen na več oddelkih?
- Ali mora biti na vsakem oddelku zaposlen vsaj en prodajalec?
- Jih je lahko več?

- Ali mora imeti vsak oddelek šefa?
- Jih ima lahko več?
- Mora biti vsak prodajalec šef vsaj enega oddelka?
- Je lahko prodajalec hkrati šef več oddelkov?

Odgovori na nekatera vprašanja so očitni, za nekatera pa lahko odgovori le nekdo, ki pozna organizacijo – poslovna pravila v podjetju, ki ga obravnavamo. Tako bo, na primer, v večini podjetij veljalo, da lahko vsak zaposleni dela na več projektih, nimamo pa nikakršnega razloga, da bi to privzeli kot splošno pravilo v vseh okoljih.

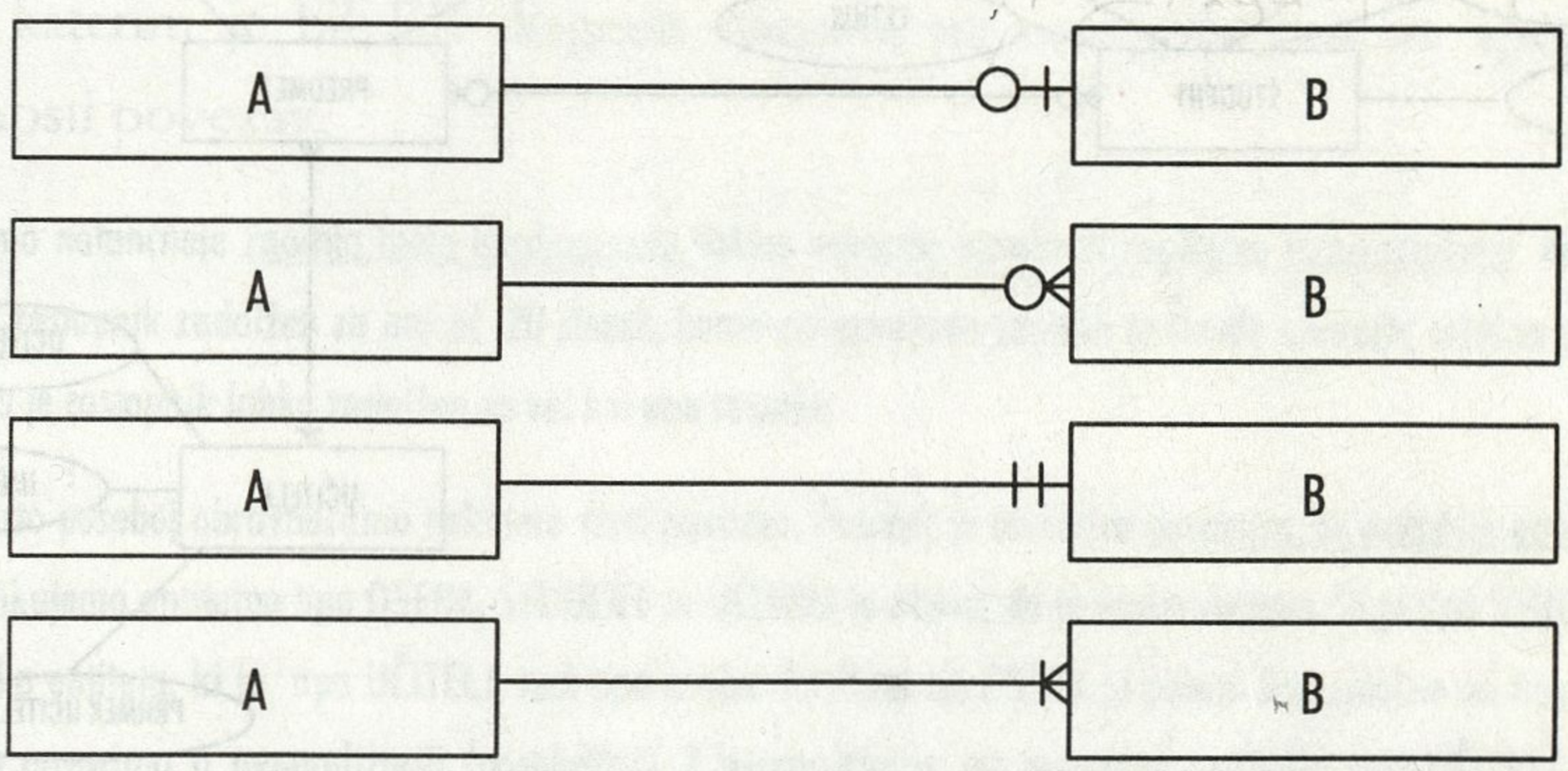
Če pozorno pregledamo vprašanja v zgornjem primeru, vidimo, da smo za vsako smer vsake povezave spraševali po spodnji in zgornji meji **številčnosti** (**kardinalnosti**) povezave. Če opazujemo povezavo med entitetnim tipom A in entitetnim tipom B, potem imamo za povezavo v smeri iz A v B štiri mogoče kombinacije:

- entiteta iz A ni nujno povezana z neko entiteto iz B (spodnja meja je 0), če pa je, je z največ eno entiteto iz B (zgornja meja je 1);
- entiteta iz A ni nujno povezana z neko entiteto iz B (spodnja meja je 0), lahko pa je tudi z večimi iz B (zgornja meja je mnogo, kar označimo z N);
- entiteta iz A mora biti v vsakem trenutku povezana z vsaj eno in največ eno entiteto iz B (spodnja meja in zgornja meja je 1);
- entiteta iz A mora biti v vsakem trenutku povezana z vsaj eno entiteto iz B (spodnja meja je 1), lahko pa je tudi z večimi (zgornja meja je N).

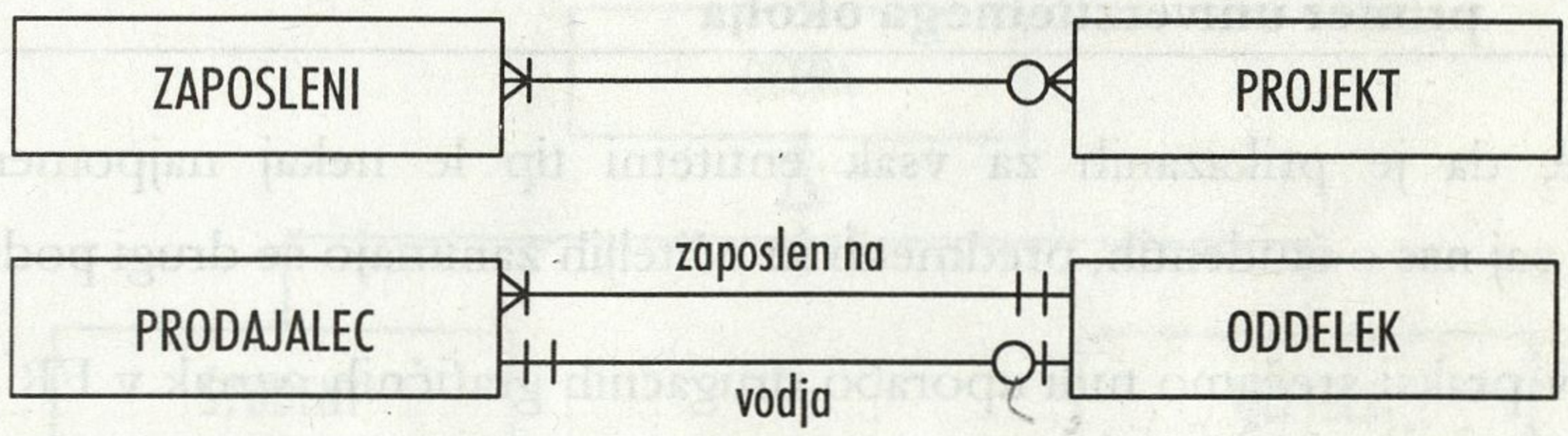
Hitro lahko ugotovimo, da imamo, če opazujemo povezavo v obe smeri, skupaj 16 možnosti. Grafično spodnje in zgornje meje povezav označimo na povezavi s simboli:

- krožec za 0,
- črtica preko povezave za 1 in
- tri razprte črtice (t.i. "grabljice" ali tudi "kurje tace") za N.

Gornje štiri kombinacije (v istem vrstnem redu) bi bile grafično potemtakem prikazane takole:



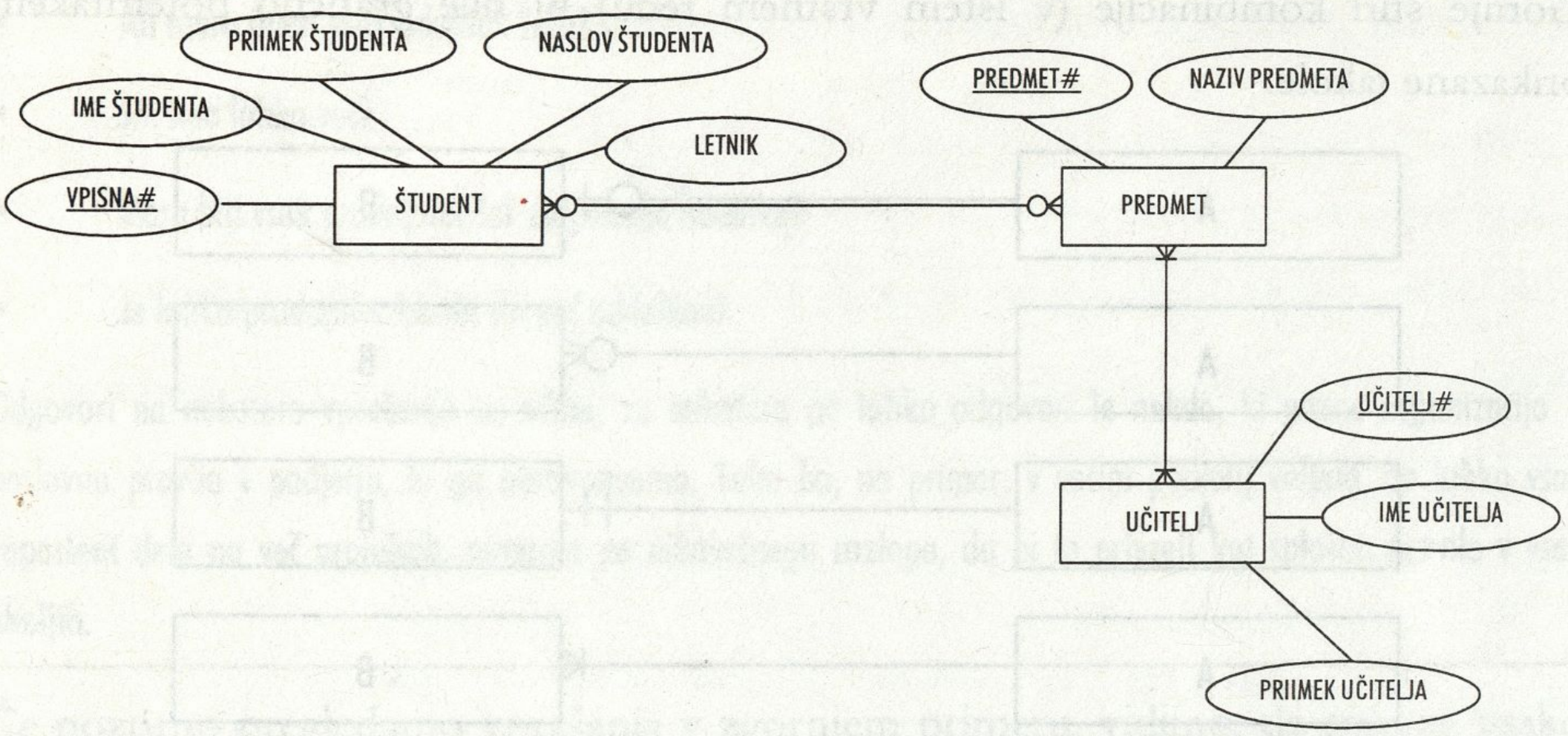
Vrnimo se na primer zaposleni-projekti in prodajalci-oddelki. Grafično bi kardinalnost prikazali takole



Grafični prikaz razložimo še z besedami. Ni nujno, da je vsak zaposleni razporejen v vsakem trenutku na vsaj en projekt, lahko pa je razporejen tudi na več projektov hkrati. Na en projekt mora biti razporejen vsaj en zaposleni, lahko pa jih je več. Vsak prodajalec je zaposlen na natanko enem oddelku (torej ga moramo takoj ob zaposlitvi razporediti). Oddelek mora imeti vsaj enega prodajalca, lahko pa jih ima več. Vsak prodajalec ni nujno vodja kakšnega oddelka in ne more biti hkrati vodja več kot enega oddelka. Vsak oddelek mora imeti natanko enega vodjo.

Naj še enkrat opozorimo, da je prikazan zgolj primer kardinalnosti; v drugem okolju, kjer bi bila poslovna pravila drugačna, bi bile posledično tudi kardinalnosti drugačne.

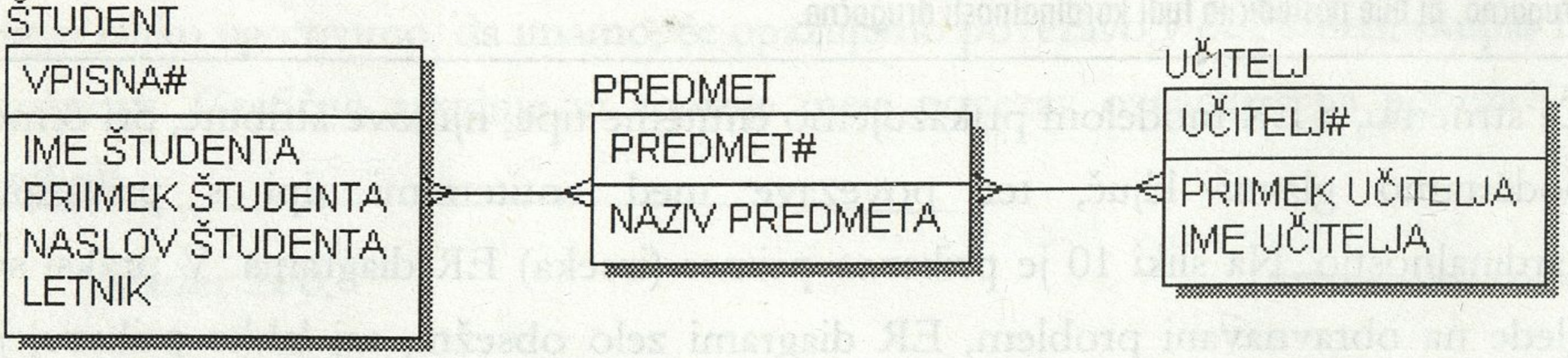
Če strnemo, z ER modelom prikazujemo entitetne tipe, njihove atribute, pri čemer podčrtamo glavni ključ, ter povezave med entitetnimi tipi s prikazano kardinalnostjo. Na sliki 10 je prikazan primer (izseka) ER diagrama. V praksi so, glede na obravnavani problem, ER diagrami zelo obsežni, saj lahko prikazujejo nekaj deset ali nekaj sto entitetnih tipov.



Slika 10: Primer izseka iz diagrama entitet-povezav za obravnavani primer univerzitetnega okolja

Očitno je, da je prikazanih za vsak entitetni tip le nekaj najpomembnejših atributov, saj nas o študentih, predmetih in učiteljih zanimajo še drugi podatki.

Neredko v praksi srečamo tudi uporabo drugačnih grafičnih oznak v ER modelih. Tako včasih povezave dodatno označimo z likom romba, kardinalnost označujemo tako, da na črte, ki simbolizirajo povezave, zapišemo oznake kar z znaki 0, 1 ali N, spodnje meje kardinalnosti označimo tako, da povezavo narišemo s polno (obvezna povezava) oziroma črtkano črto (neobvezna povezava) … Zelo pogosto atribute, namesto v elipse (mehurčke) okoli entitetnega tipa, zapišemo kar v pravokotnik, ki simbolizira entitetni tip, pri čemer ime entitetnega tipa ločimo od atributov s črto.

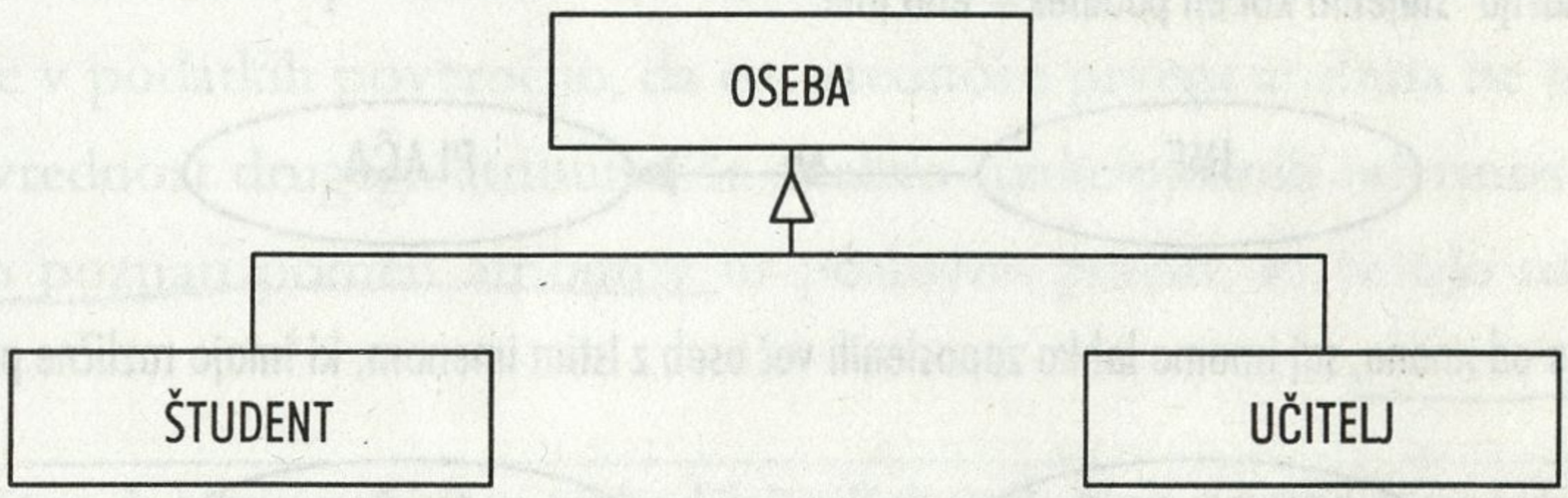


Slika 11: ER diagram s slike 10 izdelan s programskim orodjem, ki uporablja nekoliko drugačen grafični prikaz

Na sliki 7 je ime entitetnega tipa prikazano nad pravokotnikom, v prvem delu pravokotnika so navedeni ključni atributi, za črto pa še neključni. Programski paket, s katerim je bil ER diagram narisan, ne omogoča prikaza spodnjih mej kardinalnosti povezav.

Kadar poznamo natančneje zgornjo mejo kardinalnosti, lahko namesto oznake N zapišemo točno vrednost. Na primer, če je vsak prodajni zastopnik zadolžen za največ 20 strank, bomo na povezavo zapisali to število namesto splošne oznake N, ki bi označevala, da je zastopnik lahko zadolžen za več kot eno stranko.

V praksi pogosto posebej obravnavamo nekatere vrste povezav. Posebej je zanimiva povezava, ki označuje generalizacijo. Na primer, če oblikujemo entitetne tipe OSEBA, ŠTUDENT in UČITELJ je očitno, da je vsaka entiteta, ki je tipa ŠTUDENT, tudi tipa OSEBA in vsaka entiteta, ki je tipa UČITELJ, tudi tipa oseba. Entitetni tip OSEBA je potem bolj splošen od tipov ŠTUDENT in UČITELJ, zato govorimo o generalizaciji (posplošitvi). Z generalizacijo sta povezana entitetna tipa UČITELJ in OSEBA ter ŠTUDENT in OSEBA. Grafično generalizacijo prikažemo takole:

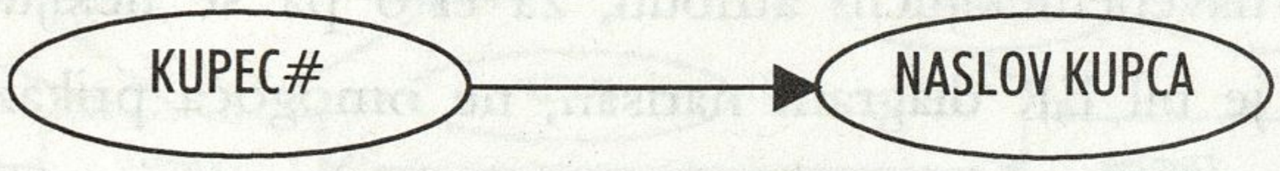


Verjetno ni potrebno posebej poudarjati, da imajo povezave med entitetnimi tipi zgornje meje kardinalnosti enake 1. O spodnjih mejah naj razmisli bralec sam. Ker je v primeru generalizacije kardinalnost določena, je na diagramu posebej ne označujemo.

Funkcionalna odvisnost

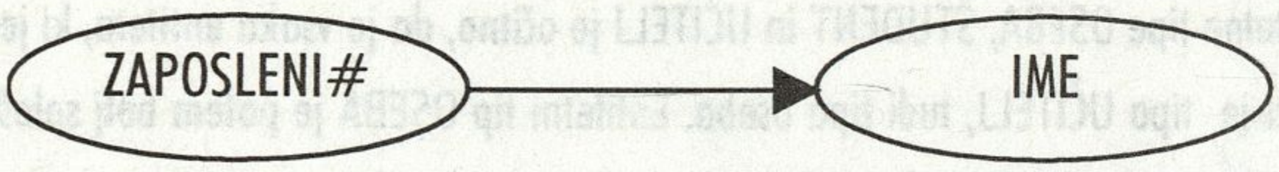
Za ustrezno organizacijo podatkov je pomembno tudi ugotavljanje **funkcionalne (funkcijske) odvisnosti** (angl. functional dependency) med atributi. *Atribut B je funkcijsko odvisen od atributa A, če v vsakem trenutku k vsaki vrednosti atributa A pripada največ ena, z njo povezana, vrednost atributa B.* Kadar ne bo dvoumno, bomo rekli kar krajše, da je atribut B odvisen od atributa A.

Na primer, povezava



ilustrira, da je naslov kupca (atribut NASLOV KUPCA) odvisen od šifre kupca (atribut KUPEC#). To velja seveda v primeru, da je v okviru našega interesa le en naslov kupca (npr. stalni naslov). Če bi nas zanimali vsi naslovi kupca (npr. tudi začasni), potem prikazana odvisnost ne bi veljala.

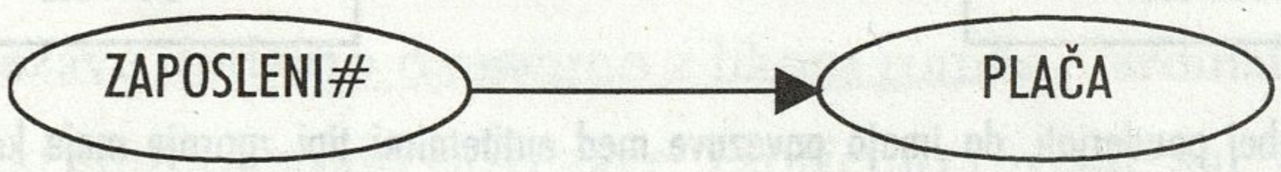
Poglejmo še nekaj primerov, ko med atributi obstaja funkcionalna odvisnost, in nekaj primerov, ko ta ne obstaja.



Ime je odvisno od šifre zaposlenega, saj eni šifri pripada le eno ime. Opozarjamo, da sestavljena imena, kot je, na primer, "Ana Marija" štejemo kot en podatek – eno ime.



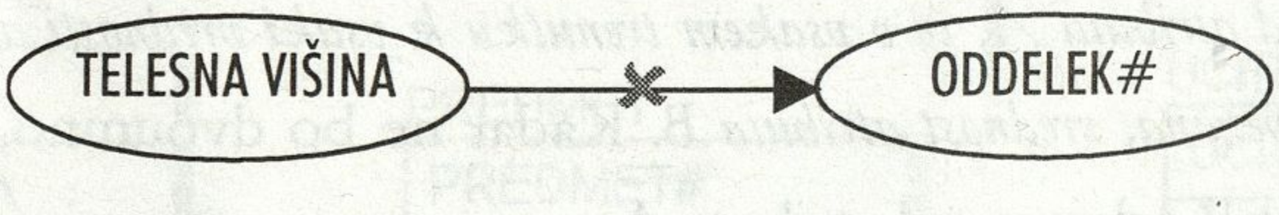
Plača ni odvisna od imena, saj imamo lahko zaposlenih več oseb z istim imenom, ki imajo različne plače.



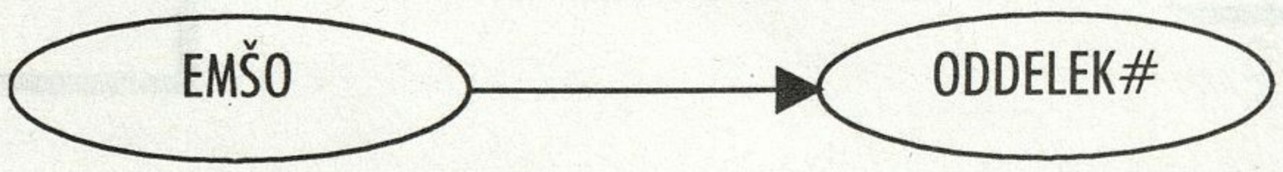
Pač pa je plača odvisna od šifre zaposlenega, saj ima vsak zaposleni (v nekem trenutku) le en pripadajoči znesek plače.



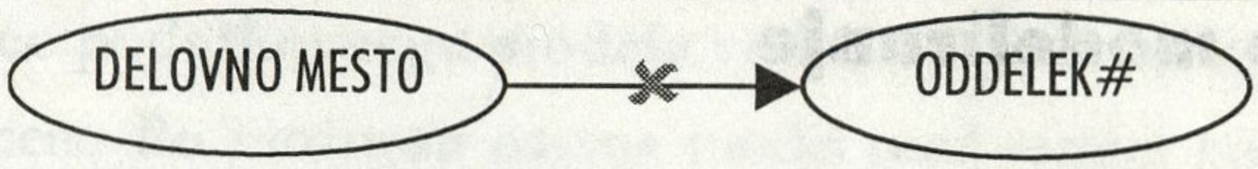
Na enem oddelku je lahko več ljudi, ki imajo isti priimek.



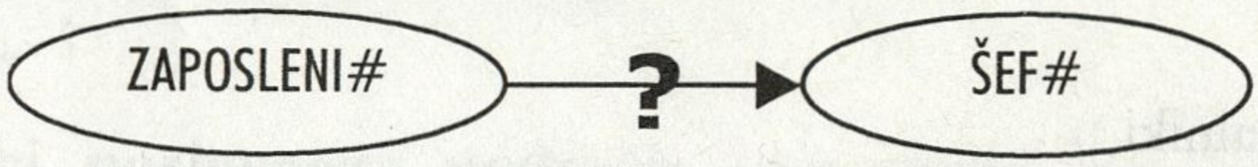
Kaj pa, če znamo meriti telesno višino zelo natančno?



Zdaj je najbrž že jasno, da je oddelek odvisen od matične številke pri izpolnjenem pogoju, da je v našem podjetju ena oseba lahko istočasno razporejena le na en oddelek (poslovno pravilo!)?



V tem primeru se moramo vprašati, kako je definiran atribut delovno mesto. Če je, na primer, vrednost atributa DELOVNO MESTO lahko "vodja oddelka", potem seveda ni odvisnosti. Če pa so delovna mesta natančneje opredeljena ("vodja oddelka za trženje"), potem odvisnost morda obstaja.



Tokrat pa bi morali za odgovor na vprašanje, ali je šifra šefa odvisna od šifre zaposlenega, vedeti, ali je s povezavo mišljen neposredni predpostavljeni ali vsi predpostavljeni.

Opozorimo na (za analizo) pomembno posledico definicije odvisnosti: Na podlagi konkretnih trenutnih podatkov ne moremo sklepati na odvisnost, saj lahko spremembe v podatkih povzročijo, da eni vrednosti prvega atributa ne pripada več samo ena vrednost drugega atributa. Za analizo funkcionalnih odvisnosti moramo torej nujno poznati pomen atributov in poslovna pravila, ki veljajo na področju obravnave.

Vzemimo za primer, da hranimo podatke o telefonskih številkah naših strank. Trenutno imamo zbrane naslednje podatke:

IME STRANKE	TELEFON
Ana	111-222
Miha	222-322
Peter	827-222

K vsaki vrednosti atributa IME STRANKE pripada trenutno natanko ena vrednost atributa TELEFON. Ali lahko sklepamo, da je atribut TELEFON odvisen od atributa IME STRANKE? Vsekakor ne, saj lahko dobimo novo stranko z imenom Ana (gre seveda za drugo Ano), ki bo imela tudi drugačno telefonsko številko.

Analiza funkcionalnih odvisnosti je, kot bomo videli v nadaljevanju, zelo pomembna za dobro načrtovanje organizacije podatkov. Kot smo že spoznali, mora pri tovrstni analizi sodelovati oseba, ki pozna pomen atributov in poslovna pravila, torej poslovno področje, ki je predmet obravnave.

Podatkovno modeliranje

Analitik lahko spoznava poslovno področje, ki ga namerava predstaviti s podatkovnim modelom, na različne načine, med katerimi so najpogostejši:

- pregled obstoječe dokumentacije in morebitnih obstoječih podatkovnih virov,
- pisni vprašalniki,
- posamični intervjuji uporabnikov,
- skupinski intervjuji,
- opazovanje uporabnikov pri delu …

Vsak izmed načinov ima svoje prednosti in slabosti, zato večinoma uporabljamo njihove kombinacije. Zagotovo pa praktične izkušnje kažejo, da analitiki najpogosteje uporabljajo pregled dokumentacije, kar je skoraj nujno, in posamične ter skupinske intervjuje.

Intervjuji

Ne glede na to, ali gre za posamične ali skupinske intervjuje, morajo biti le-ti dobro pripravljeni. Čeprav je prednost intervjujev pred vnaprej pripravljenimi vprašalniki ravno fleksibilnost – možnost sprotnega prilagajanja poteka intervjuja, pa to seveda ne pomeni, da analitik ne bo vnaprej pripravil vprašanj in smeri pogovora.

Intervjuji morajo biti dogovorjeni pravočasno, da se lahko vsi udeleženci pripravijo, dogovorjeni pa morajo biti tudi postopki in pravila poteka seje, urnik in sodelujoči.

Pri skupinskih intervjujih sodelujejo osebe z različnimi vlogami, praviloma so to:

- **vodja sestanka**, ki se dogovori za sestanek in zagotovi potrebno opremo (npr. table za risanje diagramov), zagotovi vnaprej pripravljeno dokumentacijo in vodi sestanek tako, da udeležence usmerja in zagotavlja, da se držijo pravil;

- **izdelovalec podatkovnega modela** vodi sodelujoče pri izgradnji modela in njegovi oceni. Po možnosti nastaja model med samim potekom seje, pred skupino, kar omogoča sprotno usklajevanje mnenj in ugotavljanje podrobnosti;
- **podatkovni analitik** beleži ugotovitve, zapisuje definicije entitet in atributov …;
- **poznavalci poslovnega področja** (uporabniki) so tisti udeleženci, ki predstavljajo poslovno področje in s tem zagotavljajo informacije, ki so potrebne za izgradnjo modela. Pomembno je, da so to res osebe s "poslovnega področja" in ne morda informatiki iz organizacije;
- **predstavnik vodstva – manager**, ki je zadolžen, da se sprejemajo odločitve, ki so potrebne, da lahko proces izdelave modela poteka tekoče. Če se, na primer, uporabniki ne strinjajo o poslovnih pravilih ali definiciji podatkovnih elementov, je njegova naloga, da se o tem odloči. Vendarle, do njegove intervencije naj pride le, ko je to res nujno potrebno.

Prednost skupinskih intervjujev pred posamičnimi je ta, da je mogoče nasprotujoča si mnenja sproti usklajevati, hkrati pa se izkaže, da prisotnost ostalih, predvsem managerja, pogosto zavira sproščeno izražanje mnenj in drugačnih pogledov.

Na takih skupinskih srečanjih se pogosto prvič v podjetju srečajo ljudje z različnih oddelkov, prvič izmenjajo poglede o poslovnih potrebah in pravilih. Hkrati pride do razvoja konsistentnih in natančnih definicij izrazov, ki jih uporabljajo, izmenjave poslovnega znanja, uporabniki prvič spoznajo, kako so njihove aktivnosti vključene v celoten poslovni proces …

Prednosti uporabe ER diagramov

Diagrami entitet-povezav niso po naključju najbolj razširjena tehnika za izdelavo podatkovnih modelov:

- Podatkovni model lahko nedvoumno izraža zahteve uporabnikov, kar je pomembno za razvijalca sistema, za katerega dvoumno izražene zahteve predstavljajo problem.
- Model je "poslovno" usmerjen. To pomeni, da vsebino in strukturo sistema nadzoruje uporabnik in ne informatik. Poudarek je torej na zahtevah in ne na rešitvah ali omejitvah, ki jih postavlja tehnologija.
- Pri izdelavi modela se uporablja poslovna terminologija in ne tehnična.
- Model je okvir, v katerem se lahko osredotočimo na stvari, ki so pomembne za poslovanje.

Kaj je še treba vedeti o izdelavi podatkovnih modelov?

Dokončnega odgovora, kako bo potekala analiza v vsakem posameznem primeru, ni mogoče dati, saj je potek odvisen od mnogih posebnosti konkretnega problemskega okolja. Velja pa uporabiti nekatere nasvete, ki so posledica praktičnih (predvsem seveda slabih) izkušenj.

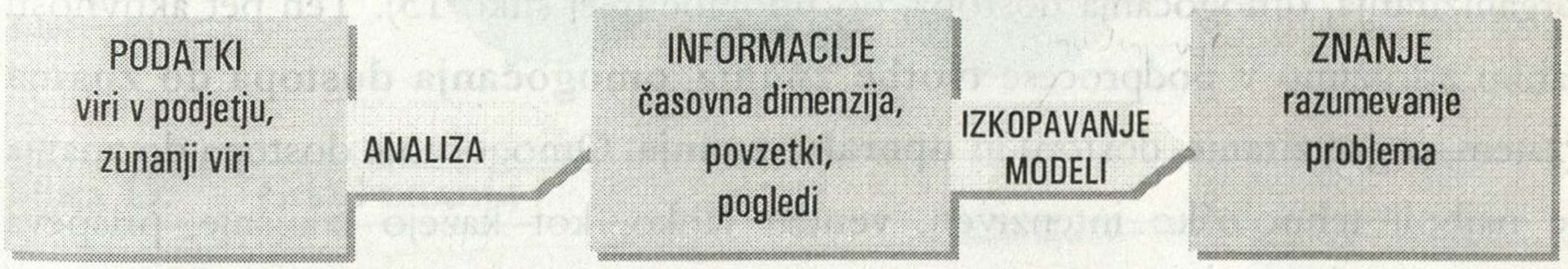
- Izdelava podatkovnega modela ni preprosta, je časovno zahtevna, zahteva sodelovanje številnih sodelavcev, vendar je bistvenega pomena za razvoj kakovostnega podatkovnega vira. Zato naj nam ne bo žal časa in trud, ki ga bomo v to delo vložili.
- Brez ustrezne podpore vodstva organizacije je trud, ki ga vložimo v analizo, zaman, saj rezultati ne morejo biti dobri in v nadaljevanju dobro uporabljeni. Vodstvo mora podpirati sodelovanje uporabnikov v skupinskih intervjujih med delovnim časom, odobriti posredovanje (včasih zaupnih) dokumentov analitikom …
- Tako vodstvo kot informatiki morajo vzpodbujati interes uporabnikov za sodelovanje. Mnogokrat se srečujemo z odpori uporabnikov do novosti in s tem povezanim pomanjkanjem interesa za sodelovanje, zato jim moramo

pravočasno predstaviti prednosti pri njihovem delu, ki jih bo prinesel nov ali na novo postavljen podatkovni vir.

- V večini primerov razvijamo podatkovni vir v okviru širšega razvoja informacijskega sistema (programske opreme ...). V takem primeru je podatkovno modeliranje le del analize, ki je seveda z ostalimi deli tesno povezano in prepleteno. Na primer, podatkovne potrebe v primeru informatizacije poslovnih procesov izhajajo iz modelov poslovnih procesov.
- Izogibati se moramo pretiranemu ukvarjanju s podrobnostmi tam, kjer to ni potrebno. Tudi tu velja, da moramo upoštevati Paretovo pravilo 80/20; to je, da z 20 % vloženega dela opravimo z 80 % problematike, za ostalih 20 % pa moramo vložiti kar 80 % dela.

2.4 Upravljanje znanja

Znanje je razumevanje, zavedanje ali poznavanje, pridobljeno z izobraževanjem ali izkušnjami (Turban, McLean, Wetherbe, 1999). Omogoča prepoznavanje znanih vzorcev v novih odločitvenih situacijah in s tem omogoča bolj učinkovito in hitrejše reševanje problemov (Sobočan, 1998). Znanje pridobivamo vse življenje, ga izmenjujemo z drugimi, ga s tem povečujemo in ustvarjamo kolektivno znanje. Znanje pridobivamo iz informacij, ki naprej izvirajo iz podatkov.



Slika 12: Porterjeva veriga vrednosti – od podatkov do znanja

Čeprav bi se v večini organizacij pridružili mnenju o pomembnosti znanja, pa marsikje ne prepoznavajo pomena, ki ga ima učinkovito upravljanje znanja. Znanje je namreč marsikje nedokumentirano (v glavah zaposlenih), se ne razširja in ni dostopno ali pa je neorganizirano. Zaradi tega prihaja do številnih težav, na primer:

- Težave nastopajo pri prenosu znanja pri odhodu iz službe ali odhodu na drugo delovno mesto. Zgodi se tudi, da na ta način izgubimo znanje, ki je kritično za izvajanje poslovnih procesov.
- Ne vemo, kakšne načine delovanja je uporabljala skupina, ki je bila uspešna. S tem ponavljamo že narejene napake.
- Preveč je ponovnega učenja istih stvari in reševanja enakih problemov.
- Ni učenja na napakah.
- Hitrost usposabljanja pri prehodu na novo delovno mesto je premajhna.

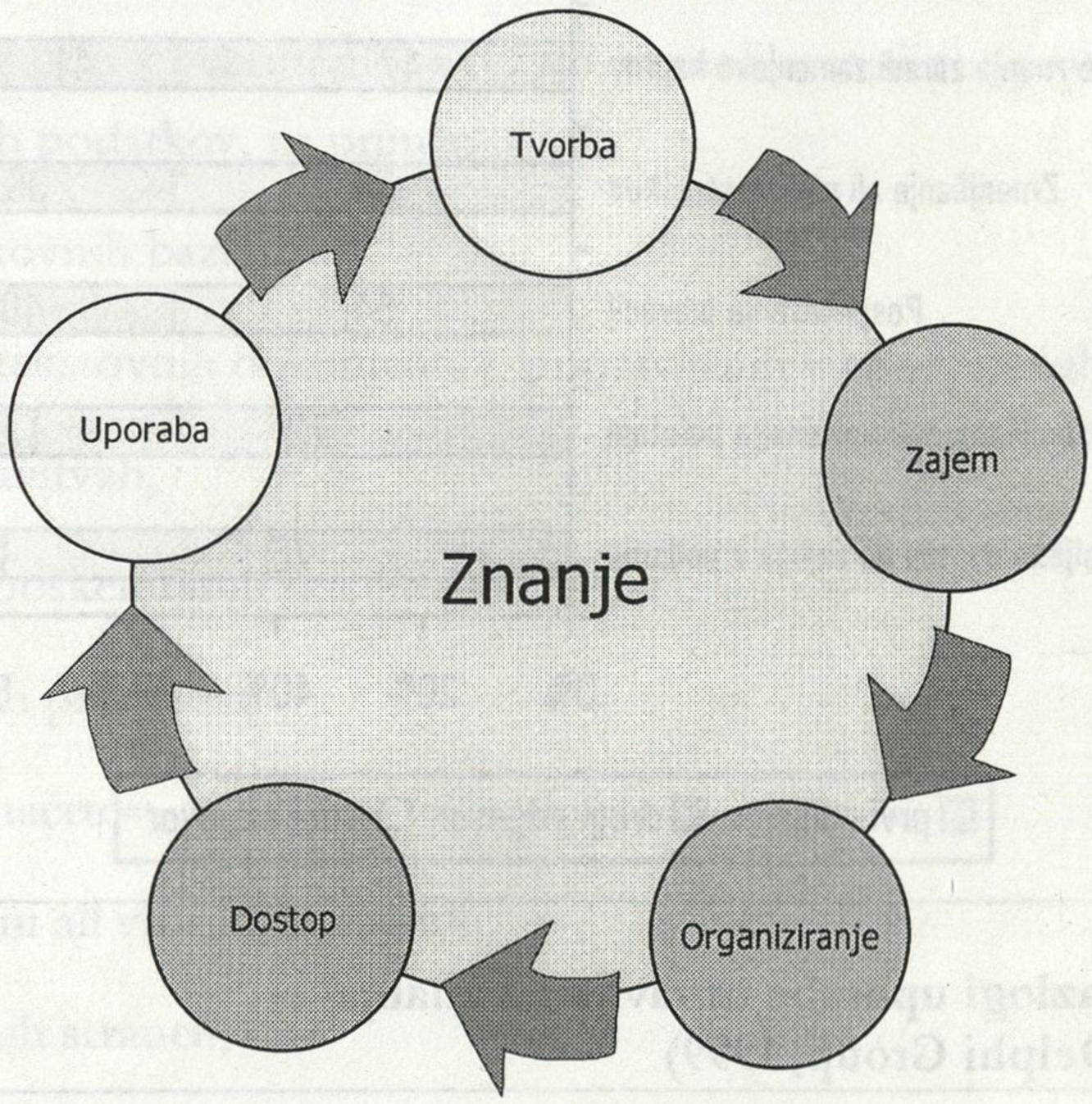
Upravljanje znanja (angl. knowledge management) je multidisciplinarni pristop, ki združuje poslovno strategijo, kulturne vrednote in delovne postopke. Obsega postopke načrtovanja, izvajanja, spremljanja in izboljšave strategij, poslovnih procesov, organizacijskih struktur in tehnologij, ki omogočajo učinkovito uporabo znanja v organizaciji. Čeprav se upravljanje znanja pogosto omenja v povezavi s sodobno informacijsko tehnologijo, je potrebno poudariti, da ne gre zgolj za tehnologijo ali implementacijo tehnologije. Res pa je, da je lahko bolj učinkovito, če je podprto z ustrezno tehnologijo. Osnovni namen upravljanja znanja je izraba intelektualnega kapitala, ki je poglavitni dejavnik razlikovanja med knjižno vrednostjo podjetja in vrednostjo na borznih trgih.

Upravljanje znanja mora pokrivati celoten cikel znanja: od tvorbe, preko zajema, organiziranja, omogočanja dostopa, do uporabe (glej sliko 13). Teh pet aktivnosti lahko združimo v podprocese **tvorbe znanja**, **omogočanja dostopa do znanja** (zajem, organiziranje, dostop) in **uporabe znanja**. Omogočanje dostopa do znanja je najbolj tehnološko intenziven, vendar lahko, kot kažejo izkušnje, prispeva bistvene koristi organizaciji.

Pri tem moramo odgovoriti na vprašanja (Sobočan, 1998):

- Kakšna znanja nastajajo v organizaciji?
- Kakšno znanje obstaja v organizaciji in kdo ga ima?
- Kdo, kdaj in kakšno znanje potrebuje?
- Kako je mogoče znanje najučinkoviteje uporabiti?

Upravljanje znanja bi moralo biti sestavni del aktivnosti vsakega zaposlenega, vendar se pri tem srečujemo z monopoli znanja in njegovo izolacijo.

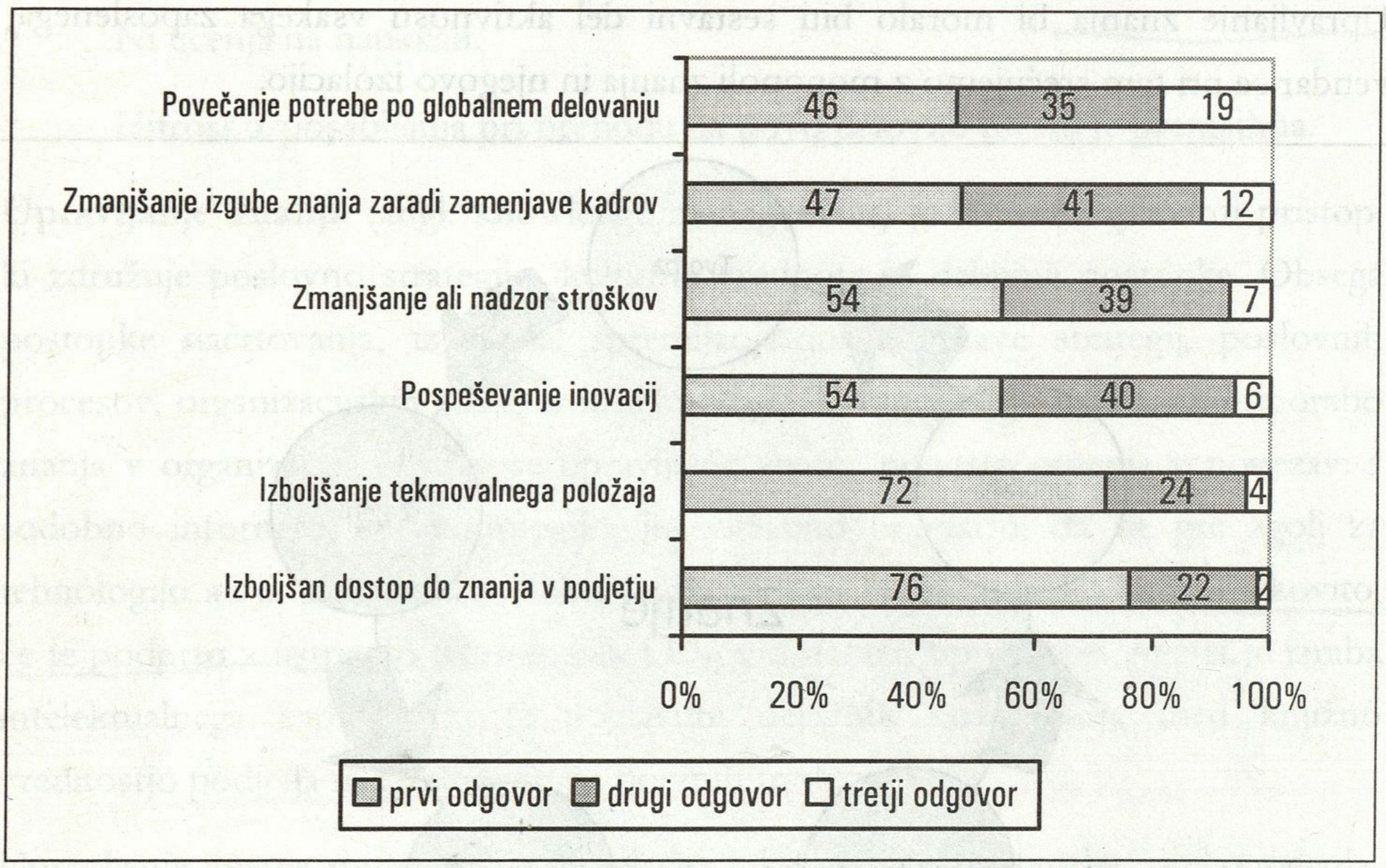


Slika 13: Cikel znanja

Na sliki 14 je prikazano, kaj podjetja (Severna Amerika in Evropa) najpogosteje pričakujejo od upravljanja znanja. Omogočanje dostopa do znanja znotraj podjetja je na prvem ali drugem mestu po pomembnosti za kar 98% podjetij.

Informacijska tehnologija za upravljanje znanja

Omenili smo že, da je predvsem omogočanje dostopa do znanja tehnološko intenziven podproces upravljanja znanja, vendar je možnosti uporabe informacijske tehnologije precej tudi pri ostalih dveh aktivnostih. Na primer, pridobivanje znanja lahko podpremo z uporabo tehnologij za rudarjenje v podatkih (glej poglavje Informacijska podpora managementu).



Slika 14: Razlogi uporabe upravljanja znanja (Delphi Group, 1999)

Podobno nam lahko pri pridobivanju znanja pomagajo **inteligentni agenti** (angl. intelligent agents). Gre za programe, ki delujejo (do določene stopnje) samostojno in to namesto ter v imenu uporabnika. Tako se lahko, na primer, sprehajajo po Svetovnem spletu in iščejo spletne stani, ki utegnejo zanimati uporabnika[16], denimo dokumente, ki opisujejo podobne probleme, kot jih trenutno rešuje uporabnik. Hkrati inteligentni agenti predstavljajo tudi uporabo znanja. V njih je

16 Zanimiva uporaba inteligentnih agentov je tudi iskanje najbolj ugodne ponudbe na spletu, ko kupujemo nek izdelek.

lahko zakodirano znanje strokovnjaka, ki ga uporablja pri rutinskem in časovno zahtevnem delu. Če je to znanje zakodirano v agentu, lahko delo opravljajo tudi drugi zaposleni, ki imajo računalnik. Le-ti vnesejo potrebne podatke, prepustijo agentu, da opravi delo, za katerega je potrebno znanje strokovnjaka, na koncu pa interpretirajo rezultate.

Pomemben del upravljanja znanja, natančneje podprocesa omogočanja dostopa do znanja, je upravljanje virov znanja. Pri tem mora biti poudarek predvsem na zajemu znanja, ki bi bilo sicer nedokumentirano, to je vpogled in izkušnje posameznih zaposlenih.

Znanje je lahko dokumentirano na zelo raznolike načine, praviloma v obliki nestrukturiranih podatkov, na primer:

- v podatkovnih bazah,
- v obliki tekstovnih dokumentov, pripravljenih v urejevalnikih besedil,
- v predstavitvah,
- v obliki poskeniranih dokumentov (slike),
- v modelih procesov,
- v obliki načrtov, grafikonov in podobno,
- kot zvočni ali video dokumenti,
- na spletnih straneh,
- v okviru diskusijskih skupin,
- v e-pošti …

Posledica je tudi veliko raznolike informacijske tehnologije, ki se uporablja za upravljanje virov znanja, na primer, baze znanja (angl. knowledge bases) v okviru ekspertnih sistemov, sistemi za upravljanje dokumentov (angl. document management systems), sistemi za upravljanje podatkovnih baz (angl. database management systems), spletni strežniki (angl. web servers), intraneti (angl. intranets) in drugo.

Za dostop do znanja se vse pogosteje uporabljajo poslovni portali (glej poglavje Informacijska podpora managementu), ki omogočajo skrivanje pestrosti tehnologij, potrebnih za dostop do znanja v raznolikih oblikah.

Večina znanja je vendarle dokumentirana v obliki različnih dokumentov: preglednice, dokumenti iz programov za urejanje besedil, načrti, modeli, večpredstavni dokumenti (zvok, video ...), kompleksni (sestavljeni) dokumenti in podobno. Ugodno je, če so ti dokumenti v digitalni obliki, torej taki, ki omogoča uporabo računalnika pri delu z njimi. Dokumenti v papirni obliki imajo naslednje slabosti:

- shranjevanje je drago,
- dostop je počasen zaradi zamudnega iskanja,
- ravnanje s podatki je utrudljivo in počasno,
- ponovna vzpostavitev uničenih dokumentov je praktično nemogoča …

Vendar tudi sama digitalizacija dokumentov še ne pomeni, da se pri delu ne bomo srečevali s težavami:

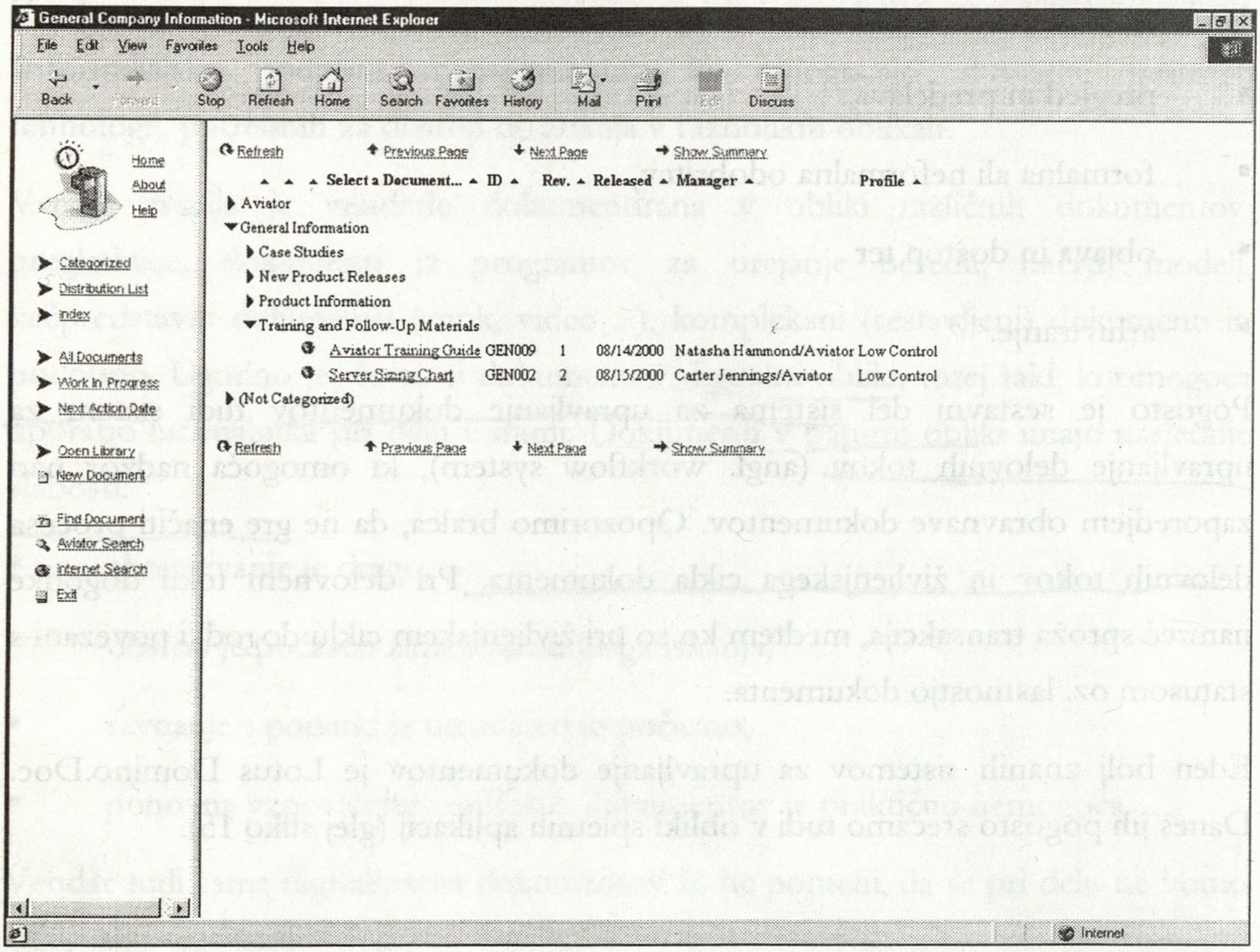
- Ali imamo vsi zadnjo inačico dokumenta?
- Kako pogosto naj se dokument spreminja?
- Kako so dokumenti zavarovani?
- Ali dokumente lahko pravočasno dostavimo vsem, ki jih potrebujejo?
- V kakšnem zaporedju zaposleni obravnavajo dokument?

Pri reševanju tovrstnih problemov nam pomagajo **sistemi za upravljanje dokumentov**, katerih namen je avtomatiziran nadzor digitalnih dokumentov. V ta namen omogočajo shranjevanje dokumentov v **dokumentni bazi**, njihovo urejanje, učinkovito iskanje, nadzor dostopa, nadzor vsebin, nadzor nad inačicami (verzijami), kategorizacijo, indeksiranje … Sestavni del upravljanja dokumentov mora biti tudi upravljanje življenjskega cikla dokumentov. Vsak dokument gre namreč praviloma skozi naslednje faze življenjskega cikla:

- priprava,
- pregled in predelava,
- formalna ali neformalna odobritev,
- objava in dostop ter
- arhiviranje.

Pogosto je sestavni del sistema za upravljanje dokumentov tudi sistem za upravljanje delovnih tokov (angl. workflow system), ki omogoča nadzor nad zaporedjem obravnave dokumentov. Opozorimo bralca, da ne gre enačiti procesa delovnih tokov in življenjskega cikla dokumenta. Pri delovnem toku dogodke namreč sproža transakcija, medtem ko so pri življenjskem ciklu dogodki povezani s statusom oz. lastnostjo dokumenta.

Eden bolj znanih sistemov za upravljanje dokumentov je Lotus Domino.Doc. Danes jih pogosto srečamo tudi v obliki spletnih aplikacij (glej sliko 15).



Slika 15: Primer sistema za upravljanje dokumèntov Aviator

Vprašanja za preverjanje razumevanja

- Katere podatke zbira fakulteta o študentih? Spomnite se tudi na vpisne liste, ki jih izpolnjujete vsako leto ob vpisu. Kje in kako vse jih uporablja?
- Za seminarsko nalogo potrebujete podatke o slovenskih podjetjih. Kako boste ugotovili, ali so potrebni podatki že zbrani in dostopni?
- Primerjajte načrtovanje razvoja novega izdelka in podatkovnega vira.
- Kako bi aktivnosti upravljanja podatkov povezali s fazami življenjskega cikla?
- Po katerih kriterijih bi se odločali, če bi pri zajemu podatkov lahko isti podatek pridobili iz več virov? Kaj je po vašem mnenju najpomembnejši kriterij? Bi izbrali primarni ali sekundarni vir, če bi imeli možnosti izbire?
- Kaj so metapodatki? Razložite na primeru.
- Kakšna je razlika med poslovnim in tehnološkim vidikom varovanja podatkov? Kakšne so po vašem mnenju naloge varnostnega inženirja v podjetjih?
- Ali sami izdelujete rezervne kopije podatkov na svojem računalniku? Kako pogosto in zakaj?
- Ali obdržimo podatke (praviloma) dlje v operativnih ali analitičnih podatkovnih virih?
- Zakaj uporabljamo diagram entitet-povezav? Narišite ER diagram za proces opravljanja izpitov na fakulteti.
- Primerjajte probleme upravljanja podatkov in znanja. Ugotovite skupne točke.
- Na kakšne načine je lahko dokumentirano znanje?
- Kakšen je odnos med upravljanjem znanja in upravljanjem dokumentov?

Ključni pojmi

aktivnosti
analiza
arhiviranje
atribut
definiranje podatkov
dokumentna baza
dostop do vira
entiteta
funkcionalna odvisnost
izobraževanje
kakovost podatkov
ključ
model entitet-povezav
načrtovanje
nadgradnja
nadzor celovitosti
nadzor kakovosti
obdelava podatkov
obnavljanje
obračun uporabe
opis podatkov
organiziranje podatkov
podatkovno modeliranje
povezava
pridobivanje podatkov
upravljanje dokumentov
upravljanje podatkovnih virov
upravljanje znanja
varovanje
vir
vzdrževanje
zaščita
zaznavanje
zbiranje podatkov
znanje
življenjski cikel

3. POGLAVJE

Podatkovne baze

Cilji poglavja

V tem poglavju boste spoznali:

- Kaj je podatkovna baza in kakšne so njene značilnosti?
- Kaj pomeni neodvisnost podatkov od načina uporabe?
- Kako uporabljamo podatkovno bazo?
- Kakšne so naloge uporabnika v zvezi z vzdrževanjem podatkovnih baz?
- Kateri so možni načini organiziranja podatkov v podatkovnih bazah?
- Zakaj se v poslovnih informacijskih sistemih največ uporabljata relacijski in večdimenzionalni podatkovni model?
- Kako se izognemo težavam (anomalijam), ki so lahko posledica slabo načrtovanih podatkovnih baz?
- Katera so programska orodja, ki nam lahko pomagajo pri načrtovanju podatkovnih baz?
- Kaj je jezik SQL ter zakaj in kako ga uporabljamo?

3.1 Podatkovne baze in njihove značilnosti

Da bi nedvoumno vedeli, o čem se pogovarjamo, najprej definirajmo pojem računalniško podprte podatkovne baze.

Podatkovna baza je

1. *urejena zbirka medsebojno povezanih podatkov*,
2. ki je *shranjena na računalniškem nosilcu podatkov*.

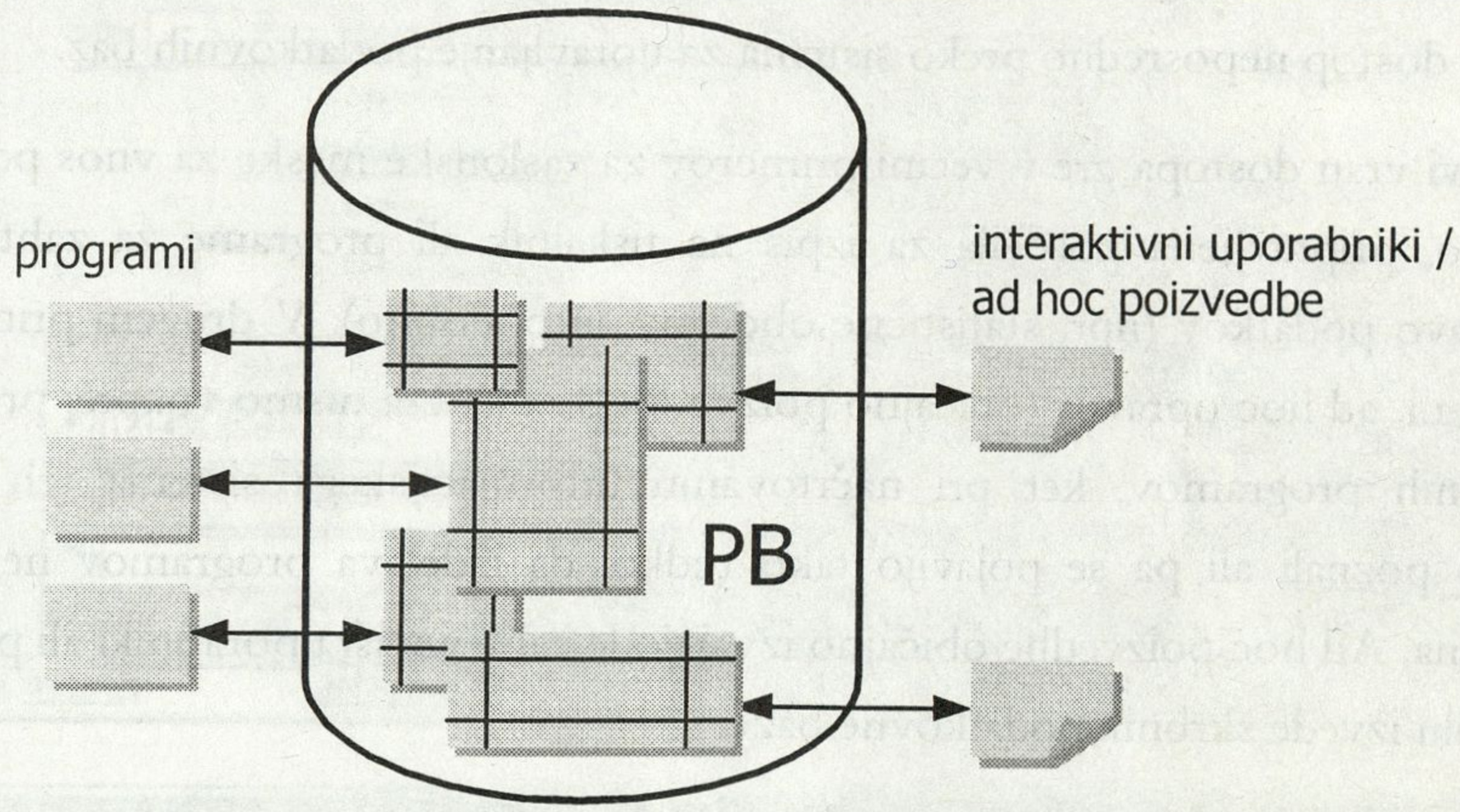
Iz definicije izhaja, da podatkovna baza ne more biti množica nekih naključno zbranih podatkov "z vseh vetrov". Prav tako je očitno, da je pridevnik "računalniško podprta", ki označuje podatkovno bazo, povsem odveč, saj to sledi že iz definicije, zato ga v prihodnje ne bomo uporabljali.

Značilnosti podatkovne baze

Podatkovna baza je načrtovana in zgrajena z nekim namenom in skladno s tem namenom odraža nek vidik realnega sveta. Z drugimi besedami, s podatki, ki so shranjeni v podatkovni bazi je opisan del realnega sveta. Seveda *se morajo* potemtakem *spremembe v realnem svetu odražati v podatkovni bazi*. Na primer, trženjska podatkovna baza bo med drugim vsebovala podatke o kupcih. Če kupec spremeni naslov (sprememba v realnem svetu), se mora spremeniti podatek o naslovu kupca tudi v podatkovni bazi.

Podatkovna baza mora biti (bi morala biti) **integrirana**. To pomeni, da vsebuje podatke za mnoge uporabnike, pri čemer vsakega od njih praviloma zanima le manjši del celote. Ker pa se potrebe uporabnikov med seboj tudi prekrivajo (uporabljajo iste podatke), to pomeni, da z integrirano podatkovno bazo zmanjšamo podvajanje podatkov na najmanjšo možno mero. Podatki niso raztreseni vsepovsod in nad njimi imamo sistematičen pregled. Na primer, podatke o dobaviteljih imamo shranjene samo enkrat, uporablja pa jih več oddelkov (služba nabave, računovodstvo ...). Zato moramo, če dobavitelj spremeni telefonsko

številko, ta podatek popraviti le enkrat, na enem mestu. Sprememba bo takoj vidna vsem, ki podatek o tem dobavitelju potrebujejo.



Slika 16: Prikaz integrirane podatkovne baze

Pri razvoju podatkovne baze ponavadi ne moremo vnaprej predvideti vseh načinov njene uporabe. Nasprotno, značilnost podatkovne baze je prav ta, da je **neodvisna od programov**, ki jo bodo uporabljali. Tak pristop omogoča večjo fleksibilnost pri razvoju informacijskega sistema, saj lahko po potrebi dodajamo nove programe, ali pa spreminjamo podatkovno bazo, ne da bi to vplivalo na že delujoče programe, ki niso neposredno povezani s to spremembo. Neodvisnost podatkovne baze od programov torej pomeni, da je informacijski sistem lažje prilagajati spremembam.

V podjetju smo uvedli možnost uporabe elektronske pošte, zato želimo za vsakega zaposlenega, ki to možnost uporablja, hraniti tudi njegov e-poštni naslov. Pred uporabo podatkovnih baz je to pomenilo, da moramo spremeniti datoteko s podatki o zaposlenih in vse programe, ki do te datoteke dostopajo, ne glede na to, ali ti programi kakorkoli uporabljajo podatek o e-poštnem naslovu. Če pa uporabljamo podatkovno bazo, dodamo v tabeli s podatki o zaposlenih še en stolpec in to ne bo imelo nikakršnega vpliva na že delujoče programe.

Uporaba in druge aktivnosti

Ker je podatkovna baza integrirana in organizirana neodvisno od uporabe, je ponavadi njena uporaba zelo raznolika.

Glede na način uporabnikovega dostopa do podatkovne baze lahko razlikujemo:

- dostop z vnaprej pripravljenimi programi ter
- dostop neposredno preko sistema za upravljanje podatkovnih baz.

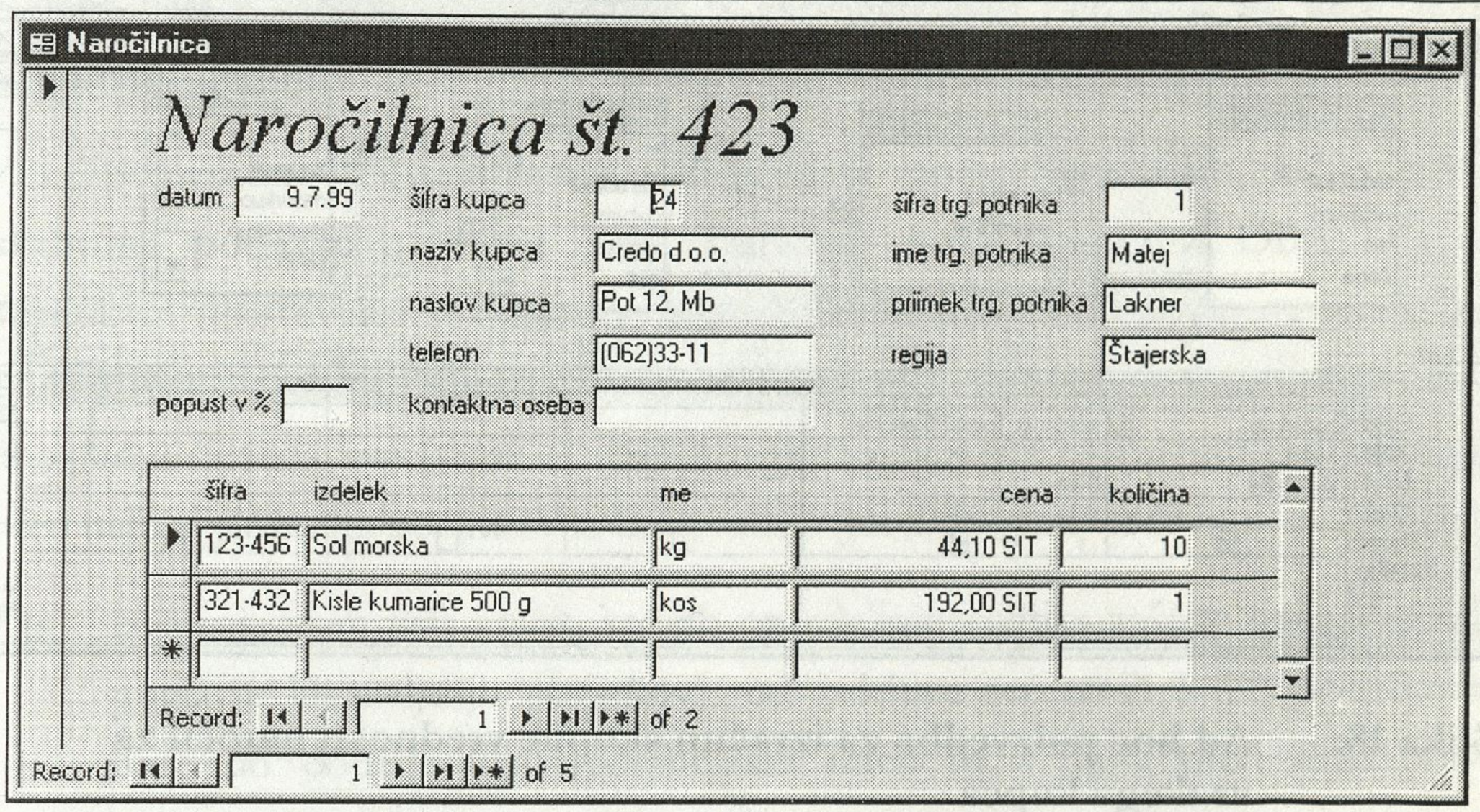
Pri prvi vrsti dostopa gre v večini primerov za zaslonske maske za vnos podatkov v bazo, pripravljena poročila za izpis na tiskalnik ali programe za zahtevnejšo obdelavo podatkov (npr. statistične obdelave in podobno). V drugem primeru pa gre za t.i. ad hoc opravila (običajno poizvedbe), za katera nismo vnaprej predvideli posebnih programov, ker pri načrtovanju informacijskega sistema teh potreb nismo poznali ali pa se pojavijo tako redko, da izdelava programov ne bi bila smotrna. Ad hoc poizvedbe običajno izvajajo le zahtevnejši uporabniki ali pa jih po naročilu izvede skrbnik podatkovne baze.

Ker se z vidika podatkovne baze načina uporabe med seboj ne razlikujeta, bomo v nadaljevanju uporabljali izraz uporabnik, ne glede na to, ali gre za program, ki dostopa do podatkovne baze, ali gre za uporabnika, ki neposredno uporablja sistem za upravljanje podatkovne baze za dostop do podatkov.

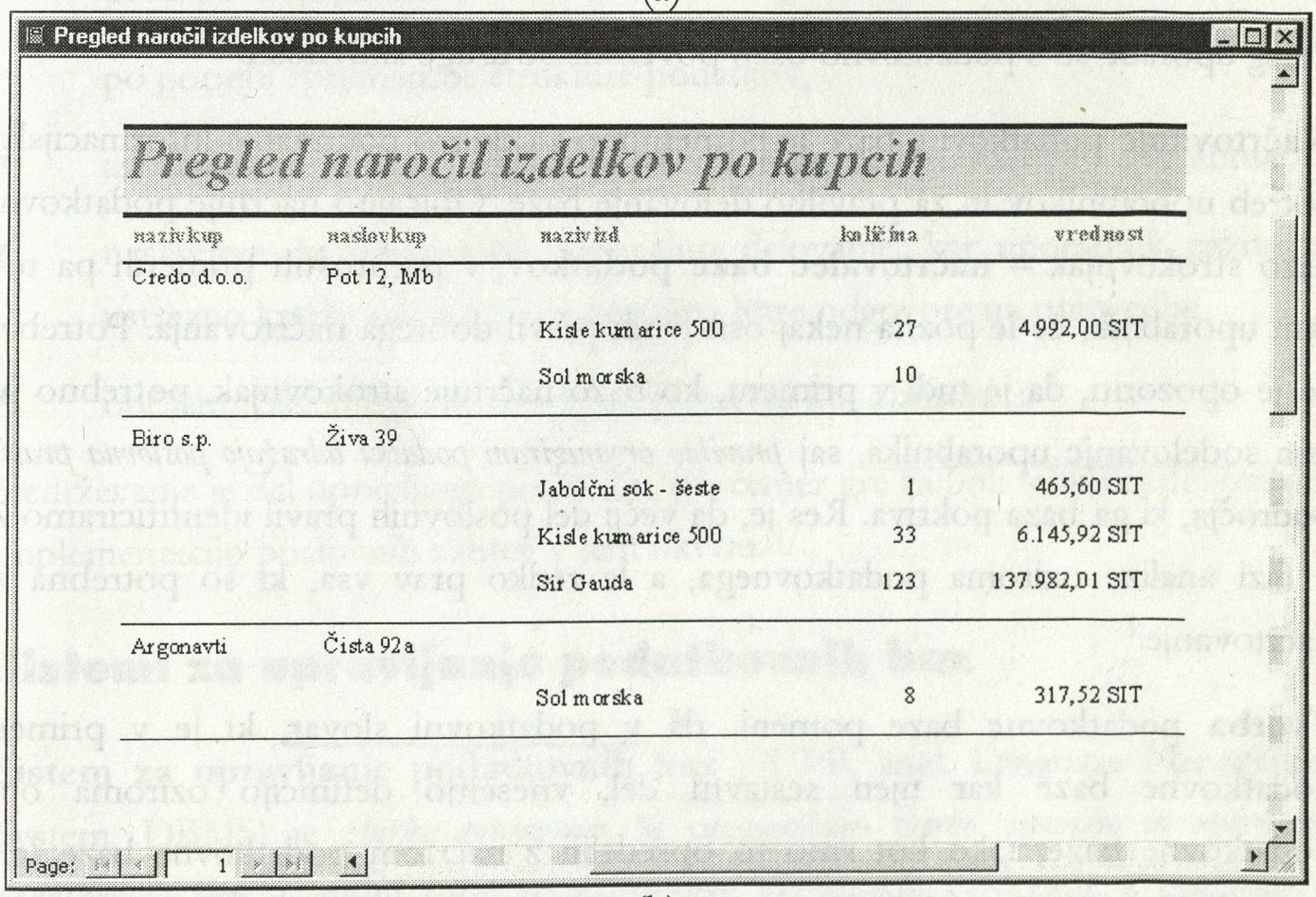
Glede na vrsto opravila pa lahko uporabo podatkovne baze razdelimo na:

- vnos novih podatkov,
- spreminjanje (ažuriranje) obstoječih podatkov,
- brisanje podatkov ter
- poizvedovanje.

Na operativnem nivoju so običajna opravila vseh vrst, pri čemer navadno dostopamo do podatkov o eni entiteti oziroma poslovnem dogodku. Na višjih nivojih upravljanja pa je najbolj običajno opravilo poizvedovanje, pri čemer nas le redko zanima posamezna entiteta, pač pa večinoma uporabljamo sumarne podatke o več entitetah oziroma poslovnih dogodkih, na primer, skupna vrednost prodaje nekega izdelka v preteklem tednu in primerjava z vrednostjo prodaje v predpreteklem tednu.



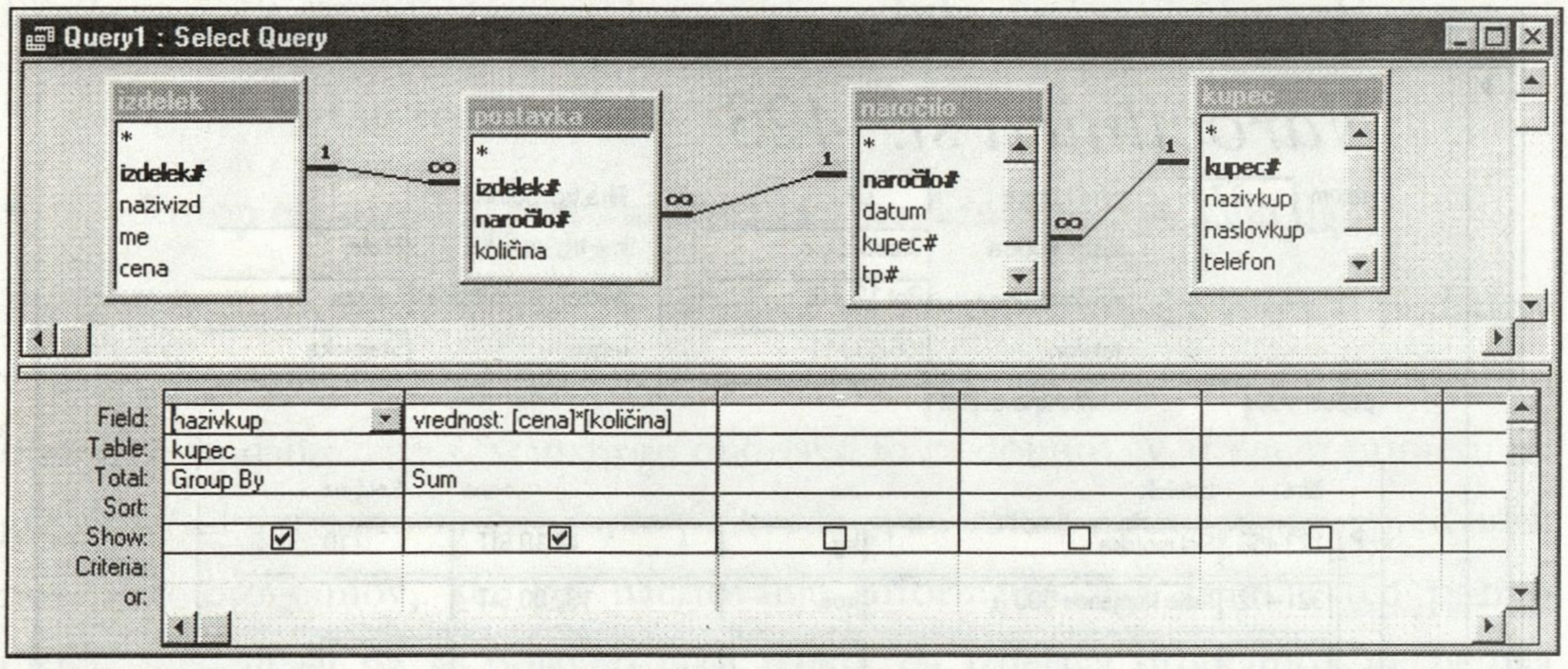
(a)



(b)

Slika 17: Zaslonska maska za vnos podatkov o naročilih (a) in poročilo – pregled podatkov o naročilih posameznih izdelkov po kupcih (b)[17]

17 V obeh primerih so prikazani podatki iz iste baze. Gre torej za dva pogleda na isto podatkovno bazo.



Slika 18: Ad hoc poizvedba za izračun skupne vrednosti naročil za vsakega kupca

Poleg uporabe so s podatkovno bazo povezane še druge aktivnosti:

Načrtovanje podatkovne baze je pomembno za dobro pokrivanje informacijskih potreb uporabnikov in za pravilno delovanje baze. Običajno načrtuje podatkovno bazo strokovnjak – **načrtovalec baze podatkov**, v preprostih primerih pa tudi sam uporabnik, če le pozna nekaj osnovnih pravil dobrega načrtovanja. Potrebno pa je opozoriti, da je tudi v primeru, ko bazo načrtuje strokovnjak, potrebno pri tem sodelovanje uporabnika, saj *pravilno organizirani podatki odražajo poslovna pravila* področja, ki ga baza pokriva. Res je, da večji del poslovnih pravil identificiramo že v fazi analize oziroma podatkovnega, a le redko prav vsa, ki so potrebna za načrtovanje.

Tvorba podatkovne baze pomeni, da v podatkovni slovar, ki je v primeru podatkovne baze kar njen sestavni del, vnesemo definicijo oziroma opis podatkovne baze, tako kot smo to opredelili z načrtom podatkovne baze (glej poglavje Upravljanje podatkov).

Čeprav se zdi **vnos začetnih podatkov**, to je trenutnega opisa realnega sveta, precej rutinsko opravilo, v praksi mnogokrat pri tem naletimo na obilo težav, posebej če podatke v novo podatkovno bazo prenašamo iz obstoječih virov (starih podatkovnih baz, datotek ...). Drugače zapisani datumi, nekonsistentnost različnih virov (npr. različne šifre za iste entitete), uporaba decimalnih pik namesto decimalnih vejic, pomanjkljivi podatki o že izvedenih transakcijah in podobno so

razlogi, zaradi katerih lahko postane ta aktivnost zelo zamudna, saj jo je težko popolnoma avtomatizirati.

Zelo pomembno je tudi redno **vzdrževanje** podatkovne baze, ki je naloga **skrbnika podatkovne baze** (angl. Database Administrator – DBA). Če sta analitik in načrtovalec podatkovne baze lahko zunanja sodelavca, pa je skrbnik podatkovne baze praviloma zaposlen v organizaciji, saj je pogosto potrebno hitro ukrepanje. Seveda ima lahko v manjših organizacijah ista oseba tudi druge naloge. Med naloge skrbnika podatkovne baze sodijo (med drugimi) tudi:

- izvajanje vseh vrst zaščit, kot jih opredeljuje politika zaščite podatkov (glej poglavje Upravljanje podatkov), od izdelave rezervnih kopij podatkovne baze do dodeljevanja uporabniških imen in z njimi povezanih pravic dostopa do podatkov,
- po potrebi spremembe strukture podatkov,
- izvajanje zahtevnejših ad hoc poizvedb in poročil po naročilu uporabnikov,
- nastavitve, ki omogočajo optimalno delovanje, kar uporabnik zazna kot ustrezno kratke odzivne čase oziroma hitre odgovore na poizvedbe,
- občasno čiščenje podatkov, ki so označeni kot zbrisani ...

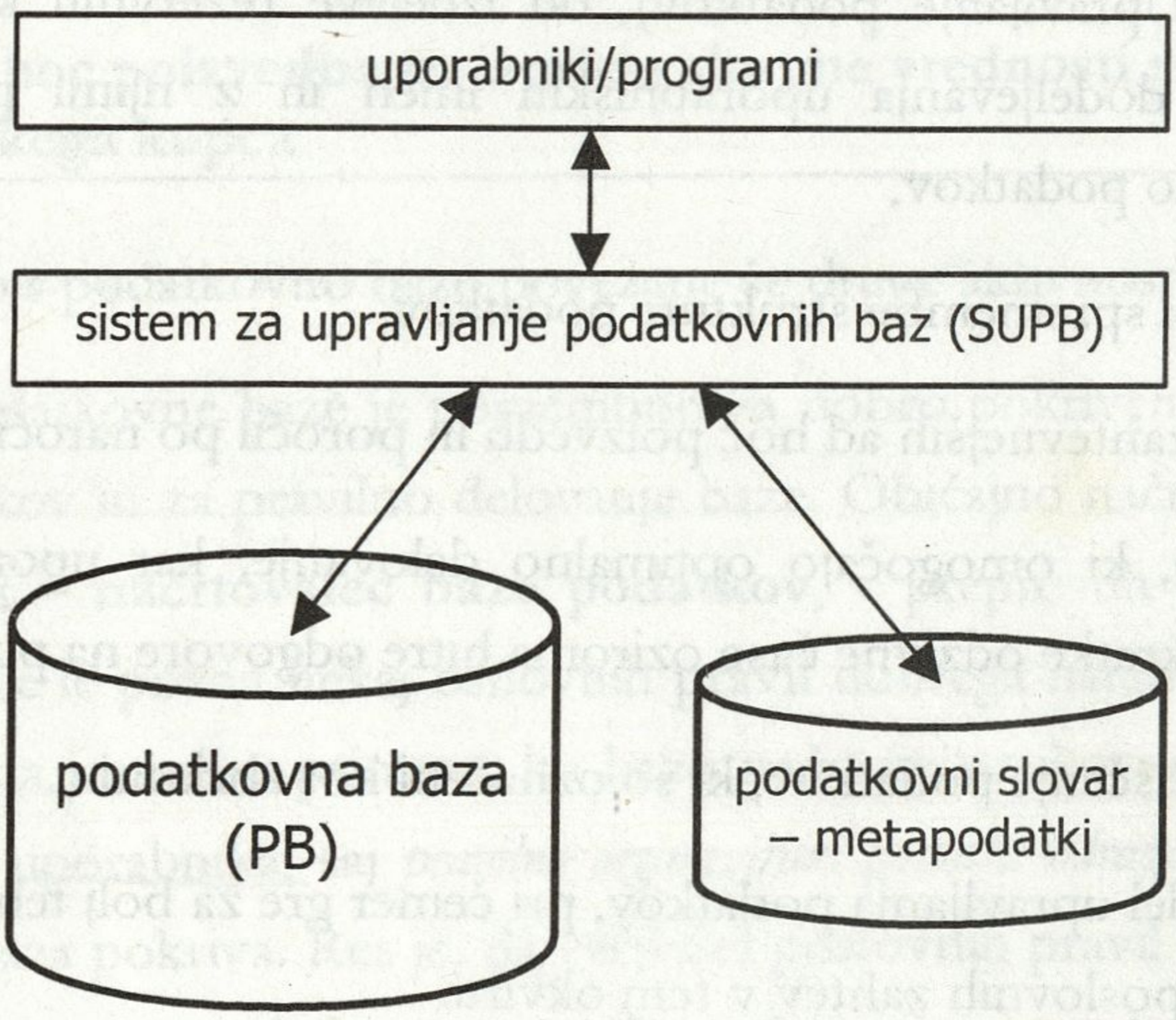
Vzdrževanje je del upravljanja podatkov, pri čemer gre za bolj tehnični del oziroma implementacijo poslovnih zahtev v tem okviru.

Sistemi za upravljanje podatkovnih baz

Sistem za upravljanje podatkovnih baz (SUPB, angl. Database Management System, DBMS) je *zbirka programov*, ki omogočajo *tvorbo, uporabo in vzdrževanje podatkovnih baz*. Z njimi torej izvajamo vse aktivnosti, povezane s podatkovno bazo, razen načrtovanja, kateremu so namenjena druga programska orodja (CASE, glej poglavje v nadaljevanju). To pomeni, da so SUPB namenjeni raznolikim uporabnikom, od načrtovalca podatkovne baze, programerja, skrbnika, do uporabnika. Zato v mnogih primerih SUPB res sestavlja množica programov, vsak od njih pa je namenjen določeni aktivnosti v povezavi s podatkovno bazo.

Preprostejši SUPB pa so integrirani, vse aktivnosti opravimo z enim samim programom.

Značilnost uporabe podatkovnih baz je predvsem ta, da uporabnik (ali program) ne more neposredno dosegati ali spreminjati podatkov v bazi. S tem je dosežena **neodvisnost med programi in podatki**, ki jo omogočajo prav SUPB. Podatke lahko dosegamo in spreminjamo samo tako, da SUPB posredujemo zahtevo, ta pa le-to izvede, kar pomeni, da spremeni podatke ali poišče in vrne odgovor na postavljeno poizvedbo. Neodvisnost med programi in podatki je prikazana na sliki 19.

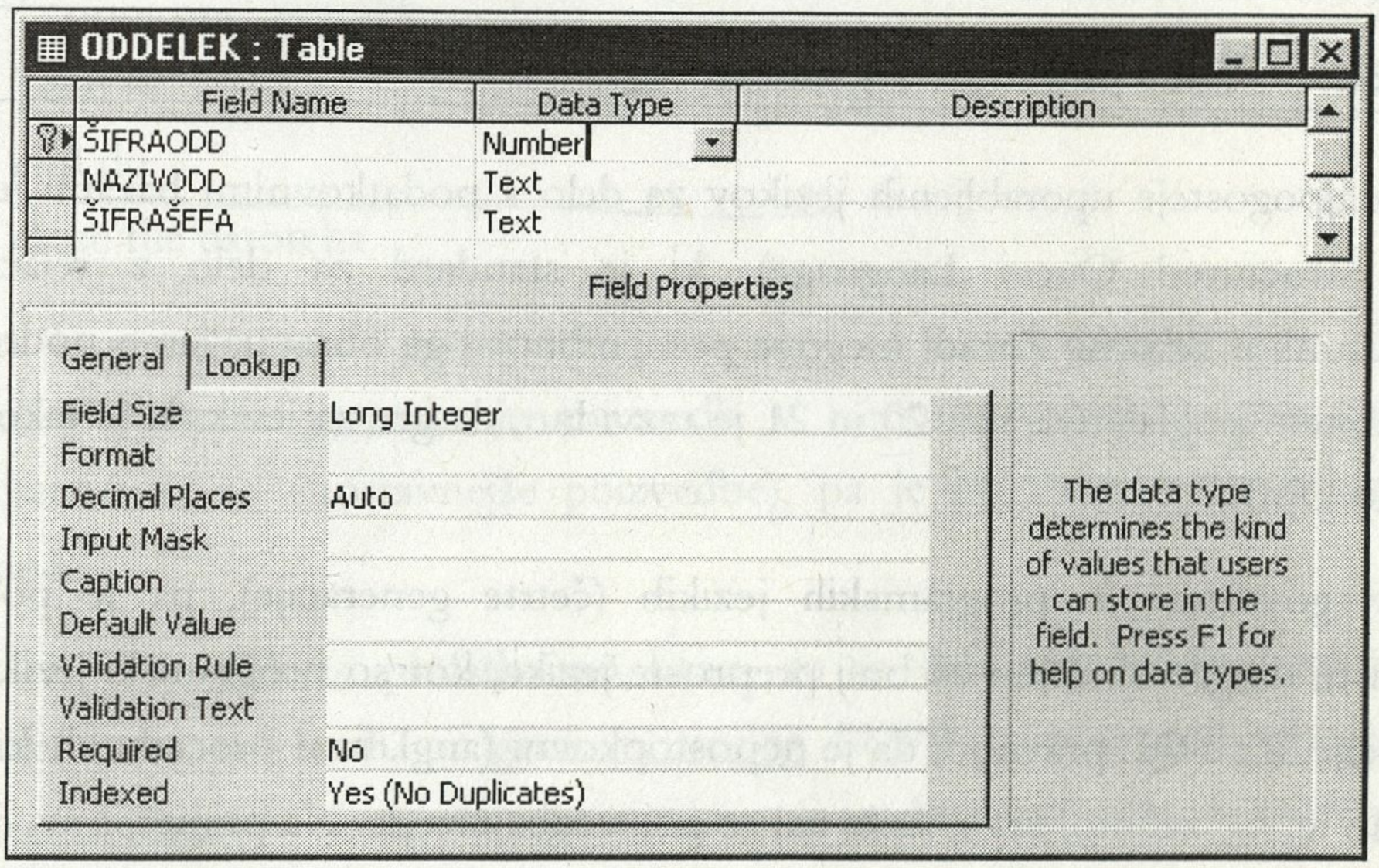


Slika 19: Uporaba sistemov za upravljanje podatkovnih baz

Ta neodvisnost je dosežena s podatkovnim slovarjem, v katerem je shranjen opis podatkovne baze: njene strukture, tipov podatkov ... Kot že vemo, tem podatkom pravimo metapodatki. V primeru podatkovne baze je podatkovni slovar tudi shranjen v računalniški obliki in je običajno kar del podatkovne baze.

Programi torej ne poznajo strukture podatkov, saj dostopajo do podatkov preko SUPB, ki pa uporablja strukturo podatkov, zapisano v podatkovnem slovarju.

Pri *tvorbi* uporabljamo za vnos opisa podatkovne baze (in kasnejšega spreminjanja le-tega), glede na SUPB, enega izmed **jezikov za definiranje podatkov** – DDL (angl. Data Definition Language). Za enostavnejše delo pa ima večina SUPB tudi grafične uporabniške vmesnike za definiranje in spremembo strukture baze.



(a)

```
CREATE TABLE ODDELEK (
    ŠIFRAODD INTEGER,
    NAZIVODD VARCHAR(20),
    ŠIFRAŠEFA CHAR(6),
    CONSTRAINT ODDPK PRIMARY KEY (ŠIFRAODD)
);
```

(b)

Slika 20: Definicija dela podatkovne baze (ene tabele) z uporabo grafičnega uporabniškega vmesnika (a) in z jezikom SQL (b)

Uporaba

SUPB omogočajo tudi vsa opravila povezana z uporabo podatkovne baze: vnos, spreminjanje (ažuriranje), brisanje ter poizvedovanje. Kot smo že poudarili, lahko uporabnik (preko programa ali uporabnik neposredno) to počne *le* skozi SUPB, kar pomeni, da morajo obstajati pravila sporazumevanja med uporabniki in SUPB

– jeziki. Jeziki, ki so namenjeni delu s podatki (angl. data manipulation), se imenujejo **jeziki za manipulacijo podatkov** (angl. Data Manipulation Language – DML).

```
INSERT INTO ODDELEK VALUES (2233, 'Trženje', '645231')
```

Slika 21: Vnos podatkov z jezikom SQL

Eden najpogosteje uporabljenih jezikov za delo s podatkovnimi bazami je **SQL** (angl. Structured Query Language), ki je standard za delo z relacijskimi podatkovnimi bazami. Zaradi njegove pomembnosti ga obravnavamo podrobneje v posebnem poglavju. S slik 20 in 21 je razvidno, da gre za jezik, ki je tako DDL kot tudi DML.

Čeprav govorimo o programskih jezikih (četrte generacije), pa je potrebno opozoriti, da gre običajno za bolj preproste jezike, kot so programski jeziki tretje generacije. Za SQL pravimo, da je nepostopkovni (angl. non procedural) jezik, saj povemo le, *kaj* želimo, *ne pa*, *kako* naj se poizvedba izvede. Na primer, s stavkom

```
SELECT IME, PRIIMEK
FROM DELAVEC
WHERE PLACA > 100000;
```

zahtevamo imena in priimke vseh delavcev, katerih plača je večja od 100.000. Verjetno za bralca, ki pozna nekaj angleškega jezika, razlaga stavka sploh ni potrebna. V jeziku tretje generacije bi morali zapisati postopek, kako naj se poizvedba izvede, na primer[18]:

18 Za prikaz primera ni uporabljen konkreten programski jezik, pač pa so uporabljeni tipični konstrukti programskih jezikov tretje generacije.

```
datoteka = open file (Delavec)
do while not eof(datoteka)
  readln (sifra, priimek, ime, placa, oddelek)
  if placa > 100000 then
    print ime, priimek
  end if
end do
close file datoteka
```

Razlika, ki je prikazana s preprostim primerom, se v zapletenejših poizvedbah še poveča. Čeprav je zastavljanje poizvedb v nepostopkovnih jezikih relativno preprosto (vsaj za enostavnejše poizvedbe), pa je po drugi strani včasih tudi omejujoče.

Za t.i. ad hoc delo s podatkovno bazo, to je delo, za katerega ni vnaprej predvidenih in izdelanih programov, večina SUPB vsebuje tudi uporabniško prijazne vmesnike, ki omogočajo poizvedovanje in druga opravila, ne da bi poznali programski jezik.

Vzdrževanje

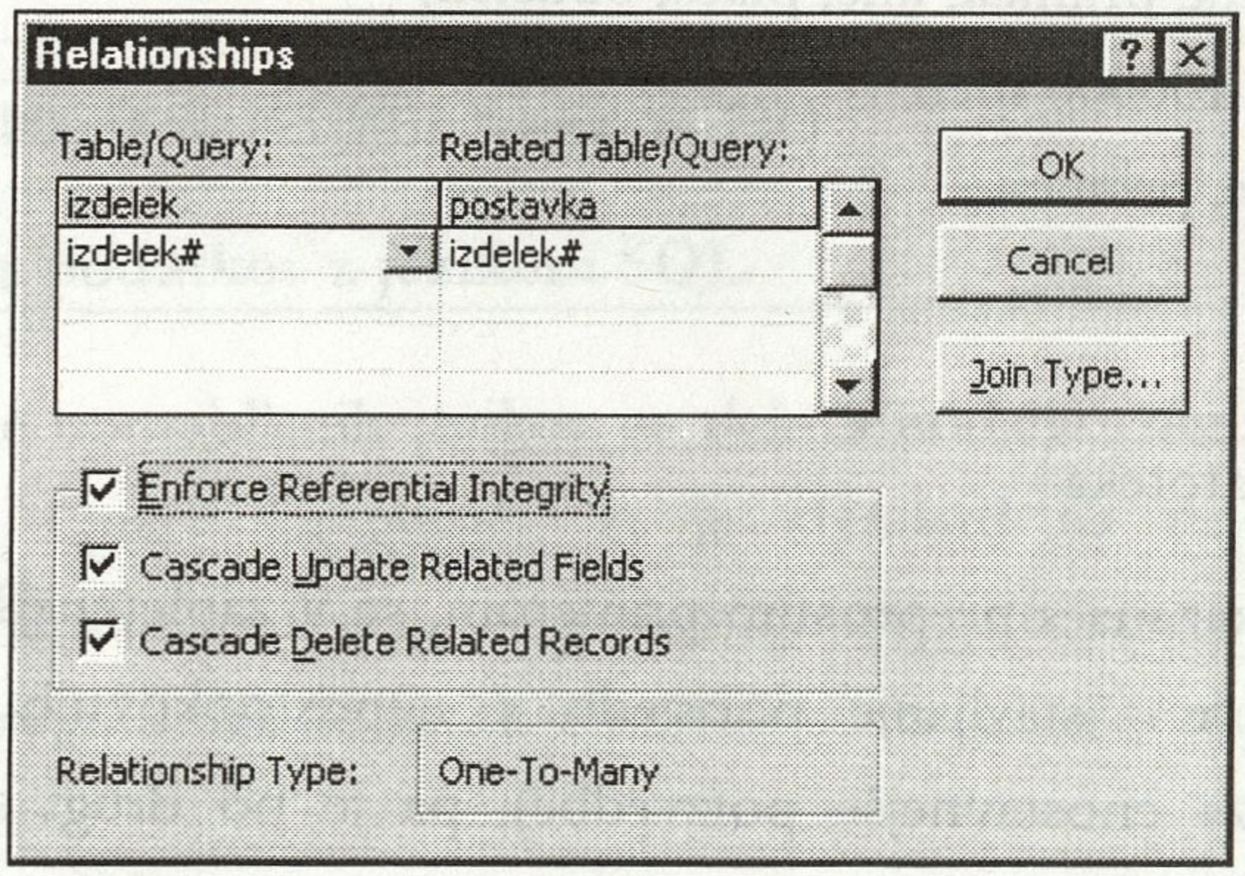
V okviru SUPB je del programov oziroma del programa namenjen tudi aktivnostim, povezanim z vzdrževanjem oziroma skrbništvom podatkovne baze. Tudi ta del SUPB postaja čedalje bolj preprost za uporabo in uporabniško prijazen, hkrati pa omogoča tudi avtomatizacijo nekaterih opravil (npr. redno izdelavo rezervnih kopij).

Mnogi SUPB omogočajo nadzor **celovitosti podatkov** (angl. data integrity). Gre za semantične (pomenske) omejitve, ki jih lahko načrtovalec definira na podatkovni bazi. Na primer, v tabeli s podatki o postavkah naročil ne moremo imeti vrstice s šifro izdelka (IZDELEK#), ki ne obstaja v tabeli s podatki o izdelkih. To pravilo je odraz očitnega poslovnega pravila, ki pravi, da kupci ne morejo naročiti izdelkov, ki jih ne prodajamo.

Omejitve, ki jih načrtovalec podatkovne baze definira, morajo izvirati iz realnega sveta in so dejansko pravila, ki veljajo na poslovnem področju, ki ga opisujejo

semantičen = pomenski

podatki v bazi. SUPB z nadzorom nad upoštevanjem teh pravil onemogoča uporabniku, da bi *nenamerno* privedel podatkovno bazo v nedovoljeno stanje.

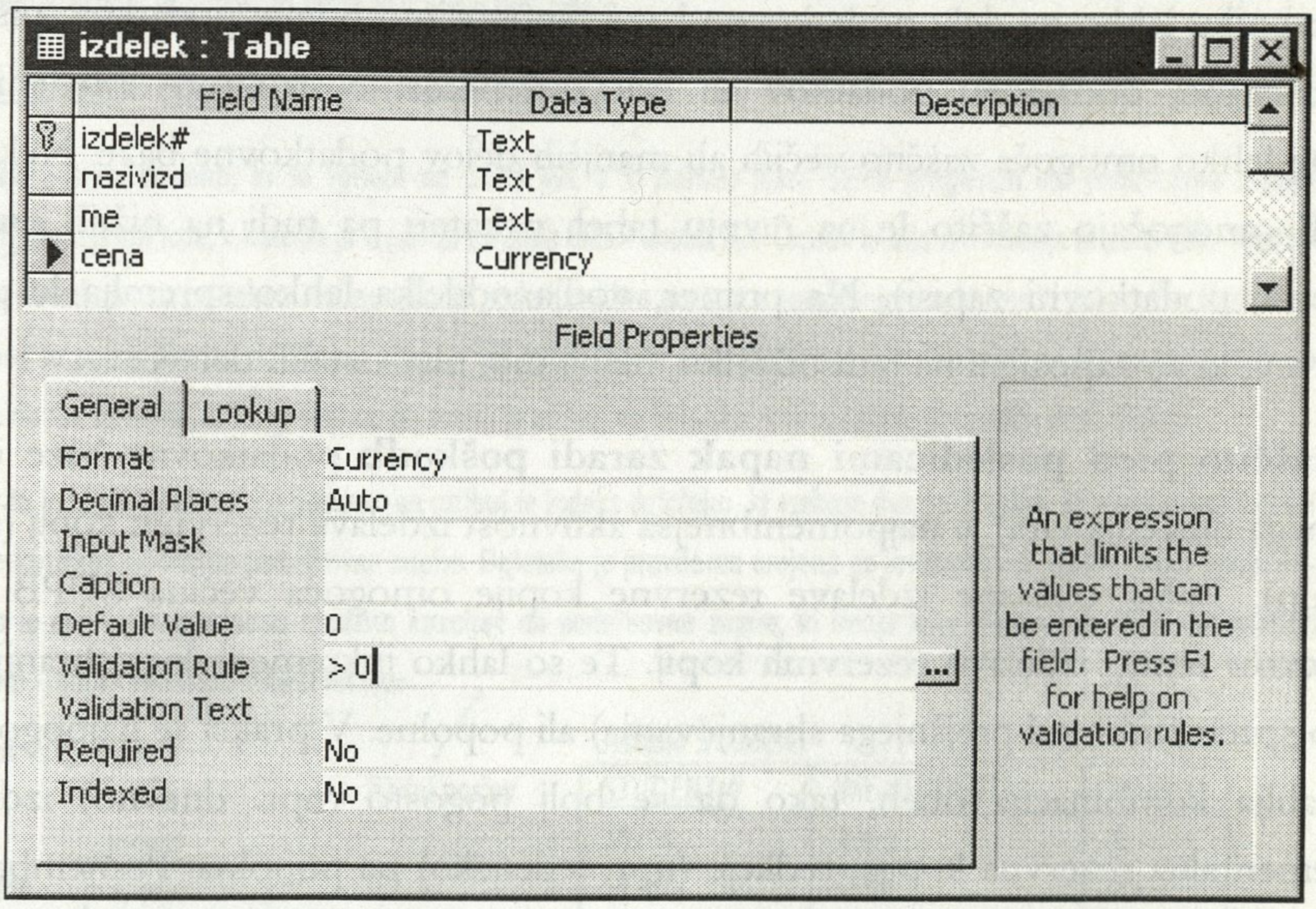


Slika 22: Definiranje celovitosti

V grobem lahko omejitve celovitosti delimo na tri vrste:

- omejitve območja,
- referenčna celovitost in
- eksplicitna celovitost.

Omejitve območja določajo, kakšne vrednosti lahko zavzame atribut. Na primer, šifre študentov so lahko cela števila med 1 in 99999, poštne številke lahko zavzamejo vrednosti med 1000 in 9999 in spol je lahko 'M' ali 'Ž'. V SUPB, ki *ne* omogočajo preverjanja omejitve območja, za vsak atribut definiramo le tip, ki je lahko eden izmed vgrajenih tipov. Na sliki 23 je prikazan primer definiranja pravila (Validation Rule), da mora biti cena izdelka pozitivno število (>0).



Slika 23: Nadzor celovitosti – omejitve območja

Referenčna celovitost določa, da lahko nek atribut zavzame le vrednosti, ki jih ima ustrezni ključ. Na primer, podatki o opravljenih izpitih vsebujejo tudi vpisno številko študenta, da vemo, kdo je opravil izpit. Vpisna številka je hkrati tudi ključ za entitetni tip ŠTUDENTI. Očitno je, da ne more biti med podatki o ocenah vpisne številke, ki je nima nihče od študentov, saj bi to pomenilo, da je izpit opravljal nekdo, ki ni študent. Če definiramo ustrezno referenčno omejitev, bi pri (nenamernem ali morda namernem) poskusu vnosa, da je izpit opravil neobstoječi študent, SUPB tak vnos preprečil.

Le redki SUPB omogočajo nadzor nad **eksplicitno celovitostjo**. Zahteve (pogoji in dovoljene vrednosti) so pri eksplicitni celovitosti navadno definirane s konkretnimi vrednostmi. Vzemimo za primer, da je predpogoj za opravljanje izpita pri predmetu Informatika 2 opravljen izpit pri predmetu Informatika 1. Potem lahko postavimo pogoj, da v tabelo s podatki o opravljanju izpitov ni mogoče vnesti vrstice s šifro predmeta IN200 (Informatika 2), če se v tej relaciji ne nahaja že vrstica z isto šifro študenta in s šifro predmeta IN100 (Informatika 1).

V okviru **zaščite pred nepooblaščenim dostopom** skrbnik vsakemu uporabniku dodeli *uporabniško ime*, na katerega so vezane pravice dostopa, ki jih preverja SUPB.

Uporabniku lahko na delu podatkovne baze omogočimo branje podatkov, pisanje (spremembo, dodajanje) podatkov ali celo spreminjanje strukture. Mehanizem zaščite lahko omogoča zaščito večjih ali manjših delov podatkovne baze. Nekateri SUPB omogočajo zaščito le na nivoju tabel, nekateri pa tudi na nižjih nivojih (atributi, podatkovni zapisi). Na primer, vodja oddelka lahko spremlja le plače delavcev, ki so zaposleni na tem oddelku, ne pa tudi plač ostalih delavcev.

Za **zaščito pred posledicami napak zaradi poškodb** podatkovne baze (npr. fizičnega uničenja vira) je najpomembnejša aktivnost izdelava rezervnih kopij (angl. backup). Poleg občasne izdelave rezervne kopije omogoča večina SUPB tudi *avtomatsko* redno izdelavo rezervnih kopij. Te so lahko inkrementalne (shranijo se samo spremembe od prejšnjega shranjevanja) ali popolne. V praksi se najpogosteje uporablja kombinacija obeh, tako da se bolj pogosto (npr. dnevno) izdeluje inkrementalna rezervna kopija, redkeje (npr. tedensko) pa popolna. Pomembno je seveda, da je rezervna kopija shranjena na drugem mediju kot original, da bi bilo tako mogoče obnoviti original tudi v primeru poškodbe medija. Kako pogosto izdelujemo rezervne kopije, mora biti opredeljeno z načrtom zaščite podatkov, kar je poslovna odločitev. V redkih primerih, ko je zahtevana neprekinjena možnost dostopa do podatkovne baze, imamo ves čas dve enaki kopiji le-te. Ta pristop imenujemo **zrcaljenje** (angl. mirroring).

V primeru poškodbe originalne podatkovne baze le-to rekonstruiramo iz rezervne kopije z modulom za **obnavljanje** (angl. recovery).

V vsaki večji podatkovni bazi so pomemben element tudi indeksi[19], ki omogočajo sistemom za upravljanje podatkovnih baz bolj učinkovito izvajanje poizvedb. Uporabnik indekse torej občuti kot praviloma krajše odzivne čase pri poizvedovanju. Namen indeksov je podoben, kot je namen stvarnih kazal (indeksov) v knjigah. V stvarnih kazalih so pomembnejši pojmi, ki se pojavljajo v knjigi, urejeni po abecedi, kar nam omogoča hitrejše iskanje teh pojmov.

19 Ker so lahko podatkovne strukture, ki jih uporabljamo za implementacijo indeksov, precej zapletene, tukaj prikazujemo indekse precej poenostavljeno in le v tolikšni meri, da bi lahko bralec razumel njihov namen.

Vzemimo za primer podatkovno bazo, ki vsebuje tudi tabelo

ŠTUDENT (ŠTUDENT#, IME-ŠTUDENTA, SMER)

Zanimajo nas vsi študenti, ki so vpisani na smeri MA. V ta namen mora SUPB pregledati vse podatkovne zapise relacije ŠTUDENT in izpisati tiste, v katerih je vrednost atributa SMER enaka MA. Čeprav bi bilo izvrševanje te poizvedbe relativno hitro, pa časovna zahtevnost pri takem načinu iskanja podatkov hitro naraste z naraščanjem zahtevnosti poizvedbe. Posebej zahtevne (časovno) so poizvedbe, ki vključujejo dve ali več relacij. Če bi imel SUPB na voljo "kazalo", ki bi imelo za vsako od smeri kazalce na vse študente, ki so vpisani na tej smeri, je očitno, da bi lahko vrnil rezultate poizvedbe dosti hitreje.

In prav to je ideja indeksov. Za posamezen atribut je indeks datoteka, ki vsebuje dve polji: polje, ki vsebuje vrednosti atributa, in polje kazalcev na fizične podatkovne zapise. Datoteka je praviloma urejena po vrednostih atributa. Vsaka vrednost atributa nastopa le enkrat in ji pripada množica kazalcev na podatkovne zapise, ki imajo tako vrednost atributa. Shematično lahko prikažemo indeks za atribut SMER takole:

	indeks
MA	2, 3
IS	1
RA	10
FI	4, 11, 13

fizični naslov

relacija ŠTUDENT

	ŠTUDENT#	IME-ŠTUDENTA	SMER
1	38214	Janez	IS
2	69137	Katja	MA
3	93884	Peter	MA
4	23744	Metka	FI
10	87478	Mitja	RA
11	28302	Alenka	FI
13	84332	Mitja	FI

Kot smo že nakazali, lahko indeks zgradimo za vsak atribut, ki nastopa v podatkovni bazi. V mnogih primerih SUPB sam zgradi indekse za ključe relacij[20].

Razlog – hitrost izvajanja poizvedb – za izgradnjo indeksov nad atributi postane pri velikih podatkovnih bazah dovolj močan. Vendar obstajajo, kot ponavadi, tudi razlogi proti. Prva slabost indeksov, ki v mnogih primerih ni zanemarljiva, je prostor na nosilcih podatkov, ki ga zavzemajo. Poleg tega postane zahtevnejši vnos novih zapisov, saj moramo ob vsaki taki spremembi ustrezno spremeniti tudi vse indekse[21], ki so zgrajeni nad atributi relacije, v katero vnašamo nov zapis.

Ko se odločamo o izgradnji indeksa, se moramo vprašati:

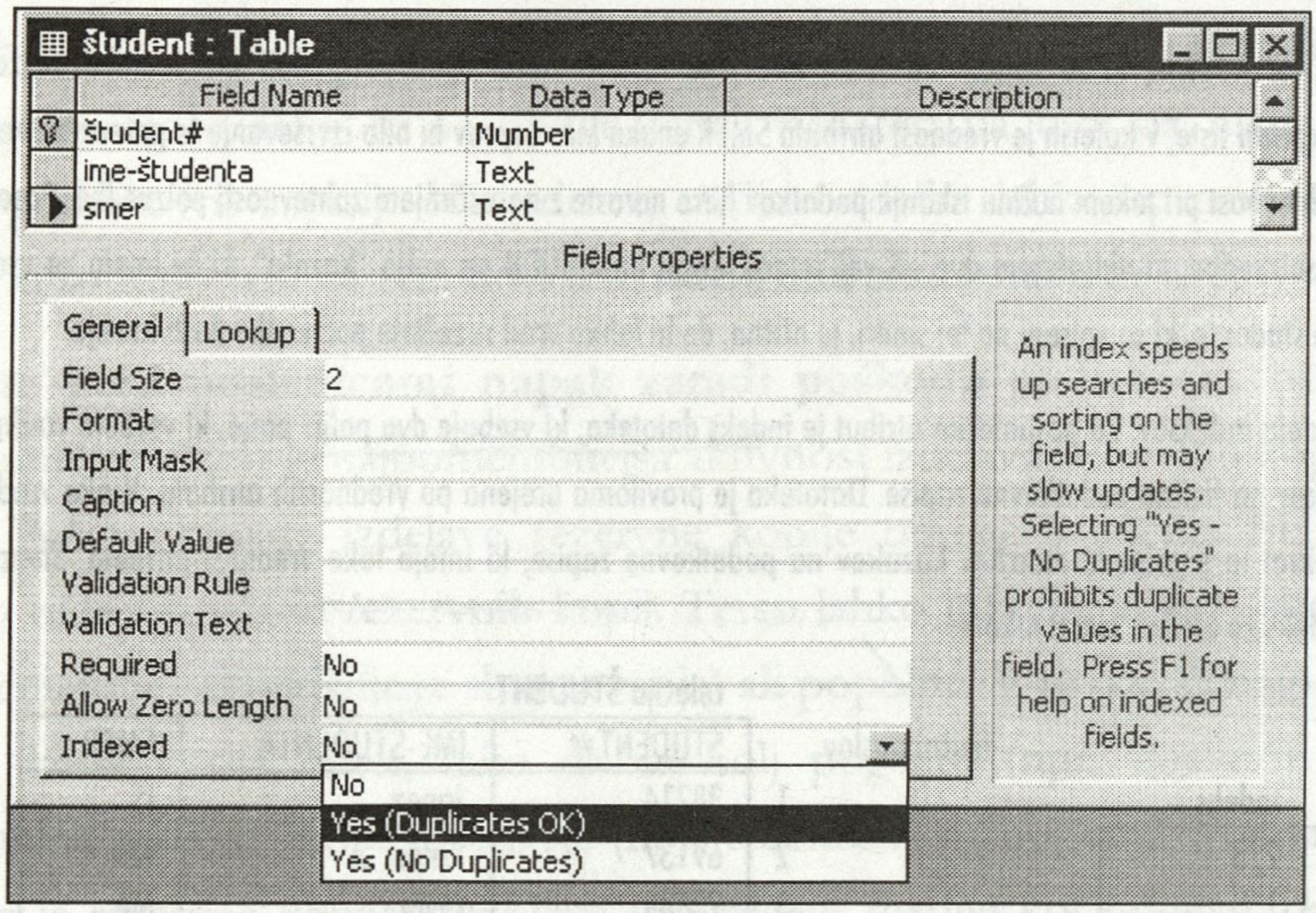
- Ali imamo na voljo dovolj prostora na nosilcu podatkov?
- Kako pogosto bomo izvajali poizvedbe in kako zahtevne bodo?
- Kako pogosto bomo spreminjali podatke v relaciji, v kateri je atribut?

---

20 To je tudi razlog, da mnogi uporabniki ne razlikujejo med ključi in indeksi.

21 Večinoma SUPB ob spremembi podatkov avtomatično ustrezno ažurirajo tudi indekse. Čeprav uporabnik z ažuriranjem indeksov nima dodatnega dela, pa lahko pri vnosu ali spremembi podatkov občuti podaljšane odzivne čase podatkovne baze.

V SUPB MS Access zgradimo indekse preprosto tako, da pri definiciji tabele za vsak atribut izberemo, ali naj bo indeksiran, ali ne. Za glavni ključ je privzeta vrednost DA – indeks naj bo zgrajen.



Poleg opisanih najpomembnejših funkcij imajo SUPB še mnoge druge funkcije, ki omogočajo zanesljivo in hitro uporabo podatkovnih baz (npr. optimizacijo izvajanja poizvedb in podobno).

Vloge

Čeprav smo o tem razpravljali že sproti, povzemimo še enkrat vloge oseb, ki so povezane s podatkovnimi bazami.

- **Analitik** ugotavlja informacijske potrebe uporabnikov, poslovna pravila, semantiko podatkov ...
- Na podlagi njegovih ugotovitev izdela **načrtovalec** podatkovne baze načrt le-te.
- Podatkovno bazo tvori programer ali skrbnik podatkovne baze.
- **Programer** tudi izdela programe, ki dostopajo do baze.
- Naloge **skrbnika podatkovne baze** so mnogotere: od skrbi za dostopnost, odzivnost in varnost podatkovne baze, preko izdelave zahtevnejših ad hoc

poizvedb, do izobraževanja uporabnikov in izbora sistema za upravljanje podatkovnih baz.

Velja opozoriti, da ima lahko ista oseba več vlog, v kompleksnejših primerih pa je specializacija smiselna in smotrna.

Na koncu (pravzaprav na začetku) je tu še **uporabnik**, kateremu je podatkovna baza namenjena. Njegova naloga je, da poskrbi za:

- **ažurnost,**
- **točnost** in
- **popolnost**

podatkov v podatkovni bazi, hkrati pa njihovo **uporabo** za boljše poslovanje.

3.2 Načrtovanje podatkovnih baz

Tako kot to velja za razvoj vsakega izdelka, je tudi pri podatkovni bazi izjemnega pomena izdelava dobrega načrta. Načrtovanje podatkovne baze je v veliki meri tehnično opravilo, ki ga praviloma opravi strokovnjak – načrtovalec podatkovne baze, vendar tudi v tej fazi razvoj podatkovne baze ni mogoč brez sodelovanja uporabnika. Res pa je, da je sodelovanje uporabnika tu manjše kot pri analizi problema.

Programska oprema za delo s podatkovnimi bazami je danes v večini primerov preprosta za uporabo, deluje v uporabniško prijaznih okoljih, zaradi podobnosti delovnega okolja pa je tudi delo v veliki meri podobno delu v drugih programih (urejevalnikih besedil, programskih paketih za delo s preglednicami). Zato mnogi uporabniki menijo, da lahko izdelajo podatkovno bazo le z dobrim (ali celo osnovnim) poznavanjem programskega paketa za delo s podatkovnimi bazami, a brez poznavanja osnovnih in najpomembnejših pravil za načrtovanje podatkovnih baz. Podatkovna baza, ki nastane na ta način, lahko povzroča različne težave pri uporabi in vzdrževanju, na primer, ko uporabnik išče določene podatke, dobi napačne odgovore.

Ker je po eni strani sodelovanje uporabnika pri načrtovanju podatkovnih baz manjše in samostojno načrtovanje le redko, po drugi strani pa je razumevanje načel organizacije podatkov pomembno tudi za kasnejšo uporabo, bomo v nadaljevanju spoznali:

- *kako moremo organizirati podatke* ter
- kakšna so temeljna pravila dobre organizacije podatkov, torej *kako moramo organizirati podatke.*

Pri tem bomo posebej poudarili tista mesta, pri katerih načrtovalec ne more dobro opraviti svojega dela brez sodelovanja uporabnika. Hkrati bomo načrtovanje spoznali v tolikšni meri, da bo uporabnik vedel, na kaj mora biti pozoren, če želi samostojno izdelati preprosto podatkovno bazo.

Podatkovni modeli

Da bi spoznali, kakšne imamo možnosti organizacije (strukturiranja) podatkov, najprej opredelimo pojem podatkovni model.

Beseda **model** ima več pomenov, med drugimi tudi (Bajec, 1994 in Grad, Jaklič, 1996):

1. prikaz proučevanega realnega sistema na drugačen, poenostavljen način, ki zaobjame njegovo bistvo, ne pa vseh podrobnosti;
2. ustaljena oblika česa, po kateri se kaj dela; vzorec, oblika, kalup.

Čeprav se v ostalem besedilu pogosteje ukvarjamo z modeli v prvem pomenu, pa bo v kontekstu podatkovnih baz model pomenil "kalup", ki nas omejuje pri strukturiranju podatkov.

Podatkovni model je potemtakem množica pravil, ki določajo, kako smejo biti podatki v podatkovni bazi organizirani oziroma strukturirani. Na primer, relacijski podatkovni model predpisuje, da smejo biti podatki organizirani v tabelah, kjer vsak stolpec tabele predstavlja en atribut, ena vrstica pa eno entiteto oziroma povezavo (podrobneje relacijski model obravnavamo v nadaljevanju).

Za podatkovne modele naj bi veljali dve nasprotujoči si zahtevi:

- omogoča naj tak opis problemskega področja s podatki, s katerim bo mogoče izraziti čimveč podrobnosti,
- hkrati pa naj bo opis čimbolj preprost, da je še obvladljiv.

V programski opremi za delo s podatkovnimi bazami se je v dosedanjem razvoju uveljavilo pet podatkovnih modelov:

- **hierarhični** podatkovni model,
- **mrežni** podatkovni model,
- **relacijski** podatkovni model,
- **objektni** model podatkov in procesov ter
- **večdimenzionalni** podatkovni model.

Modeli so navedeni v kronološkem vrstnem redu, tako kot se je njihova uporaba uveljavljala v praksi. Trenutno se v *poslovnih* informacijskih sistemih največ uporablja relacijski podatkovni model, saj je enostaven in razumljiv uporabnikom, hkrati pa lahko z njim dovolj dobro opišemo poslovni sistem in njegovo delovanje. V podatkovnih virih, namenjenih poslovnemu odločanju, predvsem v področnih podatkovnih skladiščih, pogosto srečamo tudi večdimenzionalni model, čeprav je mnogokrat realiziran kar v relacijski podatkovni bazi. Zato bomo največ pozornosti namenili prav relacijskemu in večdimenzionalnemu modelu, medtem ko bomo za ostale spoznali le najpomembnejše značilnosti, ki naj odločevalcu pomagajo pri odločitvi o morebitni uporabi katerega izmed njih glede na potrebe in značilnosti sistema.

Shema je konkretna uporaba podatkovnega modela, torej načrt (shema) neke podatkovne baze, ki je v skladu s predpisi podatkovnega modela. Hkrati pa shema odraža obravnavano problemsko področje, saj je podatkovna baza opis tega problemskega področja. Tako se mora, na primer, povezava med kupci in naročili odražati tudi v shemi kot povezava med *podatki* o kupcih in naročilih.

Podatkovni zapis je skupina vrednosti atributov, ki se nanašajo na isto entiteto ali povezavo in so shranjeni v podatkovni bazi skupaj. Primer podatkovnega zapisa s podatki o kupcu Janezu Novaku:

123456	Janez	Novak

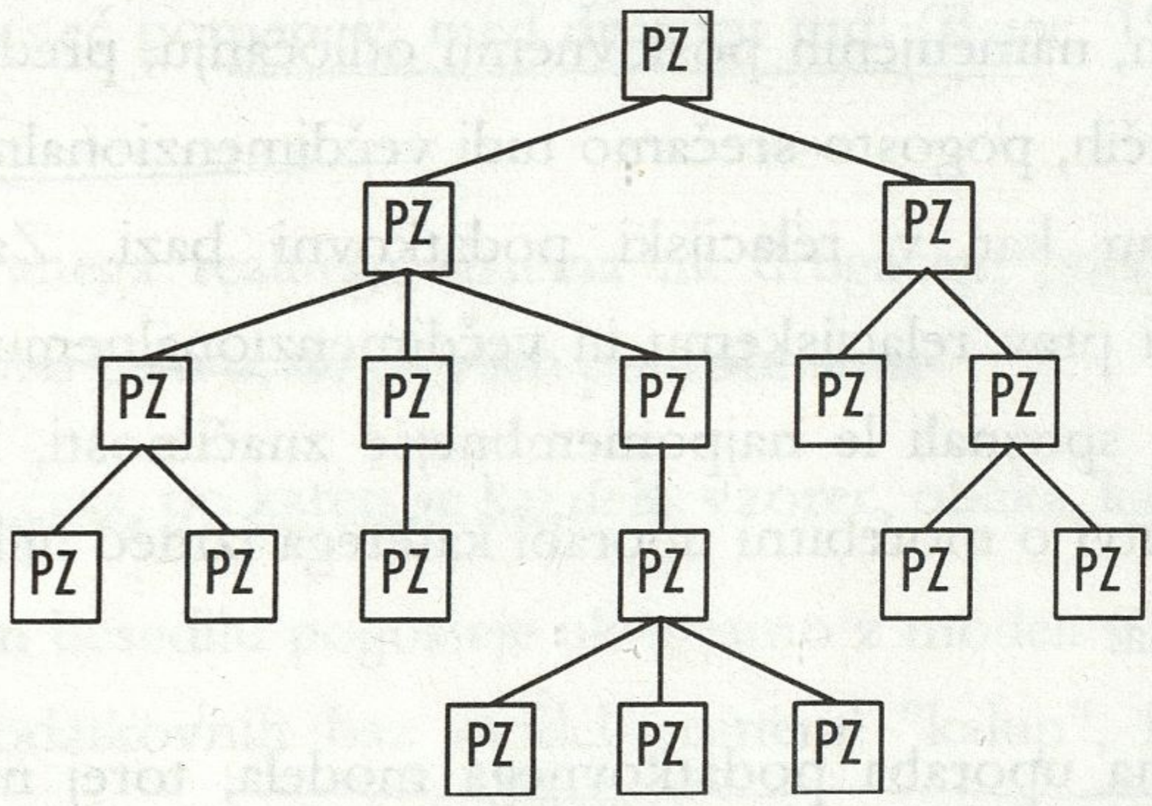
Ti podatki so ponavadi v podatkovni bazi shranjeni skupaj.

Hierarhični model

Najstarejši podatkovni model, ki se je uporabljal v prvih podatkovnih bazah v 1960. letih, je hierarhični podatkovni model. Ta, kot pove ime, predpisuje *hierarhično* oziroma *drevesno organizacijo podatkovnih zapisov.*

Hierarhija je znana struktura; hierarhija delovnih mest in odločanja je pogosta oblika organizacije, elemente rastlinskega in živalskega sveta razvrščamo v hierarhični sorodstveni strukturi (genealoško drevo)... Hierarhična struktura določa, da ima vsak element v strukturi natanko enega neposrednega prednika, razen t.i. korena, ki nima nobenega prednika. Za podrobnejši opis drevesnih struktur glej (Grad, Dacar, 1993).

Na sliki 24 je prikazan primer hierarhične oziroma drevesne strukture podatkovnih zapisov.

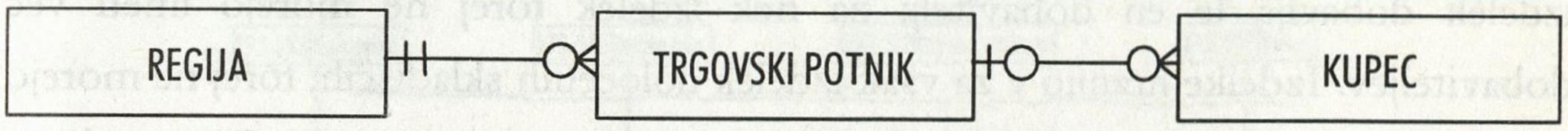


Slika 24: Hierarhična – drevesna struktura podatkovnih zapisov

Iz definicije hierarhične strukture sledi, da lahko na ta način brez težav predstavimo le primere, ko so entitetni tipi med seboj povezani s povezavami, ki so zaporedoma tipa 1:N (v tem vrstnem redu, torej ne N:1) ali 1:1.

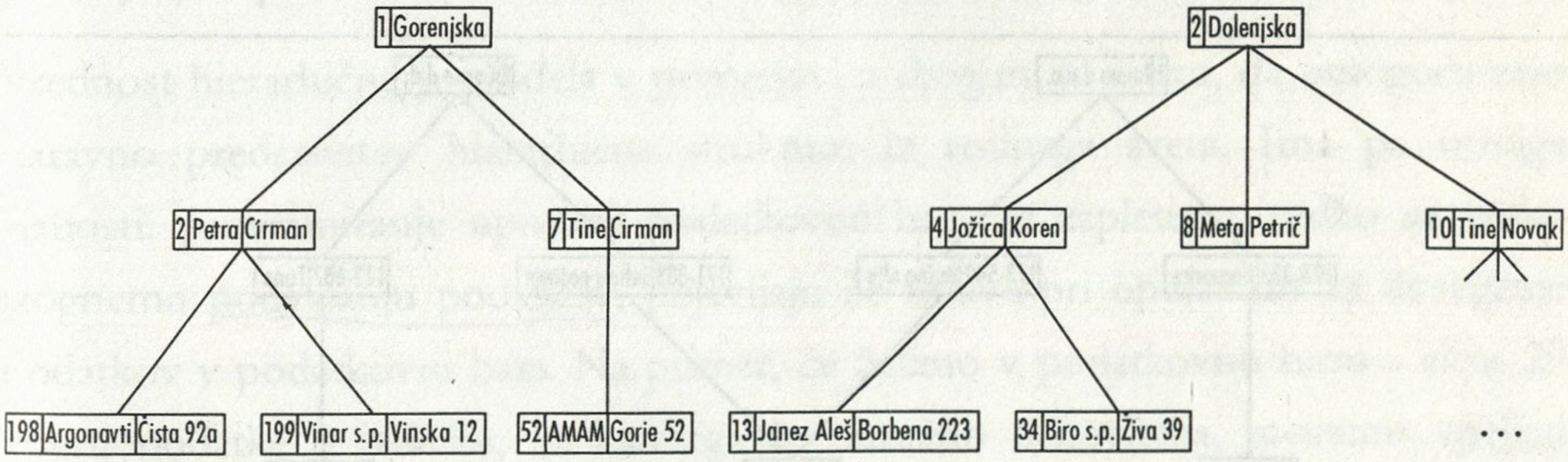
Vzemimo najprej za primer prodajno službo, ki je organizirana po regijah, v vsaki regiji imamo trgovske potnike, vsak od njih pa je zadolžen za obisk določenih

kupcev. Naj veljata še poslovni pravili, da sme vsak trgovski potnik delovati le v svoji regiji in da so kupci tudi razdeljeni med trgovske potnike, z drugimi besedami, da ne more priti do situacije, da bi istega kupca obiskala dva trgovska potnika. Iz ER diagrama na sliki 25 je razvidno, da imamo zaporedoma le povezave tipa 1:N.



Slika 25: ER diagram za primer prodajne službe

Struktura je tipično hierarhična, saj je vsak trgovski potnik lahko povezan le z eno regijo, vsak kupec pa le z enim trgovskim potnikom. Na sliki 26 je prikazan primer hierarhične podatkovne baze. V tem primeru so podatkovni zapisi o regijah predniki podatkovnih zapisov o trgovskih potnikih in podatkovni zapisi o trgovskih potnikih predniki podatkovnih zapisov o kupcih.



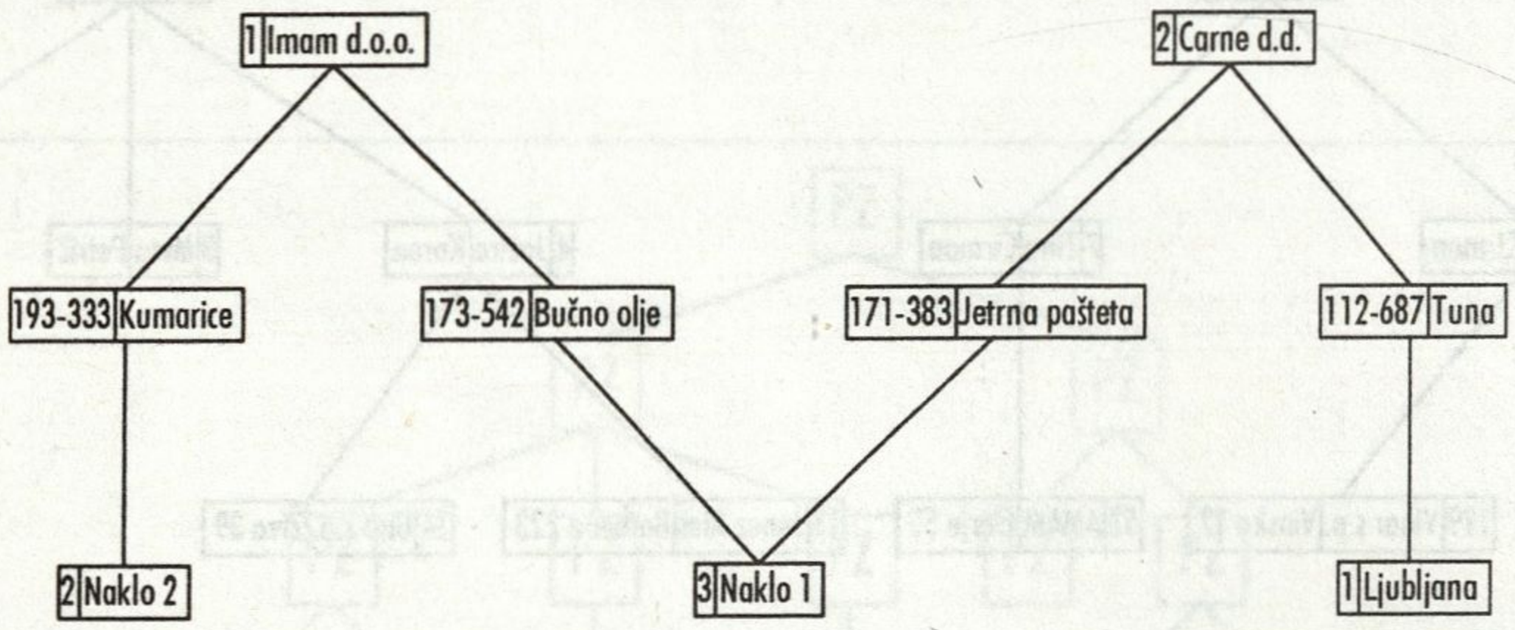
Slika 26: Primer hierarhične podatkovne baze za primer prodajne službe

Težave pa nastopijo, kadar je zaporedje povezav 1:N (ali 1:1) prekinjeno s povezavo N:1 ali M:N. V podjetju želijo hraniti v hierarhični podatkovni bazi podatke o izdelkih, njihovih dobaviteljih in skladiščih, kjer hranimo izdelke. Entitetni tipi bi bili potemtakem IZDELEK, DOBAVITELJ in SKLADIŠČE, med seboj pa naj bodo povezani kot je to prikazano na sliki 27.



Slika 27: ER diagram za primer skladiščenja izdelkov

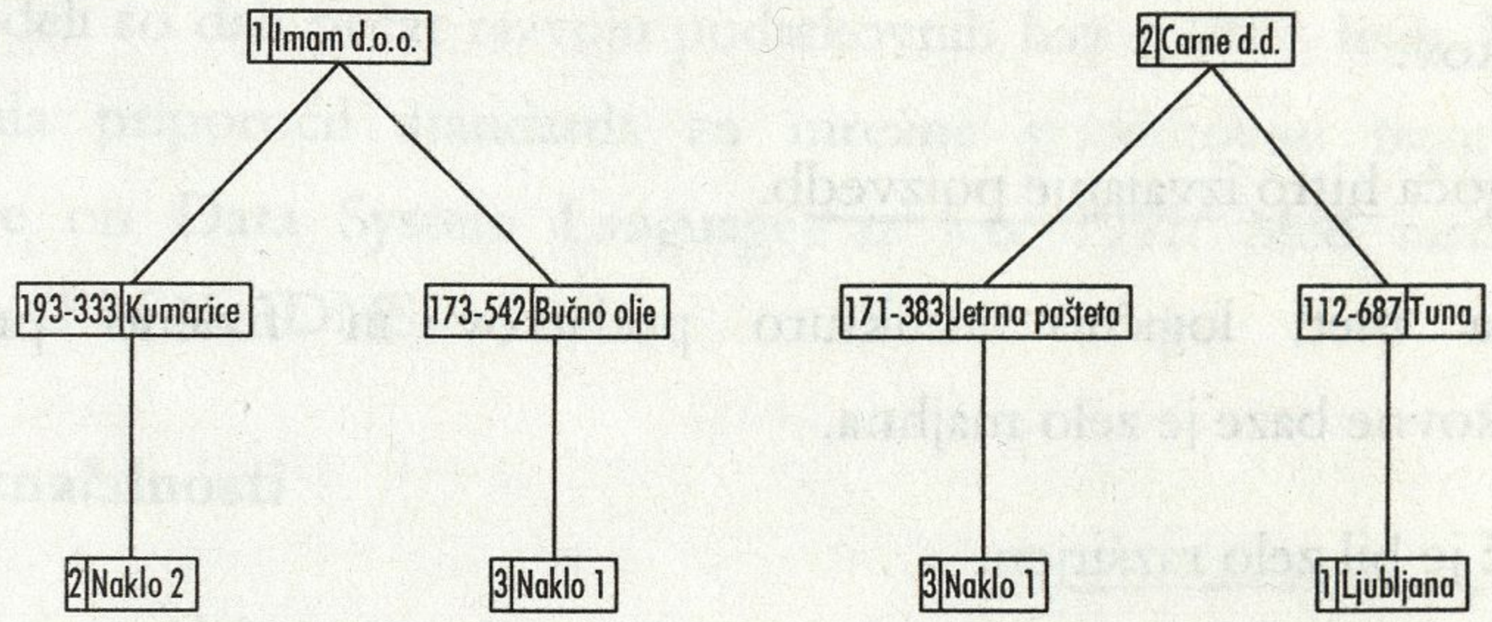
Iz modela je razvidno, da poslovna pravila tega podjetja dovoljujejo, da jim nek izdelek dobavlja le en dobavitelj; za nek izdelek torej ne morejo imeti več dobaviteljev. Izdelke hranijo v za vsak izdelek določenih skladiščih, torej ne morejo imeti istega izdelka v več skladiščih. Lahko pa imajo v enem skladišču različne izdelke. Vidimo torej, da imamo zaporedoma povezavi 1:N in N:1. V hierarhični podatkovni bazi so podatkovni zapisi o dobaviteljih predniki podatkovnih zapisov o izdelkih, ti pa so predniki podatkovnih zapisov o skladiščih. Pri delu baze DOBAVITELJ – IZDELEK nimamo težav, saj ima vsak izdelek samo enega prednika. To pa ne velja za del IZDELEK – SKLADIŠČE, saj je lahko vsako skladišče povezano z več izdelki. Na sliki 28 je prikazan primer podatkovnih zapisov in povezav med njimi.



Slika 28: Primer podatkovnih zapisov in povezav med njimi za primer skladiščenja izdelkov

Očitno je, da taka organizacija podatkov v hierarhični podatkovni bazi *ni dovoljena*, saj imamo primer, ko je en podatkovni zapis (neposredno) povezan z dvema prednikoma. Takšne primere lahko v hierarhični podatkovni bazi razrešimo le s podvajanjem podatkovnih zapisov (in s tem seveda podatkov), kar pa je nezaželena lastnost podatkovne baze (glej podpoglavje Značilnosti podatkovne baze). Na sliki 29 je prikazano, kako bi v hierarhični podatkovni bazi organizirali podatke s slike 28. Vidimo, da se podatki o skladišču Naklo 1 podvajajo, ker v tem skladišču skladiščimo dva izdelka. V realnem primeru bi imeli seveda za skladišča več

atributov, v vsakem skladišču pa veliko izdelkov, kar bi pomenilo mnogo podvojenih podatkov.



Slika 29: Primer hierarhične podatkovne baze za primer skladiščenja izdelkov

Tudi, če bi želeli hierarhično bazo organizirati obratno (skladišča predniki izdelkov, ti pa predniki dobaviteljev) bi naleteli na enak problem. Na podoben način, torej s podvajanjem podatkovnih zapisov, rešimo tudi problem povezav M:N.

Prednost hierarhičnega modela v primerjavi z drugimi je v tem, da omogoča zelo naravno predstavitev hierarhične strukture iz realnega sveta. Ima pa mnoge slabosti: programiranje uporabe podatkovne baze je zapleteno, redko se lahko izognemo podvajanju podatkov, pojavljajo se težave pri operacijah za doseganje podatkov v podatkovni bazi. Na primer, če želimo v podatkovno bazo s slike 29 vnesti podatke o izdelku, za katerega še nimamo dobavitelja, moramo vpeljati namišljenega dobavitelja, saj so podatki o izdelkih vedno vezani na dobavitelja.

Začetki hierarhičnih SUPB segajo v konec šestdesetih let, ko se je na tržišču pojavil IBM-ov Information Management System (IMS), ki velja za njihovega poglavitnega predstavnika. Še danes je eden najhitrejših SUPB, zato se še vedno včasih uporablja za velike podatkovne baze, ki jih označuje veliko število obdelav.

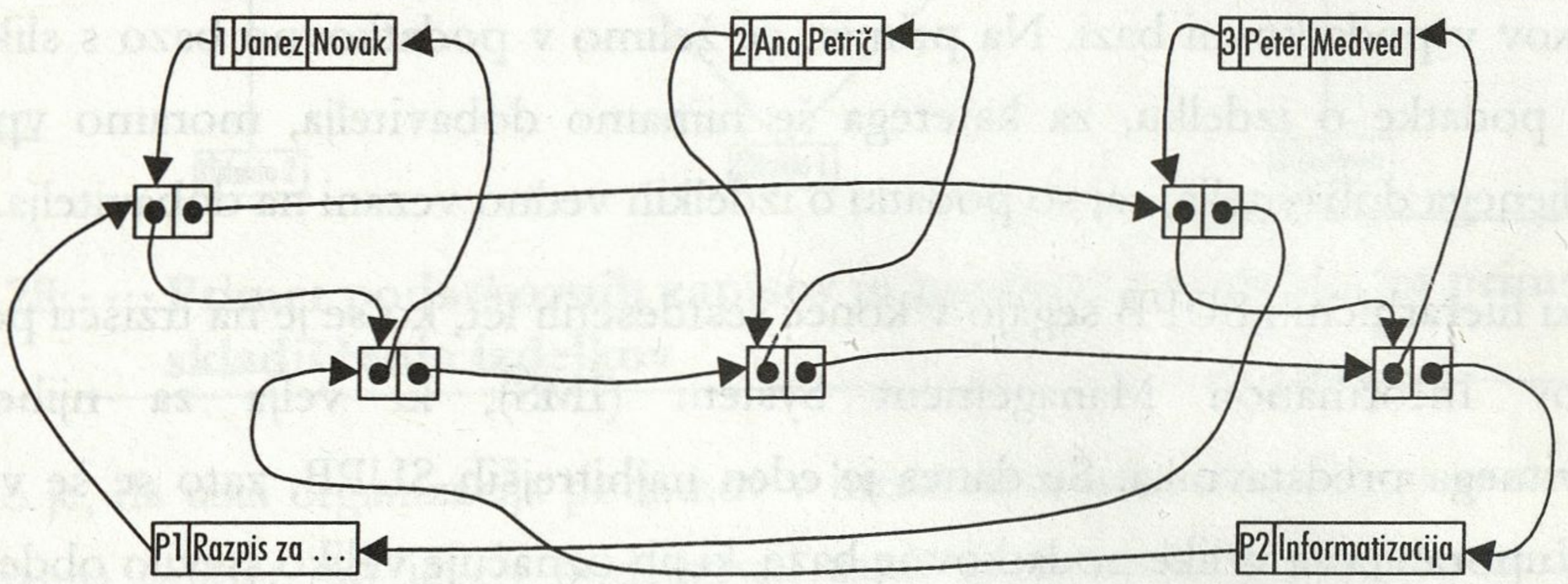
Povzetek značilnosti

- Zaradi hierarhične strukture je skoraj vedno neizogibno podvajanje podatkov.
- Omogoča hitro izvajanje poizvedb.
- Razlika med logično strukturo podatkov in fizično predstavitvijo podatkovne baze je zelo majhna.
- Nekoč je bil zelo razširjen.

Mrežni model

Mrežni model, ki se je uveljavljal predvsem v 1970. letih, odpravlja najpomembnejšo pomanjkljivost hierarhičnega modela, to je podvajanje podatkov. Mrežni model namreč dovoljuje tudi neposredno predstavitev povezav N:1, posredno pa tudi predstavitev povezav M:N brez podvajanja podatkov.

Povezave tipa M:N predstavimo tako, da med prvotne podatkovne zapise vrinemo še povezovalne podatkovne zapise. Na ta način pretvorimo eno povezavo M:N v dve povezavi, eno tipa N:1, drugo pa tipa 1:N. Na spodnji sliki je prikazan primer predstavitve podatkovnih zapisov o zaposlenih in projektih ter povezav med njimi. Vsak zaposleni lahko dela na več projektih, na vsakem projektu pa več zaposlenih.



Vsak zaposleni je lahko povezan z več povezovalnimi podatkovnimi zapisi (povezava tipa 1:N), prav tako pa je tudi vsak projekt lahko povezan z več povezovalnimi podatkovnimi zapisi (povezava tipa N:1).

S tem, ko je bilo omogočeno predstaviti tudi povezave tipa N:M brez podvajanja podatkov, pa je postal mrežni model precej zapleten za razumevanje in uporabo.

Danes se mrežni model (za razvoj novih podatkovnih baz) v praksi ne uporablja več. Tudi zato ne, ker ga je nadomestil sodobnejši objektni podatkovni model (glej

nadaljevanje), ki je glede na dovoljeno strukturo podatkov pravzaprav naslednik mrežnega modela, ima pa pred njim nekatere prednosti.

Mrežni modeli so dali pečat razvoju podatkovnih baz v 1970. letih. Večina SUPB se je držala priporočil standarda za mrežne podatkovne baze CODASYL (Conference on Data System Language) iz leta 1971. Med mrežnimi SUPB omenimo le IBM-ov IDMS.

Povzetek značilnosti

- Mrežni model omogoča predstavitev povezav tipa M:N brez podvajanja podatkov.
- Je precej zapleten za razumevanje in uporabo.
- Model je bil standardiziran.
- Danes ni več pogosto v uporabi.

Relacijski model podatkov

Relacijski podatkovni model temelji na matematični teoriji relacij[22]. Podatki so potemtakem organizirani v relacijske podatkovne strukture, ki pa so fizično predstavljene z dvodimenzionalnimi tabelami. Ker je razumevanje tako lažje, bomo v nadaljevanju razmišljali o relacijah kot o tabelah, oba pojma pa bomo uporabljali kot sopomenki, čeprav to ni popolnoma korektno.

Praviloma hranimo v eni tabeli podatke o vseh entitetah nekega entitetnega tipa ali o vseh povezavah nekega tipa povezav. Vsaka **vrstica** predstavlja eno entiteto ali povezavo; v njej so shranjeni podatki o eni entiteti (povezavi). Vsak **stolpec** pa predstavlja en **atribut**, v njem so shranjene vrednosti tega atributa za vse entitete (povezave) tega tipa. Na sliki 30 je prikazan primer relacije IZDELEK; v tabeli so shranjeni vsi podatki o izdelkih. Vsaka vrstica predstavlja en izdelek, vsak stolpec pa en atribut izdelkov.

---

22 Zahtevnejši bralec bo za spoznavanje ali osvežitev spomina o relacijah posegel po drugi literaturi, ki je temu namenjena, na primer, (Prijatelj, 1980).

RELACIJA **IZDELEK**

ATRIBUTI

IME RELACIJE

	IZDELEK#	**NAZIVIZD**	**ME**	**CENA**
PODATKI O ENTITETAH (IZDELKIH)	123-456	Sol morska	kg	49,00
	321-432	Kisle kumarice 500 g	kos	192,00
	753-443	Sir Gauda	kg	1285,00
	325-775	Jabolčni sok – šesterček	paket	480,00

Slika 30: Tabela/relacija IZDELEK

Pozoren bralec bo opazil, da tudi v relacijskem modelu glavni ključ označimo tako, da njegovo ime podčrtamo.

Stolpcu relacije pravimo tudi **polje** (angl. field), vrstici pa **podatkovni zapis** (angl. record). Ta terminologija se večinoma uporablja v sistemih za upravljanje relacijskih podatkovnih baz.

Povzemimo zdaj značilnosti relacije in dodajmo še nekatere:

1. Vsak posamezni stolpec v relaciji vsebuje vrednosti nekega atributa za vse entitete obravnavanega entitetnega tipa, ki ga relacija predstavlja.
2. Vrednost atributa neke entitete mora biti v relacijskem modelu elementarni podatek. To pomeni, da ne more biti sestavljen atribut ali večvrednostni atribut.
3. Vsak stolpec oziroma atribut je poimenovan z imenom, ki se razlikuje od imen drugih stolpcev oziroma atributov.
4. Vrstni red (poimenovanih) stolpcev je nepomemben.
5. Zaporedje vrstic v relaciji je nepomembno in ga praviloma ne poznamo.

Glavni ključ relacije

V primeru, ko relacija predstavlja entitetni tip, glavni ključ tega entitetnega tipa postane v relacijskem modelu **glavni ključ** pripadajoče **relacije**.

Vemo, da glavni ključ med seboj razlikuje entitete nekega tipa in ne moreta imeti dve entiteti iste vrednosti tega atributa. Ker v relacijskem modelu vrstica pripadajoče relacije opisuje eno entiteto, posledično *ne moreta imeti dve vrstici relacije (dva podatkovna zapisa) iste vrednosti glavnega ključa.* Iz tega pa sledi tudi to, da *vsaki vrednosti ključnega atributa pripada največ ena vrednost vseh ostalih atributov*, saj v relacijskem modelu velja, da v eni vrstici nek atribut ne more imeti več vrednosti. Z drugimi besedami, *vsi nekljčni atributi so funkcionalno odvisni od glavnega ključa.*

To je pravzaprav tudi formalna definicija kandidata za ključ v relacijskem podatkovnem modelu, ki pravi: Atribut ali skupina atributov A je kandidat za ključ relacije R natanko takrat, ko eni vrednosti A pripada v vsakem trenutku največ ena vrednost vsakega drugega atributa iz R-A (atributi, ki so v R in niso v A).

Razjasnimo zgornji razmislek še na primeru.

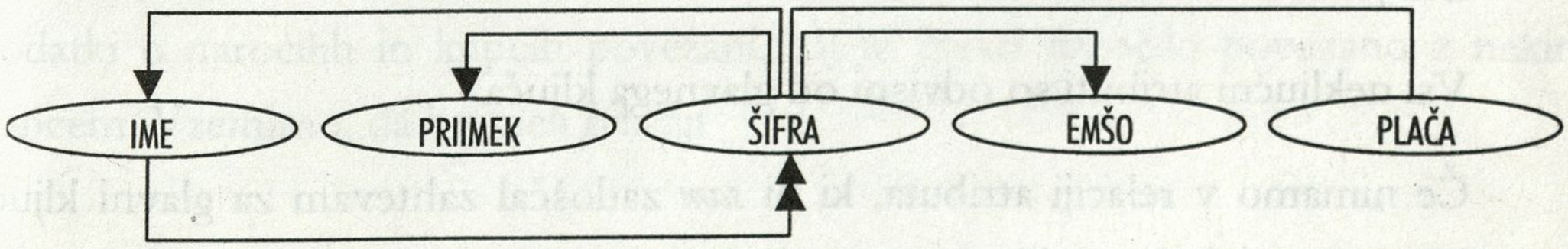
V podatkovni bazi želimo hraniti podatke o delavcih. Entitetni tip DELAVCI vsebuje vse delavce, katerih atributi, ki nas zanimajo so: IME, PRIIMEK; ŠIFRA, EMŠO, PLAČA. Očitno sta kandidata z ključ atributa EMŠO in ŠIFRA, med njima pa smo izbrali za glavni ključ atribut ŠIFRA.

V relacijskem modelu naredimo ustrezno relacijo DELAVCI, kjer vsaka vrstica vsebuje podatke o enem delavcu.

IME	PRIIMEK	ŠIFRA	EMŠO	PLAČA
Miha	Kovač	1234	1111111	90.000
Ana	Medved	2345	2222222	120.000
Ana	Kovač	3213	3333333	90.000

Ker vsaka vrstica predstavlja podatke o enem delavcu, je jasno, da ne moremo imeti dveh vrstic z isto šifro ali EMŠO, lahko pa imamo več vrstic z istim imenom, priimkom ali plačo. Ker imamo le eno vrstico z neko šifro (npr. 1234), lahko tej šifri pripada le ena vrednost drugih atributov: IME, PRIIMEK, EMŠO in PLAČA. Tako šifri 1234 pripada le ime Miha, priimek Kovač, EMŠO 1111111 in plača 90.000. Ne velja pa obratno, saj, na primer, imenu Ana pripadata dve šifri: 2345 ter 3213.

Prikažimo to odvisnost neključnih atributov od glavnega ključa še grafično.



Na sliki so prikazane odvisnosti neključnih atributov od ključnega in še primer povezave v nasprotno smer med imenom in šifro, ki kaže, da šifra ni odvisna od imena, kar pomeni, da ime ne more biti kandidat za ključ te relacije.

Pogosto pa v relacijah en sam atribut ne zadošča zahtevi za glavni ključ relacije. Praviloma to velja za relacije, ki predstavljajo tipe povezav med entitetnimi tipi. V tem primeru moramo za glavni ključ izbrati več atributov, za katere velja, da se vsaka kombinacija njihovih vrednosti pojavi samo v eni vrstici. Pravimo, da ima relacija **sestavljen glavni ključ**.

Vzemimo za primer povezave med naročili in izdelki. Če je izdelek s šifro 123-456 povezan z naročilom številka 423, pomeni, da je kupec z naročilom številka 423, morda poleg drugih izdelkov, naročil tudi izdelek s šifro 123-456. Tovrstne povezave predstavimo z relacijo POSTAVKA, na primer,

IZDELEK#	NAROČILO#	KOLIČINA
123-456	423	10
321-432	423	1
123-456	424	8
321-432	425	26
325-775	426	1
321-432	426	33
753-443	426	123

Povzemimo torej najpomembnejše ugotovitve o ključih relacij:

- Relacija ima lahko več kandidatov za ključ, od katerih izberemo enega za glavni ključ, ostali pa so nadomestni ključi.
- Glavni ključ relacije ne more imeti v dveh vrsticah iste vrednosti.
- Glavni ključ označimo tako, da ga podčrtamo. Ostalim atributom pravimo neključni atributi.
- Vsi neključni atributi so odvisni od glavnega ključa.
- Če nimamo v relaciji atributa, ki bi *sam* zadoščal zahtevam za glavni ključ, izberemo za glavni ključ ustrezno kombinacijo atributov, ki ji pravimo sestavljen glavni ključ.

Na koncu poudarimo izjemen pomen, ki ga ima pravilna izbira ključev v relacijah za kasnejšo uporabo podatkovne baze. Slaba izbira ključa lahko, na primer, pomeni, da ne bomo dobili pravih odgovorov na vprašanja (poizvedbe).

Relacijska shema

Pri prikazu relacij ponavadi ne prikazujemo celotne tabele z vsemi podatki, saj nas pogosto zanima le struktura relacije (posebej pri načrtovanju), poleg tega pa se podatki, ki odražajo realnost, pogosto spreminjajo, pri čemer pa struktura relacije ostaja ista.

Struktura relacije je dobro opredeljena s svojim imenom in seznamom atributov, kar prikažemo tako, da za imenom relacije v okroglem oklepaju naštejemo atribute relacije, glavni ključ pa podčrtamo. Na primer,

IZDELEK (IZDELEK#, NAZIVIZD, ME, CENA)

Tak zapis imenujemo relacijska shema. Relacijska shema podatkovne baze bo praviloma vsebovala več takih shem za posamezne relacije, na primer

NAROČILO (NAROČILO#, DATUM, KUPEC#, TP#, POPUST)
IZDELEK (IZDELEK#, NAZIVIZD, ME, CENA)
POSTAVKA (IZDELEK#, NAROČILO#, KOLIČINA)
KUPEC (KUPEC#, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT)
TRG-POTNIK (TP#, IMETP, PRIIMEKTP, REGIJA)

Relacijska shema podatkovne baze je hkrati njen načrt[23], saj opredeljuje, kako naj strukturiramo podatke: tabele, njihove atribute, glavne ključe …

Povezave

Podatki iz dveh ali več relacij so lahko med seboj povezani. Tako so, na primer, podatki o naročilih in kupcih povezani, saj je vsako naročilo povezano z nekim kupcem. Vzemimo, da bi imeli relaciji

---

23 Tak načrt za tvorbo podatkovne baze še ne zadostuje. V nadaljevanju izdelamo še t.i. fizični načrt, v katerem natančneje opredelimo fizične značilnosti, na primer, tipe in velikosti posameznih atributov (tekstovni podatek dolžine 20 znakov, celo število, realno število, denarni znesek …)

NAROČILO#	DATUM	TP#	POPUST
423	9.7.1999	1	NULL
424	9.7.1999	2	10
425	12.7.1999	1	0
426	12.7.1999	4	3
429	13.7.1999	4	10

in

KUPEC#	NAZIVKUP	NASLOVKUP	TELEFON	KONTAKT
12	Atlas d.d.	Mirna 2, Lj	(061)11-22	Janez
34	Biro s.p.	Živa 39	632-33	Ana Mohorko
24	Credo d.o.o.	Pot 12, Mb	(062)33-11	NULL
13	Janez Aleš	Borbena 223	332-11	Peter, Miro
198	Argonavti	Čista 92a	112-112	Sonja Kovač

ki opisujeta entitete tipov NAROČILO in KUPEC. Iz tako strukturiranih podatkov ne bi mogli nikakor ugotoviti, čigavo je posamezno naročilo.

V relacijskem modelu podatke iz dveh ali več relacij vsebinsko povezujemo s skupnimi atributi.

Če želimo v zgornjem primeru povedati, da gre za naročilo nekega kupca, bomo to storili tako, da v relacijo NAROČILO dodamo atribut KUPEC# in dobimo

NAROČILO (NAROČILO#, DATUM, TP#, POPUST, KUPEC#)
KUPEC (KUPEC#, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT)

Skupni atributi, ki služijo za povezavo dveh (ali več) relacij so tisto nujno podvajanje podatkov (npr. imamo dvakrat atribut KUPEC#), ki ga v relacijskih podatkovnih bazah dopuščamo.

Poglejmo zdaj, kako lahko prikažemo povezave med entitetami v relacijskem modelu. Možna sta dva načina.

1. Kadar med entitetnima tipoma obstaja povezava 1:1 ali 1:N, potem lahko povezavo med ustreznima relacijama vzpostavimo tako, kot smo to storili v zgornjem primeru povezave med relacijama NAROČILO in KUPEC. V tem primeru v relacijo na strani N dodamo glavni ključ relacije na strani 1. V konkretnem primeru je na strani N relacija NAROČILO, saj je vsak kupec lahko povezan z več naročili, eno naročilo pa samo z enim kupcem. Zakaj

dodamo atribut prav v relacijo na strani N, naj razmisli bralec sam. Kadar imamo povezavo 1:1, pa je izbira, kam damo atribut, načeloma poljubna, pa vendar je v večini primerov ena od izbir boljša kot druga[24].

NAROČILO (NAROČILO#, DATUM, TP#, POPUST, KUPEC#)
KUPEC (KUPEC#, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT)

2. Kadar med entitetnima tipoma obstaja povezava M:N, potem tvorimo novo relacijo, ki predstavlja prav povezavo med tema entitetnima tipoma. Glavni ključ te relacije bo sestavljen iz glavnih ključev relacij, ki ju povezuje. Na primer, povezava med entitetnima tipoma NAROČILO in IZDELEK je tipa M:N, saj je lahko na vsakem naročilu več izdelkov, en izdelek pa se lahko pojavi na več naročilih. V tem primeru ne moremo dodati glavnega ključa ene relacije v drugo, saj bi moral biti to večvrednostni atribut, ki pa v relacijskem modelu ni dopusten. Zato smo tvorili relacijo POSTAVKA, ki predstavlja povezavo med naročili in izdelki.

NAROČILO (NAROČILO#, DATUM, KUPEC#, TP#, POPUST)
IZDELEK (IZDELEK#, NAZIVIZD, ME, CENA)
POSTAVKA (IZDELEK#, NAROČILO#, KOLIČINA)

V zvezi s prvim načinom povezave med relacijami lahko definiramo pojem tuji ključ. **Tuji ključ** je neključni atribut ene relacije, ki je v drugi relaciji glavni ključ. Označimo ga tako, da ga podčrtamo s prekinjeno črto. Na primer, v relaciji NAROČILO je šifra kupca (KUPEC#) tuji ključ, saj je tu neključni atribut, v relaciji KUPEC pa je isti atribut glavni ključ.

Povzemimo: *Relacije* so med seboj povezane *s skupnimi atributi. Povezave med entitetami* lahko predstavimo s *tujimi ključi* ali z relacijo, ki ima *sestavljen glavni ključ.*

Poglejmo to na primeru naročanja izdelkov:

NAROČILO (NAROČILO#, DATUM, KUPEC#, TP#, POPUST)
IZDELEK (IZDELEK#, NAZIVIZD, ME, CENA)
POSTAVKA (IZDELEK#, NAROČILO#, KOLIČINA)

---

24 Obravnavo te teme najde zahtevnejši bralec v vsaki boljši knjigi o podatkovnih bazah.

KUPEC (KUPEC#, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT)
TRG-POTNIK (TP#, IMETP, PRIIMEKTP, REGIJA)

V tem primeru so relacije povezane takole:

relaciji	skupni atribut
NAROČILO in POSTAVKA	NAROČILO#
POSTAVKA in IZDELEK	IZDELEK#
NAROČILO in KUPEC	KUPEC#
NAROČILO in TRG-POTNIK	TP#

Relacije NAROČILO, IZDELEK, KUPEC in TRG-POTNIK predstavljajo entitete, ki so med seboj povezane takole:

entiteti	povezava predstavljena
NAROČILO in IZDELEK	povezovalna relacija POSTAVKA
NAROČILO in KUPEC	tuji ključ KUPEC# v relaciji NAROČILO
NAROČILO in TRG-POTNIK	tuji ključ TP# v relaciji NAROČILO

Relacijska podatkovna baza

Relacijska podatkovna baza je taka baza podatkov, v kateri so podatki strukturirani v skladu z relacijskim podatkovnim modelom. Z drugimi besedami, relacijska podatkovna baza je množica med seboj povezanih relacij oziroma tabel. Njeno strukturo lahko prikažemo z relacijsko shemo podatkovne baze.

Značilnosti relacijskega podatkovnega modela

Relacijske podatkovne baze so v 1980. letih postale izjemno razširjene in še danes je to v poslovnih informacijskih sistemih najpogosteje uporabljen podatkovni model. Nekatere izmed naslednjih značilnosti relacijskega podatkovnega modela in sistemov za upravljanje relacijskih podatkovnih baz so k temu gotovo prispevali:

- Relacijski podatkovni model je preprost za razumevanje.
- Kljub preprostosti pa temelji na čvrsti matematični podlagi, to je teorija relacij.

- Dejanska fizična predstavitev je uporabniku skrita. Uporabnik podatke vidi, kot da bi bili shranjeni v dvodimenzionalnih tabelah, čeprav je lahko dejanska predstavitev zaradi učinkovitosti bistveno bolj zapletena.
- Poizvedovanje je praviloma počasnejše kot pri uporabi ostalih modelov, saj so povezave med podatki implicitne (skupni atributi), pri drugih modelih pa so eksplicitne.
- Za poizvedovanje se uporablja standardni poizvedovalni jezik SQL (glej poglavje Uporaba podatkovnih baz), kar omogoča enostaven prehod na uporabo drugega sistema za upravljanje podatkovnih baz.

Sistemi za upravljanje relacijskih podatkovnih baz

Z razvojem teorije relacijskega modela so se relacijske podatkovne baze zaradi enostavnosti izredno razširile v praksi. Tako lahko danes izbiramo med večjim številom relacijskih SUPB, med katerimi velja omeniti: Oracle, MS SQL, DB2 (IBM), Sybase in Informix. Večina jih je na voljo za različne računalniške strojne in operacijske sisteme, od velikih (angl. mainframe) računalnikov do osebnih računalnikov. Uporaba omenjenih SUPB temelji na arhitekturi odjemalec/strežnik, kjer je SUPB podatkovni strežnik, programi, ki dostopajo do podatkov pa so odjemalci.

Poleg tega je v zadnjih letih na tržišču tudi večje število programskih paketov za delo z (relacijskimi) podatkovnimi bazami na osebnih računalnikih[25]. Prvi, ki je bil tržno uspešen in še danes velja za standard podatkovnih baz na osebnih računalnikih, je dBase. Čeprav so imeli ti paketi na začetku le malo tistih funkcij, ki so jih poznali strežniški SUPB, pa so danes že precej izpopolnjeni. Večinoma so več kot le SUPB, saj omogočajo tudi razvoj programov, ki dostopajo do podatkovnih baz. Skoraj vsi omogočajo tudi povezavo s podatkovnimi bazami na velikih sistemih. Ker so tudi prijazni do uporabnika (okensko okolje), jih mnogokrat uporabljamo kot vmesnik za strežniške SUPB, ki so praviloma manj

25 V tem, primeru se SUPB ne obnašajo kot strežniki.

uporabniško prijazni. Najbolj znani primeri paketov za osebne računalnike in okolje oken so: MS Access, dBase in FoxPro.

Objektni model

Čeprav segajo začetki objektnega pristopa že v konec 1960. let, pa se je v podatkovnih bazah uveljavil šele v zgodnjih 1990. letih.

Poglavitna ideja objektnega modela je, da naj bo *podatkovna baza čimbolj verna slika našega videnja realnosti.* Ker realni svet vidimo kot množico objektov s svojimi *značilnostmi* in *obnašanjem,* naj bodo ti objekti tudi v podatkovni bazi opisani z značilnostmi (lastnostmi, podatki) – **atributi** in obnašanjem – **metodami**. Poleg podatkov o objektih so torej v podatkovnih bazah shranjeni tudi programi, ki opisujejo obnašanje objektov.

Poleg atributov, ki opisujejo študente, na primer, vpisna številka, ime, priimek ..., nas pri objektnem pristopu zanima tudi njihova aktivna vloga, na primer, prijava na izpit, opravljanje izpita, prijava diplomskega dela ...

Ker so v bazi shranjeni poleg podatkov tudi programi, se pri objektnem pristopu pomembno spremeni način razvoja programske opreme, saj so metode hkrati že deli le-te. Zaradi istega razloga tudi uporabljamo izraz objektni model podatkov *in procesov,* hkrati pa še vedno (zdaj neprimeren) izraz objektne podatkovne baze.

Kljub smiselnim predpostavkam, iz katerih izhaja objektni pristop, pa se objektne podatkovne baze[26] v poslovnih informacijskih sistemih do danes niso uveljavile v večji meri. Poglavitni razlogi so:

- Ker je namen objektnega modela čimbolj natančen opis realnosti, gre to seveda nujno na škodo enostavnosti in razumljivosti. Zato je sam objektni model bistveno bolj zapleten za razumevanje kot relacijski podatkovni model, hkrati pa je načrtovanje objektne podatkovne baze bistveno bolj zahtevno kot načrtovanje relacijske podatkovne baze.

---

26 Mišljene so podatkovne baze, ki popolnoma upoštevajo zahteve objektnega modela.

- Prednosti objektnega pristopa pridejo mnogo bolj do izraza na nekaterih drugih področjih (programiranje, CAD/CAM sistemi, večpredstavni sistemi) kot v poslovnih informacijskih sistemih.
- V začetnem razvoju ni bilo enega standardnega objektnega modela, temveč je vsak proizvajalec sistemov za upravljanje objektnih podatkovnih baz uporabljal svoj model. Danes je stanje na tem področju boljše, saj sistemi za upravljanje objektnih podatkovnih baz, npr. Poet, Objectstore, Versant, Objectivity, Gemstone ..., temeljijo na skupnem standardu skupine Object Database Management Group (ODBMG).

Ker pa so nekateri elementi objektnega pristopa zelo uporabni tudi v poslovnih podatkovnih bazah, v zadnjem času tudi sistemi za upravljanje relacijskih podatkovnih baz (Oracle, IBM DB2 ...) vsebujejo čedalje več teh elementov. Na ta način so združeni enostavnost in razumljivost relacijskih podatkovnih baz ter nekatere prednosti objektnega pristopa.

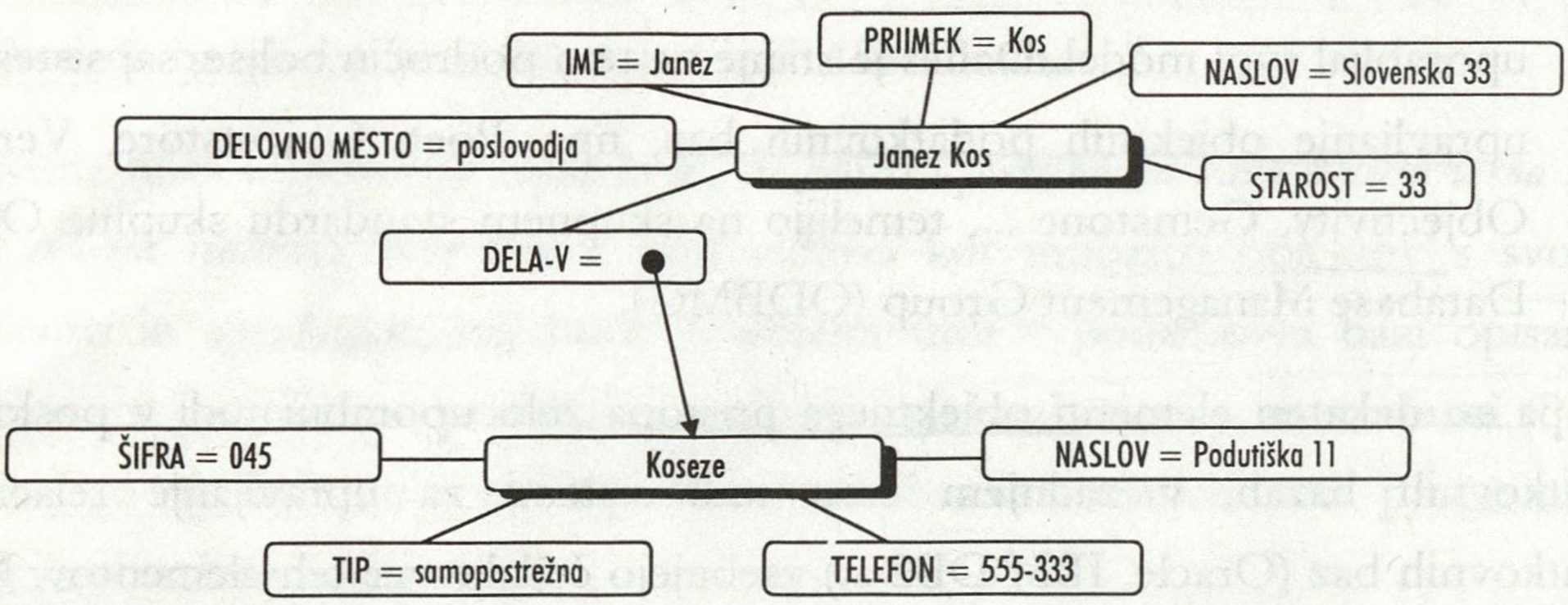
Zaradi kompleksnosti objektnega modela v nadaljevanju navajamo le njegove najpomembnejše značilnosti, za zahtevnejše bralce pa še nekaj osnovnih pojmov. Za podrobnejši opis objektnega modela glej (Grad, Jaklič, 1996).

Povzetek značilnosti

- Objektni model združuje atribute objektov in njihovo obnašanje.
- Med objekti so dovoljene povezave vseh tipov (1:1, 1:N in N:M).
- Dovoljeni so (za razliko od relacijskega modela) tudi večvrednostni atributi.
- Poizvedovanje je praviloma hitrejše kot pri relacijskih podatkovnih bazah.
- Istovrstne objekte združujemo v razrede.
- Objekt ustreza entiteti v modelu entitet povezav, razred pa entitetnemu tipu.
- Razvoj informacijskega sistema, ki temelji na objektnem modelu, je bolj zahteven, če primerjamo z razvojem informacijskega sistema, ki temelji na relacijski podatkovni bazi. Hkrati pa vzdrževanje bolj enostavno, sistem pa bolj robusten.

Povezave med objekti so lahko tipov 1:1, 1:N ali M:N. Realizirane so z referenčnimi atributi, ki jih imenujemo tudi mehki kazalci. Mehki kazalci so po svojem značaju nekje med kazalci, ki jih poznamo iz hierarhičnega in mrežnega modela, in med tujimi ključi, ki jih za realizacijo povezav uporabljamo v relacijskem modelu.

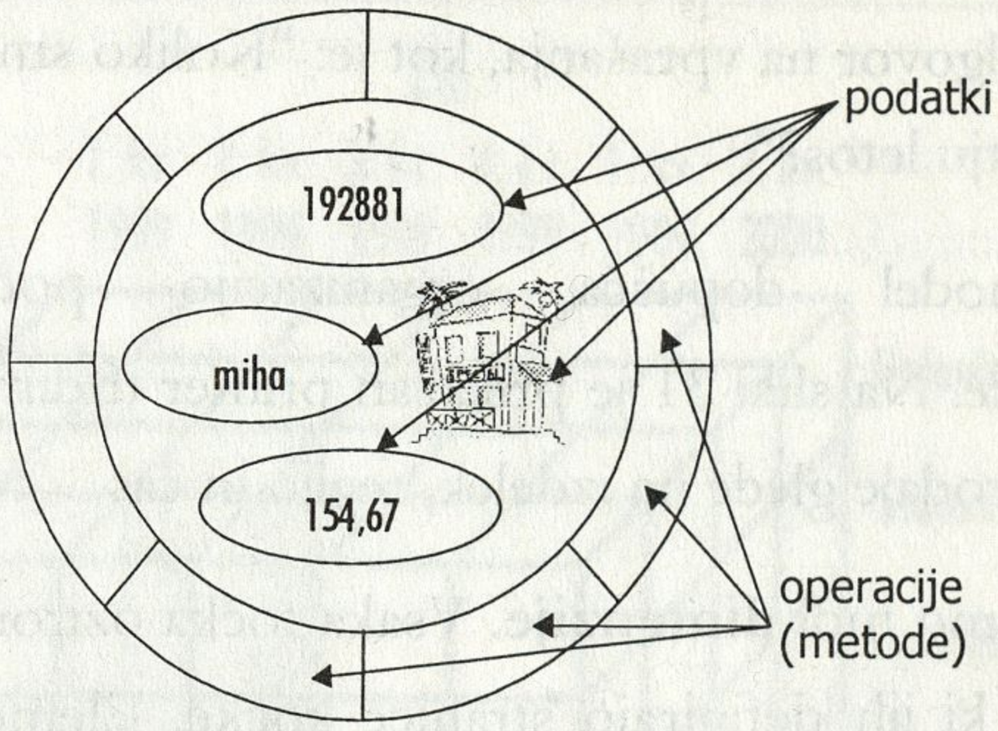
Na primer, trgovsko podjetje hrani podatke o svojih trgovinah in zaposlenih. V ta namen uporablja dva razreda objektov ZAPOSLENI in PRODAJALNA.



Na sliki sta prikazana dva objekta: Janez Kos je objekt tipa ZAPOSLENI, Koseze pa je objekt tipa PRODAJALNA. Poleg ostalih atributov imajo objekti tipa ZAPOSLENI tudi referenčni atribut DELA-V, katerega vrednost je mehki kazalec na objekt tipa PRODAJALNA. Uporabnik vrednosti tega atributa ne vidi. Če uporabnika zanima, v kateri prodajalni dela Janez Kos, bo kot odgovor dobil objekt Koseze in ne vrednosti nekega enostavnega atributa kot pri relacijskem modelu (npr. šifre prodajalne 045).

Iz tipov lahko z generalizacijo (oziroma v obratni smeri specializacijo) dobimo nove tipe. Med takima tipoma potem obstaja povezava, ki jo največkrat imenujemo ISA (je). Tip OSEBA ima lahko podtipa ZAPOSLENI in KUPEC. Iz tipa ZAPOSLENI pa lahko s specializacijo dobimo podtipa ADMINISTRATOR in PRODAJALEC. Na ta način dobimo hierarhijo tipov. Nadtip je bolj splošen od svojega podtipa. Podtip deduje podatkovno strukturo in metode svojega starša – nadtipa. Z drugimi besedami, podtip vsebuje vse atribute in vse metode nadtipa. Poleg tistega, kar podeduje, pa ima lahko podtip tudi svoje atribute in metode. Tako ima objekt tipa OSEBA atribute IME, PRIIMEK, NASLOV, TELEFON. Objekt iz podrazreda ZAPOSLENI ima iste atribute, saj glede na hierarhijo razredov sodi tudi v razred OSEBA, poleg teh pa ima še dodatne atribute PLAČA in DELOVNO MESTO. Hierarhija razredov omogoča dedovanje, ki pomembno poenostavi in skrajša čas razvoja informacijskega sistema.

Pomembna lastnost objektnega modela je tudi ograjevanje (enkapsulacija), ki pomeni, da uporabniki (in drugi objekti) ne vidijo implementacije objekta. Ograjevanje skriva podrobnosti o objektu pred uporabnikom. To pomeni, da ne moremo neposredno dostopati do vrednosti atributov in ne vemo, kako so kodirane metode – ne poznamo programov (postopkov). Vse, kar želimo vedeti o objektu, lahko ugotovimo le tako, da mu pošljemo zahtevo, objekt izvede ustrezno metodo, in nam posreduje odgovor. To lahko primerjamo tudi z objekti v realnem svetu. Če želimo vedeti ime neke osebe, ji to ponavadi "ne piše na čelu", temveč jo moramo po imenu vprašati. Prodajalci v trgovinah torej kršijo pravilo ograjevanja, ker nosijo značke s svojimi imeni. Z ograjevanjem je omogočena večja prilagodljivost pri spreminjanju obnašanja objektov. Na primer, program, ki potrebuje izračun dohodnine za državljana in v ta namen uporablja operacijo (metodo), ki pripada temu državljanu – objektu, se ne spremeni, če se spremeni način obračunavanja dohodnine. Spremeni se le metoda – operacija objekta.



Večdimenzionalni model

Čeprav je relacijski model sam po sebi enostaven za razumevanje, pa so sheme relacijskih podatkovnih baz že za relativno "majhna" problemska področja presenetljivo kompleksne. Pogosto grafični prikaz sheme pokriva celo steno sobe načrtovalca. To kompleksnost za uporabnike na operativnem nivoju skrijejo programi, s katerimi delajo, kar pomeni, da ti uporabniki sheme sploh ne poznajo oziroma ne rabijo poznati.

Popolnoma drugačno pa je stanje na področju podpore odločanja, kjer je delo s podatkovnimi viri, kot smo že ugotovili, precej bolj nerutinsko. Poizvedbe po podatkovnih virih so zelo raznolike; za ad hoc poizvedbe, za katere nimamo vnaprej pripravljenih programov, pa je nujno poznavanje sheme podatkovnega vira. Zato je razumljiva ugotovitev, da relacijske podatkovne baze, vsaj v obliki, kot jih uporabljamo na operativnem nivoju, niso primerne za podatkovne vire, namenjene poslovnemu odločanju.

Drugi pomemben razlog za drugačno organizacijo podatkov, kot je v navadi pri relacijskih podatkovnih bazah, je počasnost poizvedovanja po več tabelah, ki posebej pride do izraza pri delu analitikov, ki naenkrat delajo z ogromnimi količinami podatkov.

Zaradi teh razlogov se v podatkovnih virih, ki so namenjeni podpori odločanju (npr. podatkovna skladišča in področna podatkovna skladišča), v uporabi večdimenzionalni podatkovni model.

Njegova bistvena značilnost je, da so sheme, ki jih lahko realiziramo, preproste in blizu načinu razmišljanja analitika/odločevalca. V večdimenzionalni podatkovni

bazi zlahka najdemo odgovor na vprašanja, kot je: "Koliko smo prodali izdelka I v regiji X v prvem četrtletju letos?"

Večdimenzionalni model dopušča organizacijo podatkov v obliki večdimenzionalne kocke. Na sliki 31 je prikazan primer tridimenzionalne[27] kocke, ki prikazuje vrednost prodaje glede na izdelek, regijo in čas.

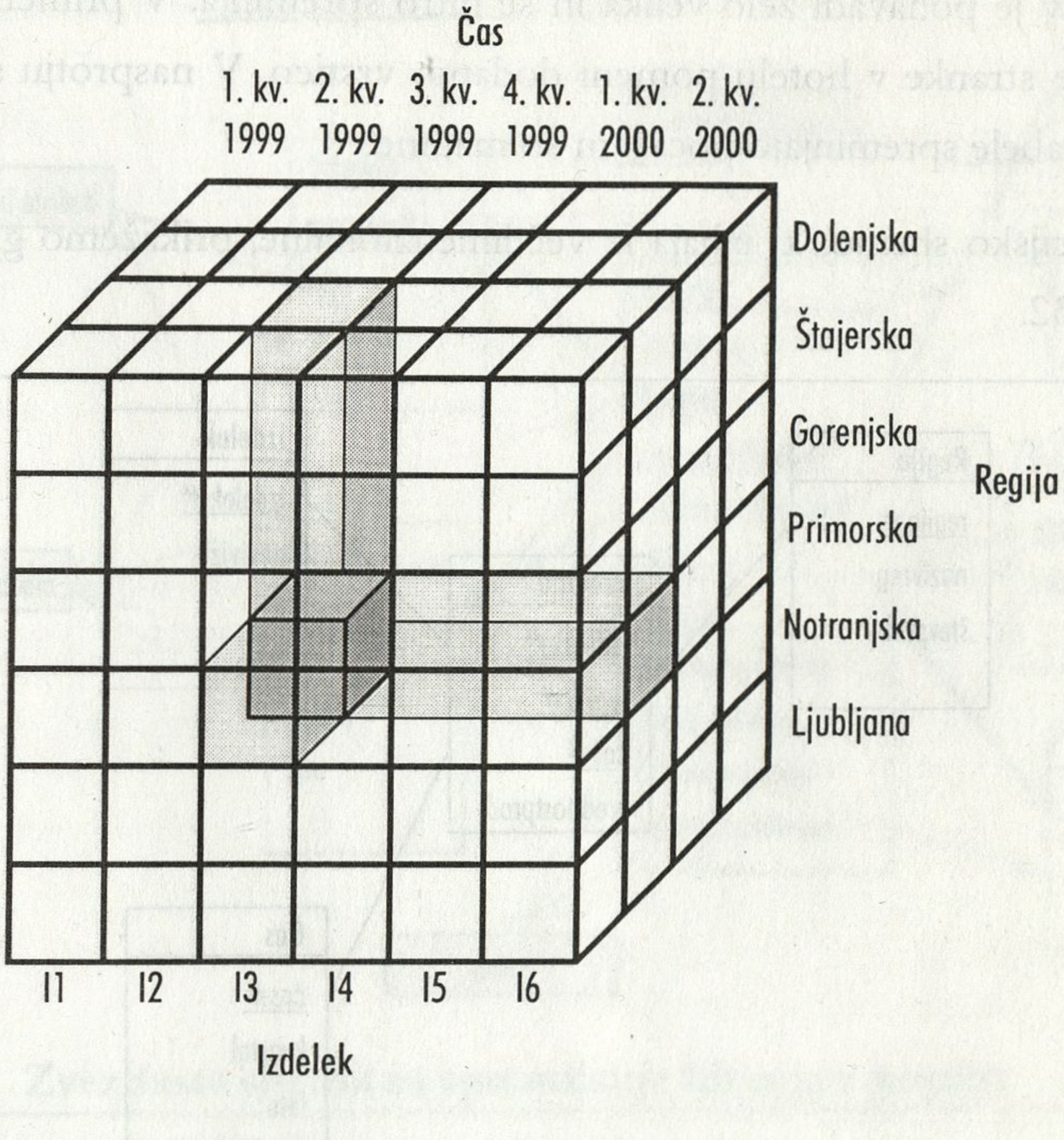
Stranice kocke imenujemo tudi **dimenzije**. Vsaka točka oziroma element kocke je na preseku koordinat, ki jih definirajo stranice kocke. Elementi kocke so **mere** poslovanja glede na dimenzije. V primeru s slike 31 je *mera* vrednost prodaje za neko kombinacijo *dimenzij* izdelek, regija in čas. Na primer, temneje označen element kocke (manjša temna kocka) je vrednost prodaje izdelka I3 na Primorskem v tretjem kvartalu leta 1999.

Naj posebej opozorimo na dimenzijo *čas*, ki je značilna za podatkovna skladišča in področna podatkovna skladišča, redkeje pa jo srečamo v operativnih podatkovnih bazah, kjer nas zanimajo predvsem trenutne vrednosti podatkov.

Za realizacijo večdimenzionalne sheme lahko uporabimo sisteme za upravljanje večdimenzionalnih podatkovnih baz, zelo pogosto pa večdimenzionalno shemo realiziramo kar z relacijsko podatkovno bazo.

---

27 Več dimenzij kot tri je težko grafično prikazati. V nadaljevanju pa bomo prikazali na nekoliko drugačen način tudi petdimenzionalno shemo.



Slika 31: Primer tridimenzionalne kocke

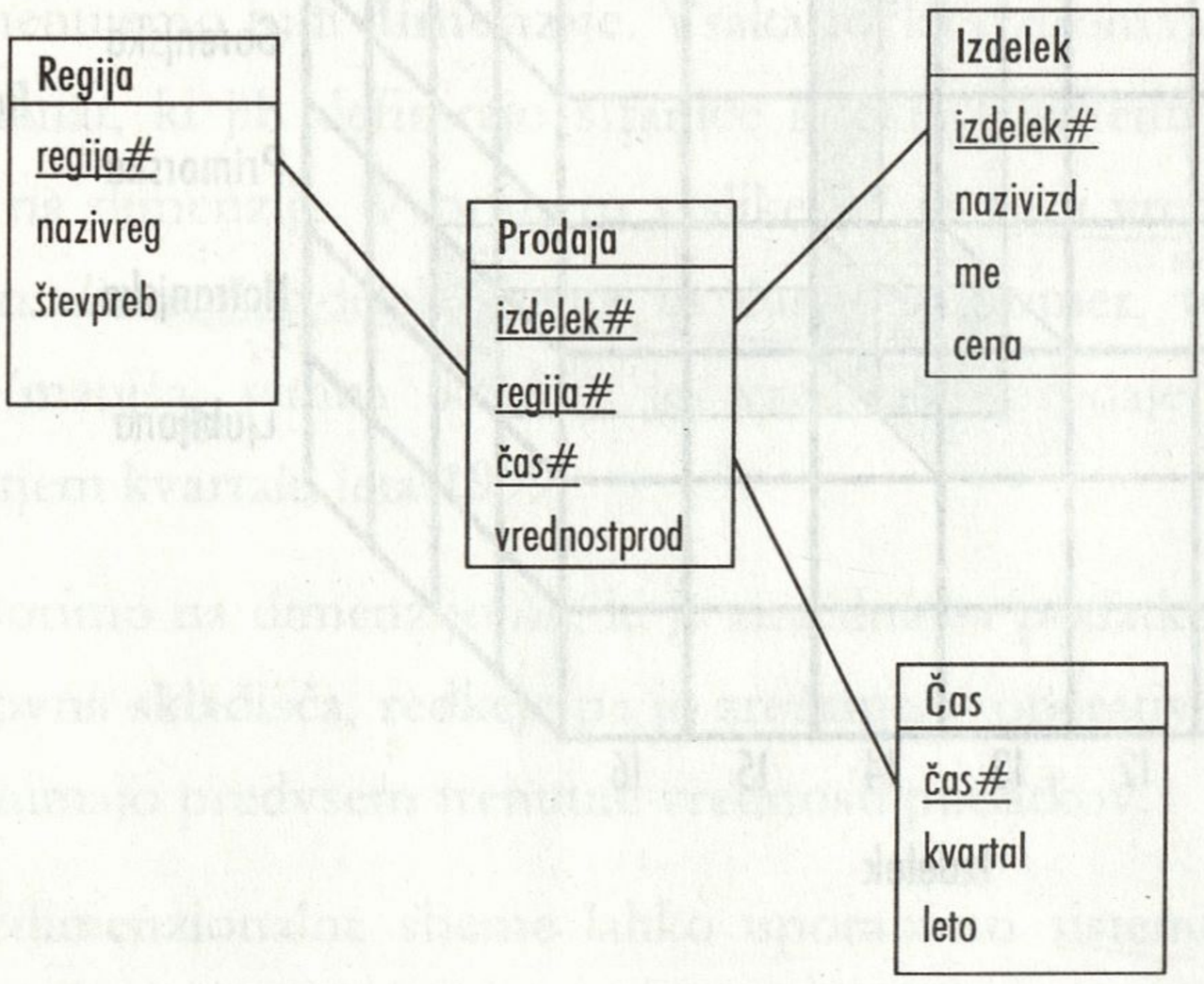
Tako lahko večdimenzionalno shemo s slike 31 enostavno pretvorimo v relacijsko shemo:

IZDELEK (IZDELEK#, NAZIVIZD, ME, CENA)
REGIJA (REGIJA#, NAZIVREG, ŠTEVPREB)
ČAS (ČAS#, KVARTAL, LETO)
PRODAJA (IZDELEK#, REGIJA#, ČAS#, VREDNOSTPROD)

Očitno je, da vsaki dimenziji pripada ena tabela, ki jo imenujemo **dimenzijska tabela** (angl. dimension table). Tabelo, v kateri shranjujemo podatke za vrednosti mer pri posamezni kombinaciji vrednosti dimenzijskih atributov, imenujemo **tabela dejstev** (angl. fact table). Tabela dejstev povezuje dimenzijske tabele, zato je njen ključ sestavljen iz glavnih ključev dimenzijskih tabel. Praviloma vsebuje več mer. Tako bi lahko v zgornjem primeru hranili še podatke o prodani količini, dobičku in podobno.

Tabela dejstev je ponavadi zelo velika in se hitro spreminja. V primeru s slike 33 vsako bivanje stranke v hotelu pomeni dodatno vrstico. V nasprotju s tem pa se dimenzijske tabele spreminjajo počasi in so majhne.

Pogosto relacijsko shemo, ki izhaja iz večdimenzionalne, prikažemo grafično, kot je to na sliki 32.

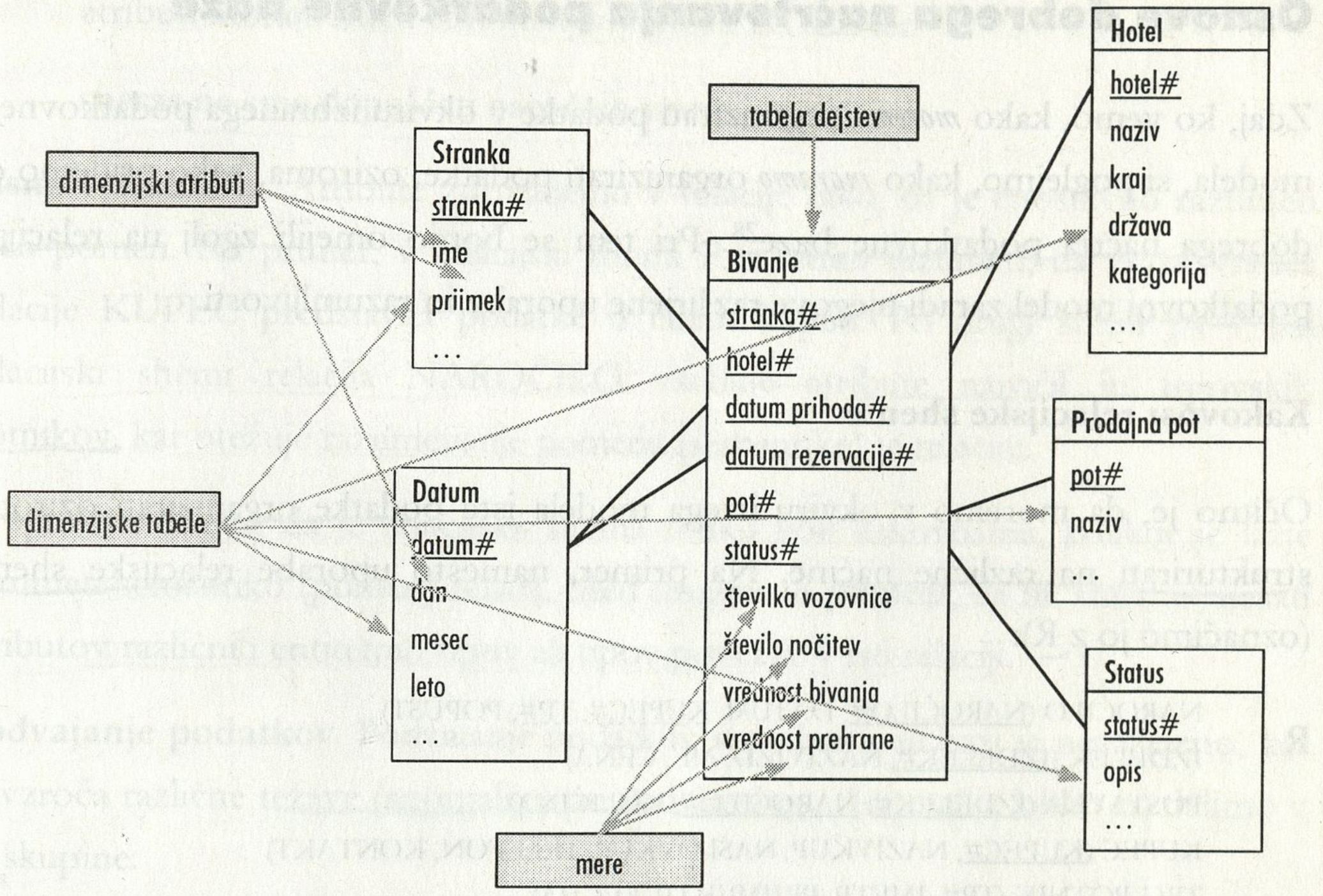


Slika 32: Grafični prikaz relacijske sheme za primer s slike 31

Zaradi specifične oblike govorimo o **zvezdasti shemi**.

Oglejmo si še en primer zvezdaste sheme, tokrat za turistično poslovanje, natančneje za spremljanje bivanja v hotelih.

Nekateri sistemi za upravljanje relacijskih podatkovnih baz so posebej optimizirani tudi za delo z zvezdastimi podatkovnimi shemami in s tem bolj primerni za izgradnjo podatkovnega skladišča ali področnega podatkovnega skladišča.



Slika 33: Zvezdasta shema za spremljanje bivanja v hotelih

Povzetek značilnosti

- Večdimenzionalni podatkovni model je preprost za razumevanje, preproste pa so tudi sheme.
- Primeren je predvsem za podatkovne vire, namenjene podpori odločanju. Ker so ti podatkovni viri namenjeni večinoma branju podatkov (poizvedovanju), je dopustno podvajanje podatkov, ki je značilno predvsem za dimenzijske tabele.
- Poizvedovanje je zaradi specifične strukture praviloma hitrejše kot pri relacijskem modelu.

Osnove dobrega načrtovanja podatkovne baze

Zdaj, ko vemo, kako *moremo* organizirati podatke v okviru izbranega podatkovnega modela, si poglejmo, kako *moramo* organizirati podatke, oziroma, kako pridemo do dobrega načrta podatkovne baze[28]. Pri tem se bomo omejili zgolj na relacijski podatkovni model zaradi njegove razširjene uporabe in razumljivosti.

Kakovost relacijske sheme

Očitno je, da moremo v okviru istega modela iste podatke organizirati oziroma strukturirati na različne načine. Na primer, namesto uporabe relacijske sheme (označimo jo z R):

R

NAROČILO (NAROČILO#, DATUM, KUPEC#, TP#, POPUST)
IZDELEK (IZDELEK#, NAZIVIZD, ME, CENA)
POSTAVKA (IZDELEK#, NAROČILO#, KOLIČINA)
KUPEC (KUPEC#, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT)
TRG-POTNIK (TP#, IMETP, PRIIMEKTP, REGIJA)

bi lahko tvorili podatkovno bazo, ki bi bila strukturirana takole (relacijska shema P):

P

NAROČILO (NAROČILO#, DATUM, KUPEC#, TP#, POPUST, IMETP, PRIIMEKTP, REGIJA)
POSTAVKA (NAROČILO#, IZDELEK#, NAZIVIZD, ME, CENA, KOLIČINA)
KUPEC (KUPEC#, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT)

S stališča zahtev relacijskega modela taka organizacija podatkov ni napačna, je pa taka relacijska shema (načrt) podatkovne baze slab.

Kakovost relacijske sheme podatkovne baze lahko ocenjujemo glede na izpolnjevanje štirih zahtev, ki jim mora ta shema ustrezati (Elmasri, Navathe, 2000):

- semantika naj bo jasna,
- podvajanje podatkov mora biti čim manjše,

---

28 Pri tem velja opozoriti, da načela, kot jih prikazujemo v tem podpoglavju, veljajo predvsem za operativne podatkovne vire. Pri podatkovnih virih, namenjenih podpori odločanju, podvajanje podatkov, katerega odsotnost je poglavitno merilo kakovosti sheme, ne predstavlja večjih težav.

- atributi morajo imeti čim manj vrednosti NULL ter
- shema ne sme dopuščati napačnih vrstic.

Jasna semantika. Atribute združujemo v relacije tako, da je enostavno razumeti njen pomen. Na primer, v relacijski shemi P je lahko razumeti, da vsaka vrstica relacije KUPEC predstavlja podatke o enem kupcu. Po drugi strani pa v isti relacijski shemi relacija NAROČILO vsebuje atribute naročil in trgovskih potnikov, kar otežuje razumevanje pomena (semantike) te relacije.

V splošnem velja, da je relacijska shema toliko bolj kakovostna, kolikor je lažje razumeti semantiko (pomen) relacij. Med drugim to pomeni, da ne smemo mešati atributov različnih entitetnih tipov ali tipov povezav v isti relaciji.

Podvajanje podatkov. Podvajanje podatkov v podatkovni bazi je nezaželeno, ker povzroča različne težave (anomalije) pri vzdrževanju. Anomalije lahko razdelimo v tri skupine:

- **Anomalije pri vnosu**. Če želimo v primeru uporabe relacijske sheme P vnesti ob zaposlitvi podatke o trgovskem potniku, tega ne moremo storiti, dokler trgovski potnik ne dobi vsaj enega naročila, saj so podatki o trgovskih potnikih vezani na naročila (relacija NAROČILO). Problem bi na (zelo neroden) način rešili tako, da bi vnesli namišljena naročila. Z vnosom podatkov v slabo načrtovano podatkovno bazo pa je povezan še en problem. Ko v relacijo NAROČILO relacijske sheme P vnašamo podatke o naročilih, moramo za vsako naročilo vnesti podatke o trgovskem potniku. To pomeni, da moramo mnogokrat vnesti iste podatke, pri čemer moramo paziti, da ohranjamo konsistentnost podatkovne baze, torej, da so podatki o istem trgovskem potniku vedno enaki. V relacijski shemi R teh problemov ni, saj podatke o vsakem trgovskem potniku vnesemo le enkrat (četudi še ni sprejel nobenega naročila, ni problemov), za vsako naročilo, ki ga sprejme, pa vnesemo le njegovo šifro.
- **Anomalije pri spreminjanju** vrednosti podatkov. Če trgovski potnik, na primer, spremeni priimek, moramo v primeru uporabe relacijske sheme P, kjer se podatki o trgovskih potnikih podvajajo, ta priimek spremeniti povsod. To pomeni poleg odvečnega dela tudi večjo možnost napak. V

NARODNA IN UNIVERZITETNA KNJIŽNICA LJUBLJANA

relacijski shemi R teh problemov zopet nimamo, saj vsebuje posebno relacijo s podatki o trgovskih potnikih, kjer je priimek vsakega trgovskega potnika shranjen le enkrat.

- **Anomalije pri brisanju**. Vzemimo za primer (spet uporabljamo nekakovostno relacijsko shemo P), da smo za nek izdelek I do sedaj dobili le eno naročilo. Podatki o izdelku so torej zabeleženi le enkrat v tabeli POSTAVKA. Če kupec prekliče omenjeno naročilo, moramo zbrisati poleg podatkov o naročilu (relacija NAROČILO) tudi postavke naročila (relacija POSTAVKA). S tem pa smo izgubili tudi edini zapis o izdelku I, torej v podatkovni bazi nimamo več podatkov o tem izdelku.

Podatkovno bazo načrtujemo tako, da ne prihaja do naštetih anomalij. Če od tega pravila odstopamo, moramo poskrbeti, da programi, ki delajo s podatkovno bazo to upoštevajo, na primer, da ob spremembi podvojenega podatka avtomatično poskrbijo za ažuriranje vseh njegovih kopij.

Vrednosti NULL. Veliko število vrednosti NULL lahko predstavlja težave pri uporabi podatkovne baze (predvsem pri združevanju podatkov iz več tabel), hkrati pa otežuje razumevanje pomena relacije. Vrednost NULL lahko razumemo na različne načine:

- entiteta, ki jo predstavlja vrstica, nima vrednosti za ta atribut,
- vrednost atributa za to vrstico ni znana ali
- vrednost atributa je znana, a manjkajoča (je še nismo vnesli v podatkovno bazo).

Iz teh razlogov atributov, za katere pričakujemo, da bodo imeli v relaciji mnogo vrednosti NULL, v to relacijo ne vključujemo. Na primer, če imamo relacijo s podatki o delavcih

DELAVEC (ŠIFRA, IME, PRIIMEK, ŠIFRA, EMŠO, PLAČA, E-POŠTA)

in vemo, da ima le okoli 5 % delavcev e-poštni naslov, potem v stolpcu E-POŠTA pričakujemo mnogo (95 %) vrednosti NULL. V tem primeru je bolj smiselno narediti posebno relacijo z naslovi e-pošte.

DELAVEC (ŠIFRA, IME, PRIIMEK, ŠIFRA, EMŠO, PLAČA)
E-NASLOV (ŠIFRA, E-POŠTA)

V relaciji E-NASLOV bodo podatki o e-poštnih naslovih le za delavce, ki tak naslov imajo, po potrebi pa v poizvedbi podatke iz obeh tabel združimo.

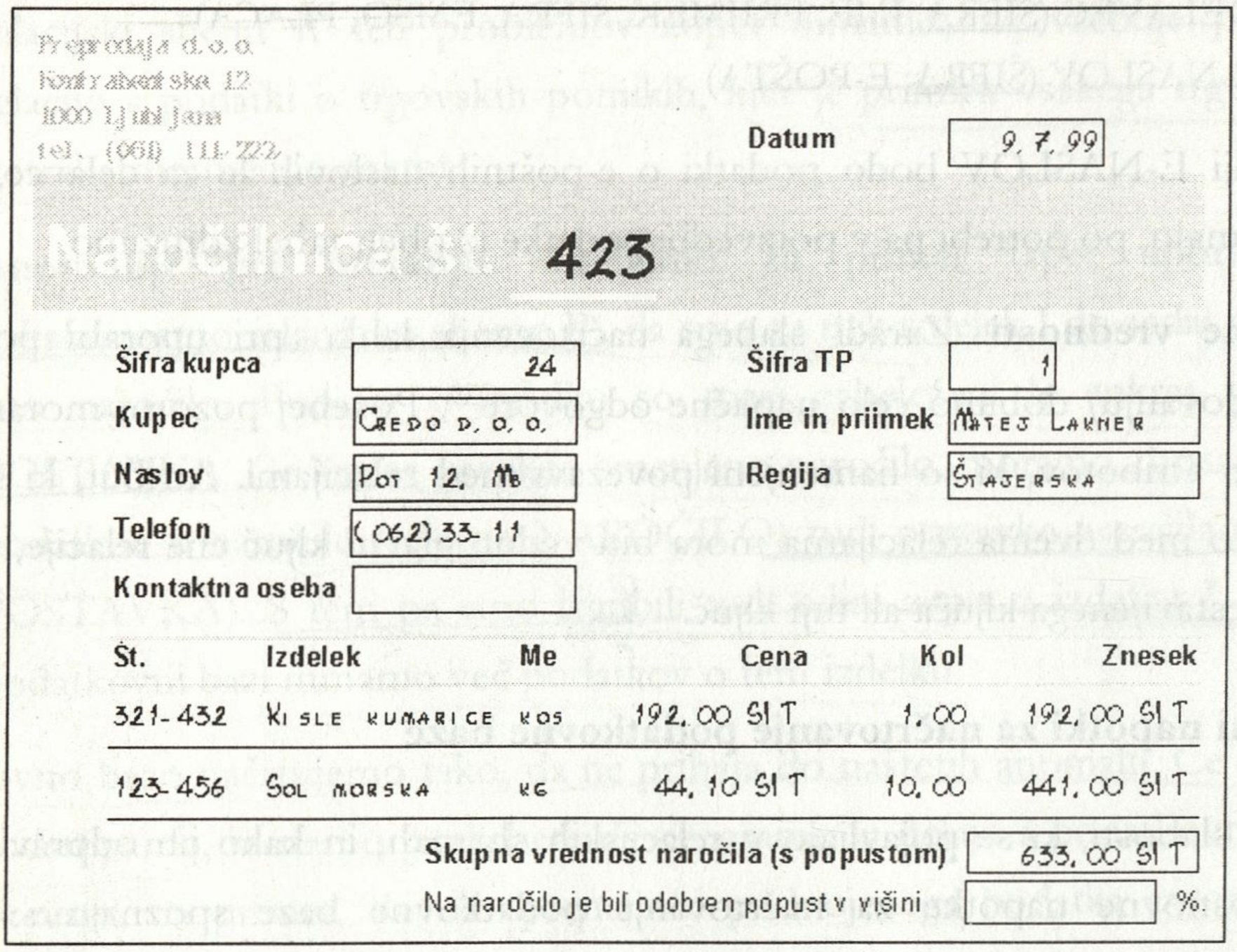
Napačne vrednosti. Zaradi slabega načrtovanja lahko pri uporabi podatkov (poizvedovanju) dobimo celo napačne odgovore[29]. Posebej pozorni moramo biti pri izbiri atributov, ki so namenjeni povezavi med relacijami. Atribut, ki služi za povezavo med dvema relacijama mora biti vedno glavni ključ ene relacije, v drugi pa del sestavljenega ključa ali tuji ključ.

Osnovni napotki za načrtovanje podatkovne baze

Tipične slabosti, ki se pojavljajo v relacijskih shemah, in kako jih odpravimo ter druge osnovne napotke za načrtovanje podatkovne baze spoznajmo kar na primeru.

Tvoriti želimo podatkovno bazo za poslovno področje prodaje. Zaradi enostavnosti se bomo omejili samo na sprejem naročil. Naši trgovski potniki obiskujejo stranke (podjetja); vsak od njih je zadolžen za neko slovensko regijo. Ob vsakem obisku izpolnijo naročilnico, katere primerek je prikazan na sliki 34.

29 Ker je razumevanje težav, ki se pri tem pojavijo, nekoliko težje, jih tu ne obravnavamo. Opisane so, na primer, v (Elmasri, Navathe, 2000).


Proizvodnja d.o.o.
Kontrabantska 12
1000 Ljubljana
tel. (061) 111-222

Datum 9. 7. 99

Naročilnica št. 423

Šifra kupca 24
Kupec CREDO D. O. O.
Naslov POT 12, MB
Telefon (062) 33-11
Kontaktna oseba

Šifra TP 1
Ime in priimek MATEJ LAKNER
Regija ŠTAJERSKA

Št.	Izdelek	Me	Cena	Kol	Znesek
321-432	KISLE KUMARICE	KOS	192.00 SIT	1.00	192.00 SIT
123-456	SOL MORSKA	KG	44.10 SIT	10.00	441.00 SIT

Skupna vrednost naročila (s popustom) 633.00 SIT

Na naročilo je bil odobren popust v višini %.

Slika 34: Primer naročilnice

Zaradi boljšega spremljanja in učinkovitejše realizacije naročil ter boljše podpore poslovnemu odločanju želimo te podatke shranjevati v podatkovni bazi [30].

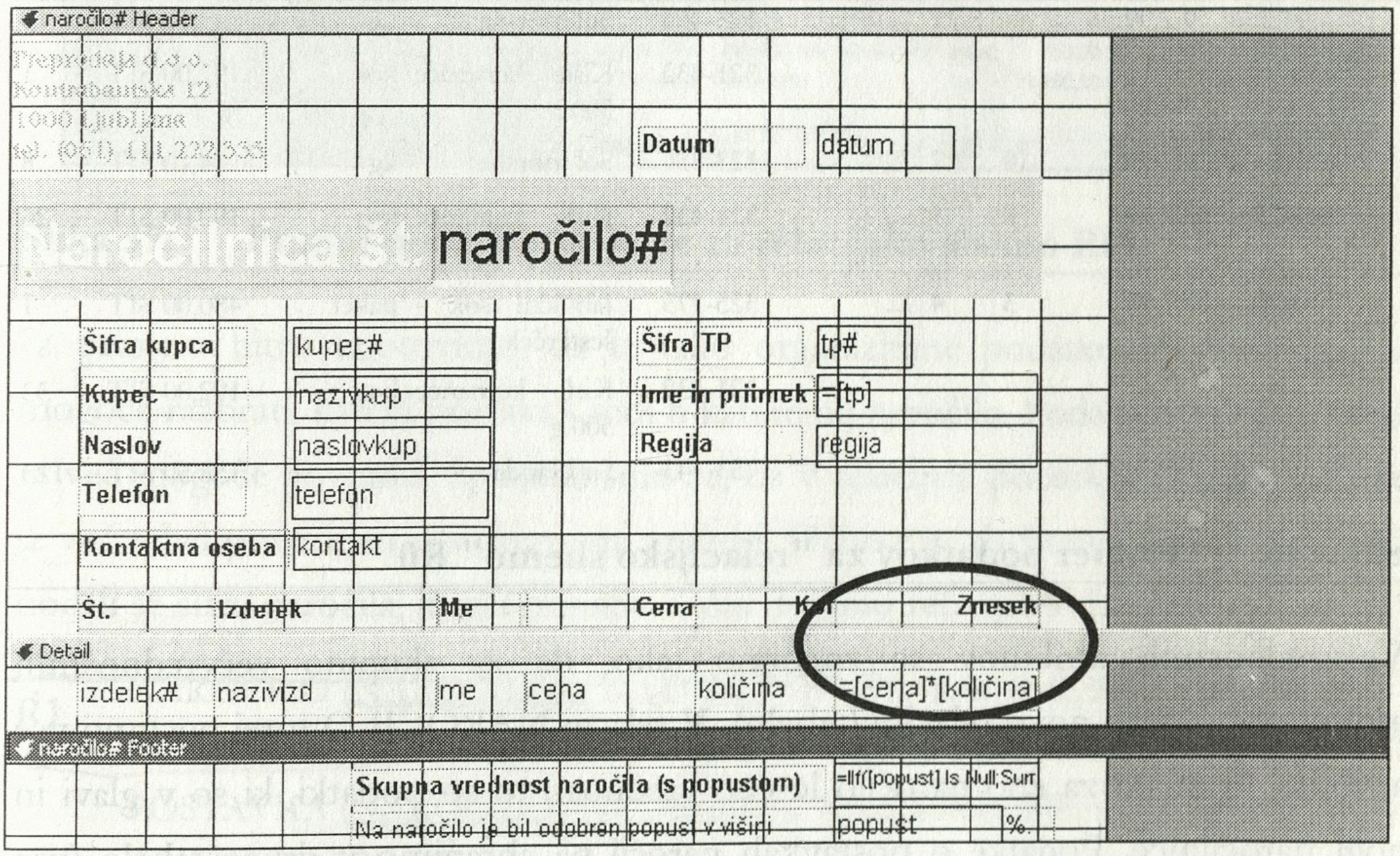
Katere podatke shranimo v bazo? *Praviloma* v podatkovni bazi *ne* shranjujemo podatkov, ki so enaki na vsakem izvodu dokumenta. V našem primeru se na vsaki naročilnici pojavi naziv, naslov in drugi podatki o našem podjetju. Teh podatkov ne bomo shranili v bazi, pač pa bodo na šabloni naročilnice (poročilu). Izpis naročilnice na tiskalniku bo torej sestavljen iz šablone, ki vsebuje fiksne elemente vsake naročilnice in opredeljuje obliko izpisa, ter podatkov o naročilu, ki bodo prebrani iz podatkovne baze. Prav tako *praviloma ne* shranjujemo izračunanih (izpeljanih, izvedenih) podatkov, to je podatkov, ki jih izračunamo (izpeljemo) iz drugih, saj jih lahko programi hitro izračunajo[31]. V bazi torej ne bomo shranili zneska za posamezni izdelek in skupne vrednosti naročila. Šablona za izpis

---

30 Pri tem nas trenutno ne zanima, na kakšen način bo potekal vnos naročil. Vsekakor ne bi bilo smiselno izpolnjevanje naročilnic in naknadno pretipkavanje podatkov v podatkovno bazo.

31 V praksi zaradi različnih vzrokov to pravilo tudi kršimo. Na primer, zaradi problema zaokroževanja in natančnosti knjigovodskih knjižb pogosto hranimo tudi znesek naročila, računa, davka …

naročilnice bo na mestu teh podatkov vsebovala ustrezne formule za izračun iz cene izdelka in naročene količine, ob izpisu naročilnice (na zaslon ali tiskalnik) pa se vrednosti izračunajo po formuli in izpišejo. Zakaj ni smiselno shranjevati izračunanih podatkov? Če bi se spremenila vrednost nekega podatka, bi morali spremeniti vse shranjene vrednosti iz tega podatka izračunanih podatkov. Če imamo shranjen datum rojstva, ni smiselno shranjevati starosti, saj bi morali venomer vrednost tega podatka (starost) spreminjati.



Slika 35: Šablona naročilnice

Znebimo se večvrednostnih atributov. Pri razmišljanju, kako bi organizirali podatke z naročilnice s slike 34, bi bila prva in najbolj preprosta misel, da bi vse podatke shranjevali v eni sami tabeli, v kateri bi vsaka vrstica predstavljala eno naročilo, torej podatke z ene naročilnice ("relacijska shema" R0):

R0 NAROČILO (NAROČILO#, DATUM, POPUST, TP#, IMETP, PRIIMEKTP, REGIJA, KUPEC#, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT, IZDELEK#, NAZIVIZD, ME, CENA, KOLIČINA)

Izpeljanih atributov nismo vključili v shemo v skladu s prejšnjim nasvetom. Problem pri tej shemi so večvrednostni atributi, to so tisti, ki imajo na eni naročilnici več vrednosti (IZDELEK#, NAZIVIZD, ME, CENA, KOLIČINA).

Vemo, da relacijski model ne dopušča večvrednostnih atributov (glej poglavje Relacijski model podatkov) in ni mogoče tvoriti podatkovne baze, ki bi bila definirana z R0, ki potemtakem sploh ni relacijska shema. Sistemi za upravljanje podatkovnih baz torej *ne* dovoljujejo tvorbe tabel, kakršna je prikazana na sliki 36.

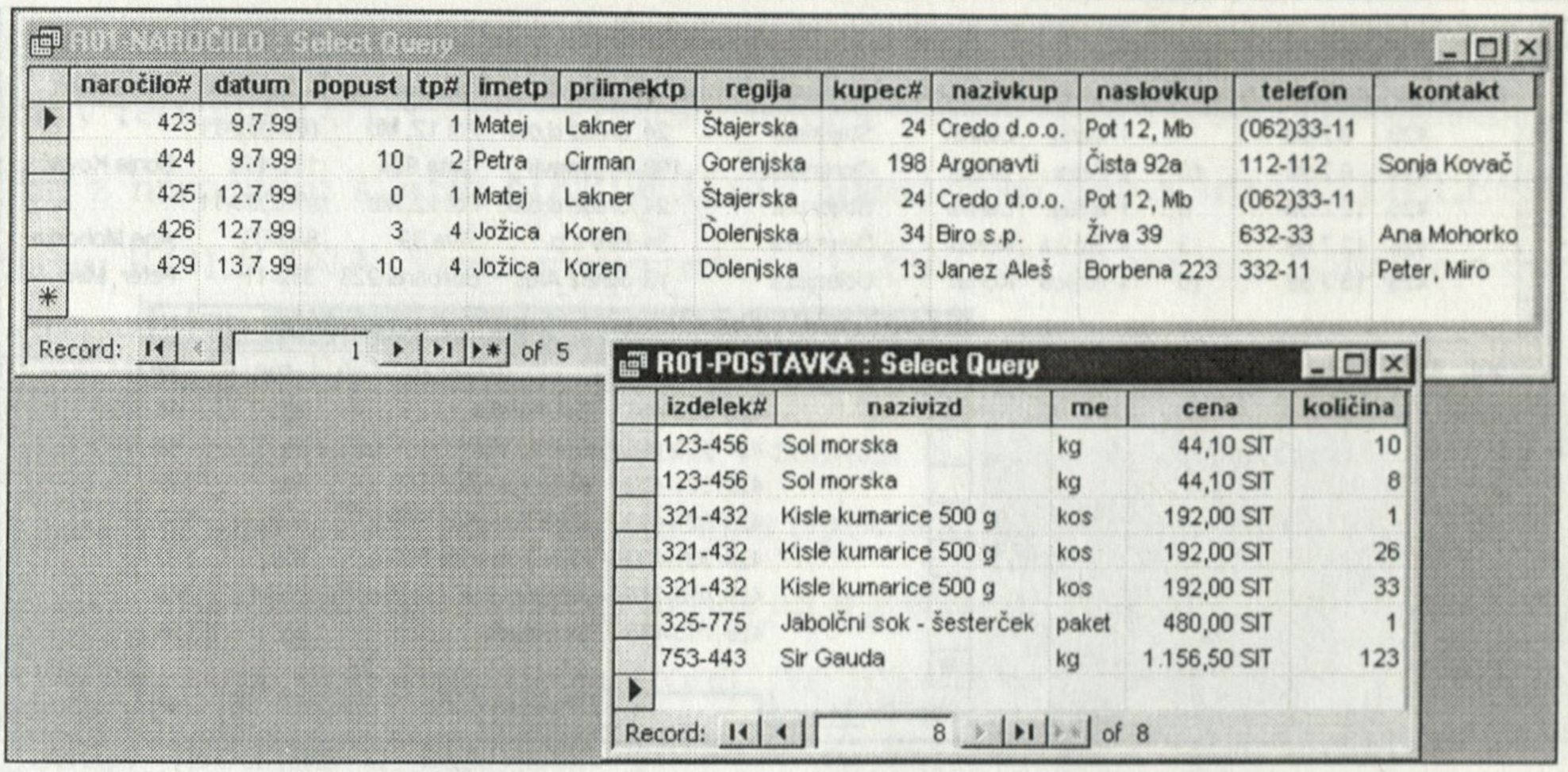
naročilo#	datum	popust	tp#	ostali podatki o TP in kupcu	izdelek#	nazivizd	me	cena	količina
423	9.7.99		1	...	123-456	Sol morska	kg	44,10 SIT	10
					321-432	Kisle kumarice 500 g	kos	192,00 SIT	1
424	9.7.99	10	2	...	123-456	Sol morska	kg	44,10 SIT	8
425	12.7.99	0	1	...	321-432	Kisle kumarice 500 g	kos	192,00 SIT	26
426	12.7.99	3	4	...	325-775	Jabolčni sok - šesterček	paket	480,00 SIT	1
					321-432	Kisle kumarice 500 g	kos	192,00 SIT	33
					753-443	Sir Gauda	kg	1.156,50 SIT	123

Slika 36: Primer podatkov za "relacijsko shemo" R0

Večvrednostnih atributov se znebimo tako, da za skupino večvrednostnih atributov tvorimo novo relacijo (tabelo). V relaciji NAROČILO torej pustimo vse atribute, ki imajo za eno naročilo le eno vrednost; to so podatki, ki so v glavi in nogi naročilnice. Podatke o postavkah naročil pa shranimo v drugo tabelo (vse postavke z vseh naročil). Relacijska shema *bi bila* potemtakem (R01):

R01 NAROČILO (NAROČILO#, DATUM, POPUST, TP#, IMETP, PRIIMEKTP, REGIJA, KUPEC#, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT)
POSTAVKA (IZDELEK#, NAZIVIZD, ME, CENA, KOLIČINA)

Primer podatkov je prikazan na sliki 37.



Slika 37: Primer podatkovne baze za relacijsko shemo R01

Iz primera hitro ugotovimo, da iz tako organizirane podatkovne baze ni več mogoče razbrati, katera postavka sodi h kateremu naročilu. Podatkov iz obeh tabel ni več mogoče povezati. Spoznali smo že, da v relacijski podatkovni bazi podatke iz več tabel povezujemo s skupnimi atributi. Očitno je, da bomo k vsaki postavki dodali še šifro naročila, ki mu pripada. Tako dobimo relacijsko shemo R1:

R1
NAROČILO (NAROČILO#, DATUM, POPUST, TP#, IMETP, PRIIMEKTP, REGIJA, KUPEC#, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT)
POSTAVKA (NAROČILO#, IZDELEK#, NAZIVIZD, ME, CENA, KOLIČINA)

V obeh relacijah smo določili glavna ključa. Opazimo, da ima druga relacija sestavljen glavni ključ, saj nobeden od atributov sam ne zadostuje za ključ. Pomen obeh relacij je, upajmo, očiten: vsaka vrstica relacije NAROČILO predstavlja podatke o enem naročilu, vsaka vrstica relacije POSTAVKA pa eno postavko nekega naročila.

R1-NAROČILO : Select Query

naročilo#	datum	popust	tp#	imetp	priimektp	regija	kupec#	nazivkup	naslovkup	telefon	kontakt
423	9.7.99		1	Matej	Lakner	Štajerska	24	Credo d.o.o.	Pot 12, Mb	(062)33-11	
424	9.7.99	10	2	Petra	Cirman	Gorenjska	198	Argonavti	Čista 92a	112-112	Sonja Kovač
425	12.7.99	0	1	Matej	Lakner	Štajerska	24	Credo d.o.o.	Pot 12, Mb	(062)33-11	
426	12.7.99	3	4	Jožica	Koren	Dolenjska	34	Biro s.p.	Živa 39	632-33	Ana Mohorko
429	13.7.99	10	4	Jožica	Koren	Dolenjska	13	Janez Aleš	Borbena 223	332-11	Peter, Miro

Record: 1 of 5

R1-POSTAVKA : Select Query

naročilo#	izdelek#	nazivizd	me	ce
423	123-456	Sol morska	kg	44
424	123-456	Sol morska	kg	44
423	321-432	Kisle kumarice 500 g	kos	192
425	321-432	Kisle kumarice 500 g	kos	192
426	321-432	Kisle kumarice 500 g	kos	192
426	325-775	Jabolčni sok - šesterček	paket	480
426	753-443	Sir Gauda	kg	1.156

Record: 1 of 7

Slika 38: Primer podatkovne baze za relacijsko shemo R1

Splošno pravilo je: V primeru, ko bi imēli v relaciji večvrednostne atribute, jih izločimo v *novo* relacijo, kateri (razen večvrednostnih atributov) *dodamo* še *ključ* relacije, ki nima večvrednostnih atributov. Nova relacija bo imela vedno sestavljen glavni ključ, katerega del bo tudi dodani ključ druge relacije.

Odpravimo podvajanje podatkov. Shema R1 je sicer povsem v skladu z zahtevami relacijskega modela in je mogoče tvoriti tako podatkovno bazo. Pogled na primer podatkov za to shemo (glej sliko 38) pa pove, da R1 ni najboljša shema, saj je v primeru precej podvajanja podatkov. V relaciji POSTAVKA se pri vsakem naročilu nekega izdelka pojavijo vsi njegovi podatki (NAZIVIZD, ME, CENA). Tako ponavljanje povzroča anomalije (glej podpoglavje Kakovost relacijske sheme). Če se, na primer, spremeni cena izdelka, moramo ta podatek spremeniti pri vseh naročilih. Zato je smiselno tvoriti novo relacijo IZDELEK, ki bo vsebovala podatke o izdelkih, o vsakem izdelku samo enkrat. Ta nova tabela vsebinsko predstavlja šifrant izdelkov oziroma cenik. Povezavo med relacijama POSTAVKA in IZDELEK vzpostavimo s skupnim atributom, očitno bo to IZDELEK#. Nova relacijska shema je potem (R2):

R2
NAROČILO (NAROČILO#, DATUM, POPUST, TP#, IMETP, PRIIMEKTP, REGIJA, KUPEC#, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT)
POSTAVKA (NAROČILO#, IZDELEK#, KOLIČINA)
IZDELEK (IZDELEK#, NAZIVIZD, ME, CENA)

Ker se količina naročenega izdelka razlikuje od naročila do naročila, bo ta atribut ostal v relaciji POSTAVKA. Vrstica relacije POSTAVKA vsebuje informacijo, da je bil z naročilom s šifro NAROČILO# naročen izdelek s šifro IZDELEK# v količini KOLIČINA. Vsaka vrstica relacije IZDELEK pa vsebuje podatke o enem izdelku[32].

Podvajanje podatkov je prisotno tudi v relaciji NAROČILO, pojavi se kar za dve skupini atributov. Podatki o kupcih in o trgovskih potnikih se ponavljajo pri vseh naročilih, pri katerih gre za istega kupca ali trgovskega potnika. Tudi to podvajanje odstranimo na podoben način, torej tvorimo novi relaciji, za vsako skupino atributov (podatki o trgovskih potnikih in kupcih) eno. Spet moramo seveda poskrbeti za atribute, ki bodo služili za povezavo podatkov iz tabel. Dvom lahko povzroča atribut POPUST, če ne poznamo njegovega pomena. Je to popust, vezan na kupca (določenemu kupcu priznavamo vedno isti popust), morda na trgovskega potnika? Vzemimo, da se lahko vsak trgovski potnik za vsako naročilo posebej odloča za višino popusta. Torej je atribut POPUST vezan na naročilo. Zdaj lahko razbijemo relacijo NAROČILO in dobimo relacijsko shemo (R3).

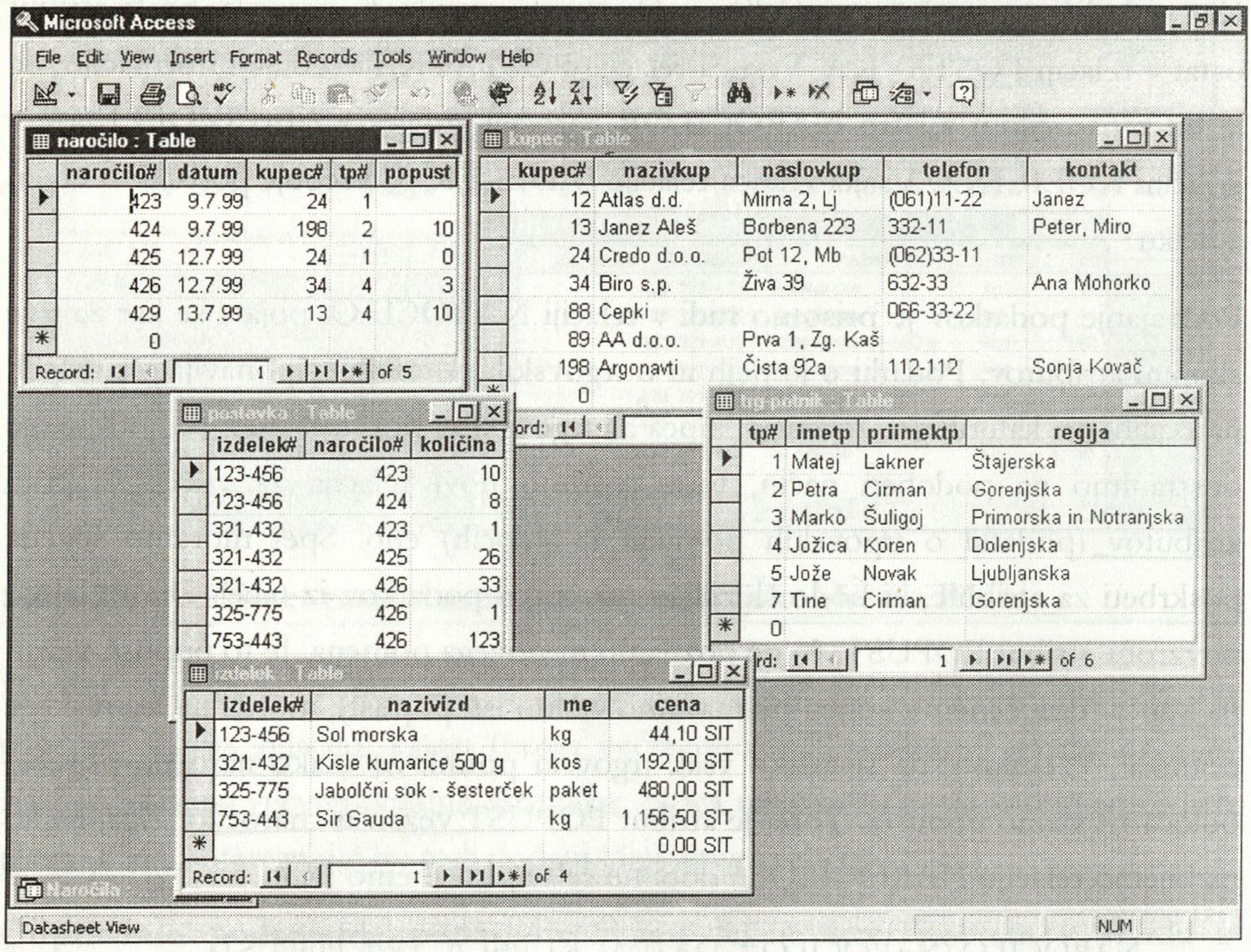
R3

NAROČILO (NAROČILO#, DATUM, KUPEC#, TP#, POPUST)
IZDELEK (IZDELEK#, NAZIVIZD, ME, CENA)
POSTAVKA (IZDELEK#, NAROČILO#, KOLIČINA)
KUPEC (KUPEC#, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT)
TRG-POTNIK (TP#, IMETP, PRIIMEKTP, REGIJA)

Za vsako naročilo torej zapišemo le podatke o šifri kupca in trgovskega potnika, saj lahko ustrezne podrobnejše podatke najdemo v ustreznih tabelah – šifrantih.

---

32 Zaradi poenostavitve smo v tem primeru namenoma spregledali problematiko časovne odvisnosti cene izdelka, kar v praksi običajno rešujemo z uvedbo še ene relacije, ki za isti izdelek vsebuje cene, veljavne v različnih časovnih obdobjih, na primer CENIK (IZDELEK#, DATUMOD#, DATUMDO#, CENA).

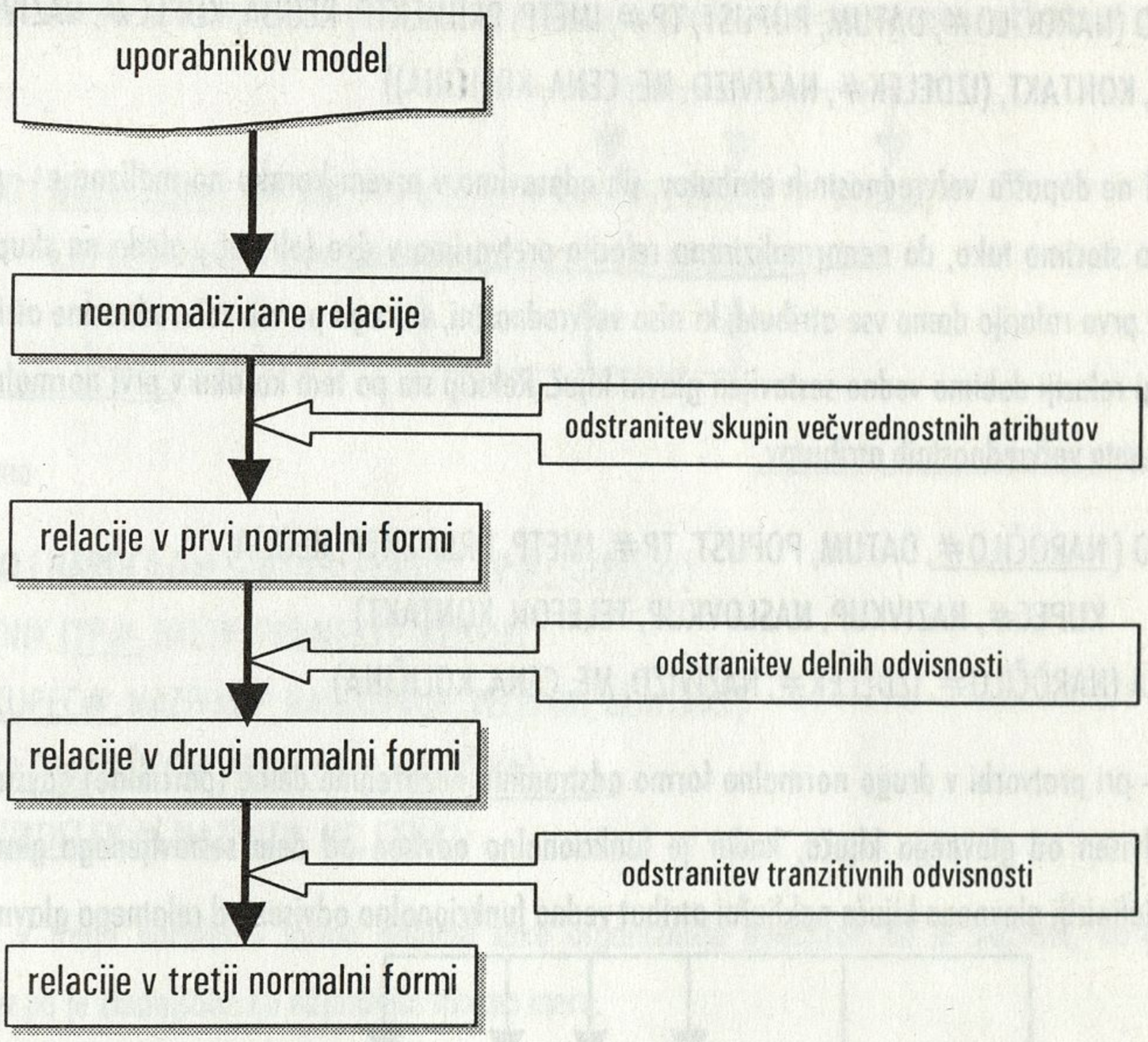


Slika 39: Primer podatkovne baze za relacijsko shemo R3

Drugih (nepotrebnih) podvajanj podatkov v relacijski shemi R3 ni, zato je primerna kot načrt za podatkovno bazo.

Opozorimo bralca, da v primeru, ko imamo v bazi dve enaki vrednosti nekega atributa, to ni nujno podvajanje podatkov. Na primer, če imamo dvakrat isto vrednost atributa POPUST v relaciji NAROČILO, vsaka izmed (sicer enakih) vrednosti predstavlja informacijo o višini popusta za eno izmed naročil. Te vrednosti ne moremo shraniti samo enkrat, ne da bi izgubili informacijo o vrednosti popustov za enega izmed teh dveh naročil.

Opisani razmisleki so zaobjeti v formalnem postopku, s katerim v treh korakih začetno shemo zanesljivo prevedemo v tako shemo, pri kateri je podvajanje podatkov minimalno. Pri normalizaciji, kot se postopek imenuje, uporabljamo analizo funkcionalnih odvisnosti med atributi (glej poglavje Funkcionalna odvisnost). Normalizacijo je mogoče uporabiti na vseh vrstah modelov podatkov (hierarhičnem, mrežnem, relacijskem in objektnem), vendar se v nadaljevanju omejimo le na primer relacijskega modela, ki je za današnjega uporabnika najzanimivejši in najlažje dostopen.



Kot bomo videli, postopek normalizacije le formalizira razmisleke, ki smo jih že spoznali v bolj neformalni obliki.

Začnemo z nekim uporabniškim modelom, to je uporabnikovim pogledom na podatke, ki ga prevedemo v t.i. nenormalizirano relacijo. To naredimo tako, da nenormalizirana relacija vsebuje vse atribute uporabnikovega modela. Nekateri atributi so lahko večvrednostni, kar ugotovimo tako, da analiziramo povezave med atributom, ki identificira uporabniški model, in ostalimi atributi. Vzemimo za primer uporabniškega modela spet dokument Naročilnica s slike 34. Šifra naročila identificira naročilnico, zato analizirajmo povezave med atributom NAROČILO# in ostalimi atributi.



Vidimo, da skupino večvrednostnih atributov sestavljajo postavke naročil. V relacijski shemi skupine večvrednostnih atributov označimo tako, da jih omejimo s še enim (notranjim) oklepajem.

NAROČILO (NAROČILO#, DATUM, POPUST, TP#, IMETP, PRIIMEKTP, REGIJA, KUPEC#, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT, (IZDELEK#, NAZIVIZD, ME, CENA, KOLIČINA))

Ker relacijski model ne dopušča večvrednostnih atributov, jih odpravimo v prvem koraku normalizacije – pri pretvorbi v prvo normalno formo. To storimo tako, da nenormalizirano relacijo pretvorimo v dve (ali več , glede na skupine večvrednostnih atributov) relaciji. V prvo relacijo damo vse atribute, ki niso večvrednostni, v drugo pa vse večvrednostne atribute in glavni ključ prve relacije. V drugi relaciji dobimo vedno sestavljen glavni ključ. Relaciji sta po tem koraku v prvi normalni formi (1 NF), kar pomeni, da ne vsebujeta večvrednostnih atributov.

NAROČILO (NAROČILO#, DATUM, POPUST, TP#, IMETP, PRIIMEKTP, REGIJA, KUPEC#, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT)
POSTAVKA (NAROČILO#, IZDELEK#, NAZIVIZD, ME, CENA, KOLIČINA)

V drugem koraku – pri pretvorbi v drugo normalno formo odstranimo nezaželene delne (parcialne) odvisnosti. Nek neključni atribut je delno odvisen od glavnega ključa, kadar je funkcionalno odvisen od dela sestavljenega glavnega ključa. Velja opozoriti, da je po definiciji glavnega ključa neključni atribut vedno funkcionalno odvisen od celotnega glavnega ključa.

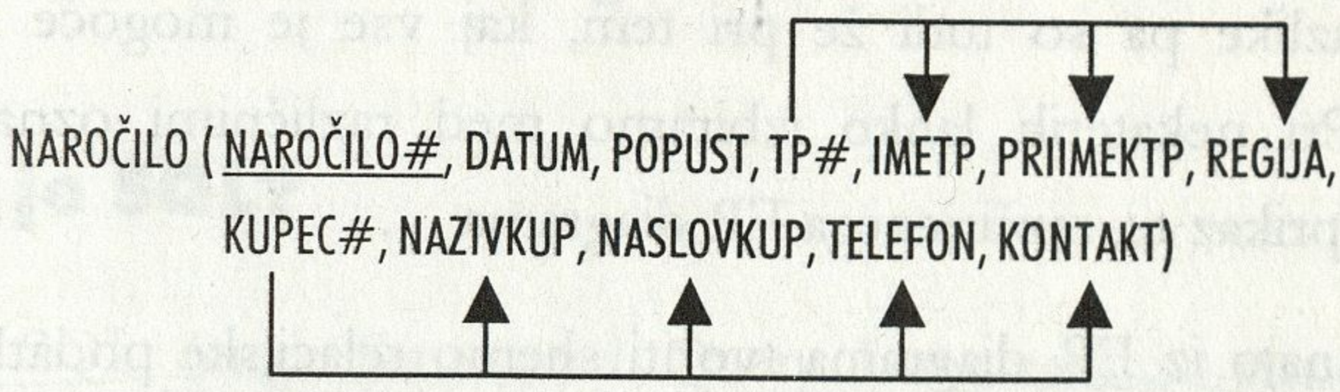


Očitno je tudi, da v relacijah, ki nimajo sestavljenega glavnega ključa, ne moremo imeti delnih odvisnosti. Delne odvisnosti odstranimo tako, da relacijo, ki jih vsebuje, razbijemo v dve (ali več, če obstajajo delne odvisnosti od različnih delov glavnega ključa). V eno relacijo damo celoten glavni ključ in vse atribute, ki so odvisni samo od celotnega glavnega ključa, v drugo relacijo pa vse neključne atribute, ki so delno odvisni od glavnega ključa, in še ustrezni del glavnega ključa. Po razbitju so relacije v drugi normalni formi (2 NF), ker so vsi neključni atributi odvisni le od celotnega sestavljenega glavnega ključa.

NAROČILO (NAROČILO#, DATUM, POPUST, TP#, IMETP, PRIIMEKTP, REGIJA, KUPEC#, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT)
POSTAVKA (NAROČILO#, IZDELEK#, KOLIČINA)
IZDELEK (IZDELEK#, NAZIVIZD, ME, CENA)

V zadnjem koraku normalizacije, pretvorbi v tretjo normalno formo, odstranimo še tranzitivne odvisnosti, ki prav tako povzročajo težave (anomalije) pri uporabi podatkovne baze. Tranzitivna odvisnost je funkcionalna odvisnost med dvema neključnima atributoma. Relacijo, ki vsebuje tranzitivne odvisnosti, spet razbijemo v dve (ali več, glede na število neključnih atributov, od katerih so drugi neključni atributi odvisni) relaciji. Če je skupina neključnih atributov odvisna od neključnega atributa A, potem damo v eno relacijo vse te atribute vključno z A. V tej relaciji bo A glavni ključ. V drugi relaciji bodo vsi atributi, ki niso v prvi, dodamo pa še atribut A, ki bo tu tuji ključ. Po razbitju so vse relacije v tretji normalni formi (3 NF), saj ne vsebujejo neželenih tranzitivnih odvisnosti.

Spet torej uporabimo analizo funkcionalnih odvisnosti, da poiščemo morebitne odvisnosti med neključnimi atributi.



in z razbitjem dobimo

NAROČILO (NAROČILO#, DATUM, POPUST, TP#, KUPEC#)
TRG-POTNIK (TP#, IMETP, PRIIMEKTP, REGIJA)
KUPEC (KUPEC#, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT)
POSTAVKA (NAROČILO#, IZDELEK#, KOLIČINA)
IZDELEK (IZDELEK#, NAZIVIZD, ME, CENA)

S pretvorbo relacij v tretjo normalno formo dobimo tako organizacijo podatkov, ki je stabilna, ne povzroča anomalij, podvajanje podatkov pa je zmanjšano na najmanjšo možno mero.

Poleg omenjenih normalnih form obstajajo še višje normalne forme, vendar so te bolj zapletene, hkrati pa obravnavajo probleme, ki se v praksi pojavijo zelo redko. Zato jih tukaj ne obravnavamo.

To, da smo podatke z ene naročilnice porazdelili v več tabel, seveda ne pomeni, da bo moral uporabnik, če bo želel pregledati naročilnico, brskati po vseh tabelah. Za uporabnika vedno pripravimo tak pogled na podatke, kot ga potrebuje. Na sliki 34 je prikazan primer pogleda na podatke (naročilnica), ki prikazuje hkrati podatke iz več tabel. Drug pogled na iste podatke (iz istih tabel) je prikazan na sliki 17 (b).

Prednost organizacije podatkov, kot smo jo oblikovali zgoraj, pa je, da v primeru novega naročila istega kupca uporabnik ne bo več vnašal vseh podatkov o kupcu, pač pa se z vnosom šifre kupca takoj izpišejo vsi njegovi podatki, ki so že shranjeni v tabeli o kupcu. Hkrati taka organizacija podatkov omogoča, da na isto podatkovno bazo "obesimo" različne poglede. Tako bi lahko izdelali poročila o prodaji po kupcih, po izdelkih ali po mesecih, seznam vseh kupcev, cenik izdelkov. Pri tem pa se vsaka sprememba v podatkih takoj odraža v vseh teh pogledih.

Programska orodja za podporo razvoju podatkovnih baz

Za podporo razvoju podatkovnih baz je na voljo velika izbira programskih orodij, ki sodijo v skupino orodij CASE (angl. Computer Aided Software Engineering).

Orodja se med seboj bistveno razlikujejo po možnostih, ki jih poleg risanja ER diagramov ponujajo, razlike pa so tudi že pri tem, kaj vse je mogoče z ER diagramom prikazati. Pri nekaterih lahko izbiramo med različnimi oznakami, nekateri dopuščajo tudi prikaz t.i. razširjenega ER diagrama …

Kakovostnejša orodja znajo iz ER diagrama tvoriti shemo relacijske podatkovne baze, ustrezne stavke jezika SQL za tvorbo podatkovne baze, ali celo kar neposredno tvoriti podatkovno bazo[33]. Pri tem poskrbijo tudi za pravilno strukturo podatkov – odpravijo podvajanje podatkov. Seveda za slednje potrebujejo poslovna pravila, iz katerih razberejo funkcionalne odvisnosti med atributi.

Pogosto so orodja za podporo razvoju podatkovnih baz le del orodij CASE, ki podpirajo razvoj celotnega informacijskega sistema, od modeliranja preko načrtovanja pa vse do generiranja programske kode in podatkovne baze.

Najboljša orodja imajo tudi možnost t.i. vzvratnega inženirstva (angl. backward engineering), kar pomeni, da iz obstoječe podatkovne baze izdelajo ER diagram. Ta možnost je koristna pri prenovi informacijskega sistema, saj naredimo spremembe na ER diagramu in potem ponovno tvorimo prenovljeno podatkovno bazo.

Pomembna je tudi možnost izdelave dokumentacije podatkovne baze, saj so sistemi brez dokumentacije vse prepogost pojav v praksi, kar otežkoča vzdrževanje in kasnejše prenove sistema. Če orodje avtomatično tvori dokumentacijo, je verjetnost, da bo podatkovna baza dokumentirana, gotovo večja.

Seveda je kakovosti primerna cena; nekatere bolj preproste pakete lahko dobimo celo brezplačno, cene najboljših orodij CASE pa so zelo visoke.

Orodja CASE lahko bistveno prispevajo k učinkovitosti in kakovosti dela analitikov, načrtovalcev sistema in programerjev, vendar to ne pomeni, da bodo ob uporabi takega orodja ostali brez dela. Ta orodja so jim le v pomoč pri delu in jih ne morejo nadomestiti.

---

33 Vrednost orodja ocenjujemo tudi po tem, s katerimi SUPB se zna neposredno povezati.

3.3 Uporaba podatkovnih baz

Kaj je SQL?

Jezik SQL (angl. Structured Query Language) je standardni jezik za delo z relacijskimi podatkovnimi bazami, kar pomeni, da ga razumejo vsi pomembnejši SUPB. Ime je pravzaprav ponesrečeno, saj omogoča mnogo več kot le poizvedovanje (angl. querying). S stavki tega jezika lahko:

- definiramo tabele (strukturo, tipe atributov ...),
- spreminjamo definicijo tabel,
- brišemo tabele,
- vnašamo podatke v tabele,
- brišemo vrstice tabel,
- spreminjamo (posodabljamo) podatke v tabelah,
- poizvedujemo in
- še mnogo drugega.

Iz seznama je očitno, da sodi jezik SQL tako v skupino jezikov za definiranje podatkov (DDL), kot tudi v skupino jezikov za manipulacijo podatkov (DML).

Jezik SQL je standardiziran pri organizaciji ANSI (American National Standards Institute) in pri ISO (International Standard Organization). Prva inačica jezika SQL je bila standardizirana leta 1986, tretja inačica SQL 3 pa prinaša nekatere novosti, ki v relacijske podatkovne baze že vključujejo elemente objektnega pristopa. Želja pri razvoju jezika je bila, da bi bil čimbolj podoben naravnemu (seveda angleškemu) jeziku, hkrati pa naj bi bil dovolj strukturiran tudi za računalniško okolje. Zato se je prvotno imenoval SEQUEL (Structured English QUEry Language – strukturiran angleški poizvedovalni jezik).

Mnogi sistemi za upravljanje relacijskih podatkovnih baz ponujajo prijazne grafične uporabniške vmesnike za zgoraj našteta opravila, ki uporabniku omogoča delo, ne da bi poznal jezik SQL.

Zakaj SQL?

Zakaj naj bi torej uporabnik poznal (vsaj osnove) dela z jezikom SQL, če so mu na voljo grafični uporabniški vmesniki, s katerimi lahko z nekaj kliki miške in le malo tipkanja naredi isto kot s stavki jezika SQL, pri čemer mora poznati še njegovo sintakso (slovnična pravila)?

Najpomembnejši razlog je dejstvo, da se uporabniški vmesniki različnih SUPB med seboj razlikujejo. Po drugi strani pa je SQL standard, ne samo na papirju, pač pa tudi *de facto* standard, ki ga uporabljajo praktično vsi sistemi za upravljanje relacijskih podatkovnih baz[34]. Tako lahko nekdo (tudi program), ki pozna SQL, nemoteno prehaja med različnimi relacijskimi SUPB.

Pri zastavljanju poizvedb večinoma tudi ni problem poznavanje jezika SQL (ta vsebuje le malo elementov in se ga hitro naučimo) ali grafičnega okolja. Ponavadi je veliko bolj zahtevno pravilno sestaviti vprašanje, da dobimo odgovor, kot ga pričakujemo in želimo. Vsebinsko je torej poizvedovanje lahko zelo bogato, kljub preprostosti jezika SQL. Logika zastavljanja poizvedb je povsem enaka, ne glede na to, ali uporabljamo jezik SQL ali grafično okolje. Razlika med obema načinoma dela je predvsem v količini tipkanja. Učenje samega jezika je torej obrobnega pomena, poudarek naj bo na vsebini.

Jezik SQL

Kot je znano, moremo in moramo ločiti dve skupini opravil v okviru dela s podatkovnimi bazami:

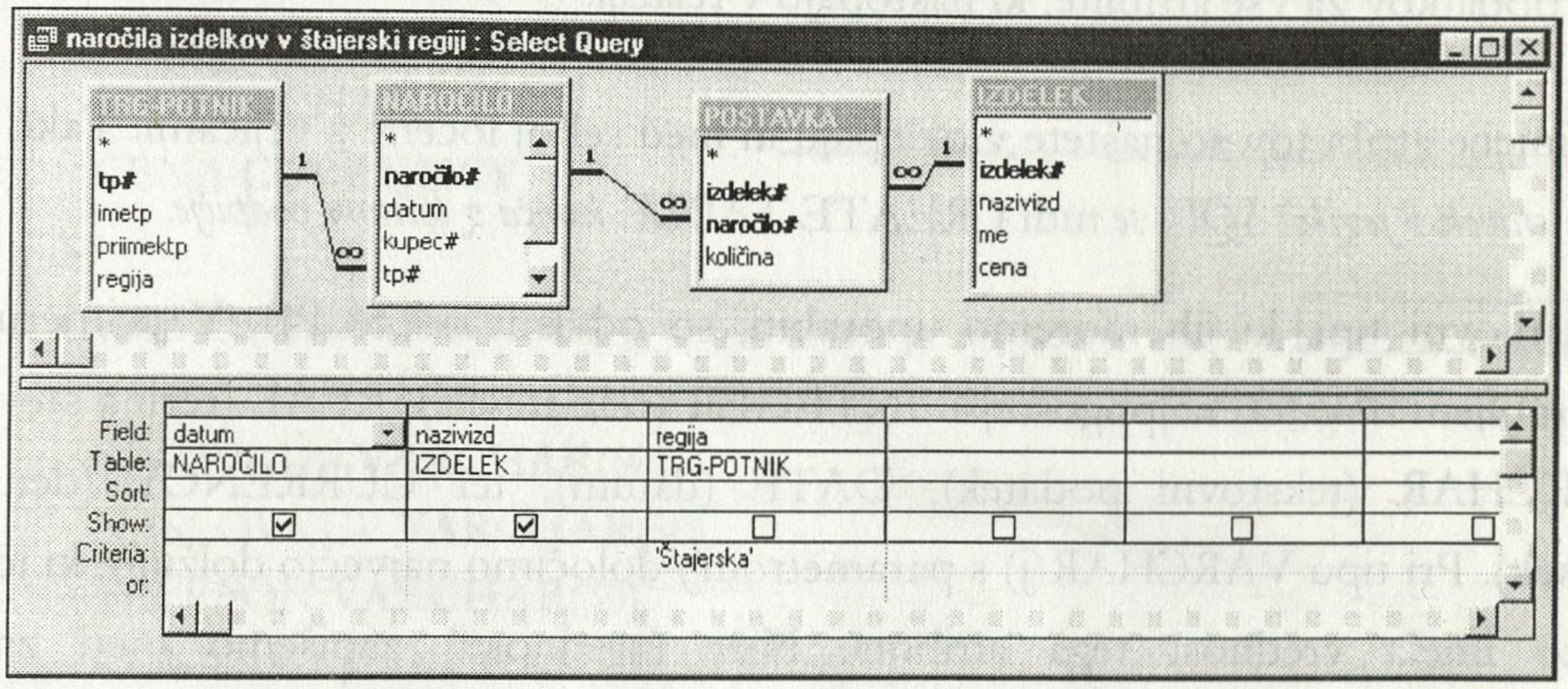
- delo z metapodatki (definiranje baze, sprememba strukture tabel in podobno) ter
- delo s podatki (vnos, posodabljanje, zajemanje – poizvedovanje po bazi).

Za uporabnika je bolj zanimivo slednje, zato bomo stavkom, ki so namenjeni delu s podatki, namenili več pozornosti, posebej še stavku za poizvedovanje po bazi.

---

34 Res pa je, da pozna skoraj vsak SUPB nekatere posebnosti jezika SQL. Stanje lahko primerjamo s slovenskim jezikom, ki ima sicer narečja, pa vendar vsi govorimo isti jezik in se med seboj bolj ali manj razumemo.

Jezik SQL bomo spoznavali predvsem na primerih in se ne bomo spuščali v pretirane podrobnosti[35].




```
SELECT   NAROČILO.NAROČILO#, DATUM, NAZIVIZD
FROM     NAROČILO, TRG-POTNIK, POSTAVKA, IZDELEK
WHERE    NAROČILO.TP# = TRG-POTNIK.TP#
AND      NAROČILO.NAROČILO# = POSTAVKA.NAROČILO#
AND      POSTAVKA.IZDELEK# = IZDELEK.IZDELEK#
AND      REGIJA = 'Štajerska';
```

Slika 40: Primerjava poizvedbe v grafičnem načinu in z jezikom SQL

Definiranje baze

Za ilustracijo dela z jezikom SQL bomo tvorili in v nadaljevanju uporabljali relacijsko podatkovno bazo, katere načrt predstavlja shema R3 iz poglavja Osnove dobrega načrtovanja podatkovnih baz. Ker je shema že v primerni obliki (podvajanje podatkov je odpravljeno), jo lahko takoj uporabimo za definiranje podatkovne baze.

Stavek jezika SQL, ki je namenjen definiranju tabele, je **CREATE TABLE**. Ker z njim vnesemo opis (strukturo) tabele, torej metapodatke, sodi v skupino DDL stavkov jezika SQL.

---

35 Podrobneje lahko zahtevnejši bralec spozna jezik SQL v vsakem učbeniku o podatkovnih bazah; nekoliko več kot tu tudi v (Grad, Jaklič, 1996).

Na sliki 41 so prikazani stavki CREATE TABLE za vseh pet relacij primera naročanja. Opazimo, da moramo za vsako tabelo podati ime ter našteti imena in tipe podatkov za vse atribute, ki nastopajo v relaciji.

Definicije atributov so naštete v oklepaju in med seboj ločene z vejicami. Tako kot *vsak stavek v jeziku SQL, se* tudi CREATE TABLE *konča z ločilom podpičje.*

Podatkovni tipi, ki jih moremo uporabiti, so odvisni od SUPB. V primeru so uporabljeni nekateri najpogostejši: INTEGER (cela števila), REAL (realna števila), VARCHAR (tekstovni podatek), DATE (datum), ter CURRENCY (denarni znesek). Pri tipu VARCHAR(j) s parametrom *j* določimo največjo dolžino, ki jo bo lahko imela vrednost tega atributa. Nize (vrednosti) zapišemo med znaka apostrofa, npr. 'To je niz z dolžino 23.' Pripomnimo, da s tem, ko določimo tip atributa, že v določeni meri definiramo tudi omejitve celovitosti (glej podpoglavje Sistemi za upravljanje podatkovnih baz). Na primer, pri vnosu podatka v datumsko polje bo SUPB preverjal, če je vrednost dopustna. Tako bo, na primer, preprečil vnos vrednosti 31.2.1999, ker ta datum ne obstaja.

Opozoriti velja, da morata biti atributa, ki pomenita isto, a se nahajata v dveh (ali več) tabelah, istega tipa. Na primer, atributa IZDELEK# v tabeli IZDELEK in v tabeli POSTAVKA pomenita isto, zato bosta vsebovala istovrstne podatke in morata biti istega tipa.

```
CREATE TABLE IZDELEK (
  IZDELEK# VARCHAR(7) CONSTRAINT IZDELEKPK PRIMARY KEY,
  NAZIVIZD VARCHAR(50) NOT NULL,
  ME VARCHAR(10),
  CENA CURRENCY
);
CREATE TABLE KUPEC (
  KUPEC# INTEGER CONSTRAINT KUPECPK PRIMARY KEY,
  NAZIVKUP VARCHAR(50),
  NASLOVKUP VARCHAR(50),
  TELEFON VARCHAR(30),
  KONTAKT VARCHAR(50)
);
CREATE TABLE TRG-POTNIK (
  TP# INTEGER CONSTRAINT TPPK PRIMARY KEY,
  IMETP VARCHAR(20),
  PRIIMEKTP VARCHAR(20),
  REGIJA VARCHAR(30)
);
CREATE TABLE NAROČILO (
  NAROČILO# INTEGER CONSTRAINT NARPK PRIMARY KEY,
  DATUM DATE,
  KUPEC# INTEGER,
  TP# INTEGER,
  POPUST INTEGER
);
CREATE TABLE POSTAVKA (
  IZDELEK# VARCHAR(7),
  NAROČILO# INTEGER,
  KOLIČINA REAL,
  CONSTRAINT POSTPK PRIMARY KEY ([IZDELEK#],[NAROČILO#])
);
```

Slika 41: SQL stavki za tvorbo podatkovne baze

Z določilom **NOT NULL** povemo, da mora vsaka vrstica tabele vsebovati vrednost za ta atribut. Drugače povedano, vrednost tega atributa ne sme manjkati za nobeno vrstico tabele. Tako moramo za vsak izdelek pri vnosu navesti poleg šifre še najmanj njegov naziv, lahko pa manjka podatek o ceni (npr. dokler cena še ni določena).

Glavni ključ je ena od omejitev[36], ki jo lahko definiramo na tabeli. Zato jo definiramo v določilu **CONSTRAINT** (omejitev), kjer moramo omejitvi dati neko ime. Na primer, v tabeli IZDELEK smo definirali za atribut IZDELEK# omejitev vrste **PRIMARY KEY** (glavni ključ) z imenom IZDELEKPK. V tabeli postavka, ki ima sestavljen glavni ključ, nismo mogli omejitve vezati na en sam atribut, zato smo jo navedli posebej, v oklepaju pa smo še navedli, kateri atributi sestavljajo glavni ključ.

Če se zgodi, da neke tabele ne potrebujemo več v podatkovni bazi, lahko tabelo (podatke in definicijo) odstranimo s stavkom **DROP TABLE**.

Če bi se, na primer, odločili, da ne želimo več hraniti podatkov o trgovskih potnikih, bi lahko uporabili:

DROP TABLE TRG-POTNIK;

Lahko pa tudi kasneje dodamo atribut (stolpec) neki tabeli. To storimo s stavkom **ALTER TABLE**. Na primer:

ALTER TABLE TRG-POTNIK **ADD** TELEFON VARCHAR(30);

Ta stavek doda tabeli TRG-POTNIK stolpec, kjer hranimo telefonske številke trgovskih potnikov.

Vnos podatkov

Ko smo definirali tabele, je vse pripravljeno za vnos podatkov. S stavkom **INSERT** dodajamo v tabelo nove vrstice. Najpreprostejša oblika stavka je

---

36 Že iz poglavja Relacijski model podatkov vemo, da vrednost tega atributa ne sme biti NULL in da dve vrstici ne moreta imeti enake vrednosti.

```
INSERT INTO   TRG-POTNIK
VALUES        (7,'Tine','Cirman','Gorenjska');
```

S tem stavkom smo v tabelo KUPEC vnesli podatke o enem kupcu, to je eno vrstico. Pri tem moramo vrednosti atributov navesti v istem vrstnem redu, v katerem so bili atributi našteti v definiciji tabele. Poglejmo še en primer, kjer en podatek (atribut) manjka. Na mestu, kjer bi morali zapisati vrednost atributa, zapišemo vrednost NULL:

```
INSERT INTO   KUPEC
VALUES        (88,'Cepki',NULL,'066-33-22',NULL);
```

Naslednja oblika stavka INSERT pa omogoča vnos vrednosti samo nekaterih atributov in to v poljubnem vrstnem redu, ki ga določimo v samem stavku INSERT. Na primer,

```
INSERT INTO   KUPEC (KUPEC#, NASLOVKUP, NAZIVKUP)
VALUES        (89,'Prva 1, Zg. Kašelj','AA d.o.o.');
```

Atributi, katerih vrednosti ne navedemo, dobijo vrednost NULL. Če poskusimo vnesti vrstico brez vrednosti atributa, za katerega je bilo pri definiciji uporabljeno določilo NOT NULL, bo tak poskus zavrnjen. Tako nam bo po vnosu stavka

```
INSERT INTO   IZDELEK (IZDELEK#,ME,CENA)
VALUES        ('746-554','kg',300);
```

javljena napaka, saj je bilo za naziv izdelka (atribut NAZIVIZD) uporabljeno določilo NOT NULL.

Za nadaljnjo uporabo predpostavimo, da smo z uporabo ustreznih stavkov INSERT v tabele, ki smo jih definirali s stavki CREATE, vnesli nekaj podatkov (glej sliko 39).

Brisanje vrstic

S stavkom **DELETE** brišemo eno ali več vrstic iz tabele. Oblika stavka je naslednja:

```
DELETE FROM   <ime tabele>
WHERE         <pogoj>
```

Odstranijo se vrstice, ki ustrezajo pogoju navedenem v **WHERE** določilu.

Če bi prenehali prodajati olje, bi podatke o tem izdelku izbrisali iz tabele s stavkom

```
DELETE FROM   IZDELEK
WHERE         IZDELEK# = '776-444';
```

Če določilo WHERE izpustimo, se stavek nanaša na vse vrstice (pogoj je vedno izpolnjen).

Spreminjanje vrednosti

S stavkom **UPDATE** lahko spremenimo vrednosti atributov za eno ali več vrstic hkrati. Sestavljen je iz treh določil. V določilu **UPDATE** povemo, za katero tabelo želimo spreminjati vrednosti. V določilu **SET** navedemo atribute in njihove nove vrednosti v obliki

<ime atributa> = <vrednost>

in te prireditvene stavke med seboj ločimo z vejicami. V določilu **WHERE** pa povemo pogoje, ki določijo, za katere vrstice naj se spremenijo vrednosti atributov iz določila **SET**.

Oglejmo si dva primera. S stavkom

```
UPDATE      IZDELEK
SET         CENA = CENA*0.9
WHERE       ME = 'kg';
```

za 10 % pocenimo vse izdelke, ki se prodajajo na kilograme.

Če želimo razširiti prodajo tudi na področje notranjske regije, ki do sedaj ni bila pokrita, in bomo za to regijo zadolžili trgovske potnike, ki delajo v primorski regiji, potem uporabimo stavek:

```
UPDATE      TRG-POTNIK
SET         REGIJA = 'Primorska in Notranjska'
WHERE       REGIJA = 'Primorska';
```

Poizvedovanje

S **poizvedbo** (angl. query) izberemo iz podatkovne baze želene podatke. Za uporabnika je poznavanje možnosti poizvedovanja zagotovo najbolj zanimivo in uporabno, saj za zajem (vnos) podatkov in njihovo ažuriranje v večini primerov izdelamo ustrezne uporabniške vmesnike – programe. Vseh poizvedb pa ponavadi ni mogoče vnaprej predvideti in izdelati ustreznih programov. Iz tega sledi potreba po ad hoc poizvedbah, ki je posebej značilna za višje nivoje upravljanja v podjetju.

V relacijskih podatkovnih bazah in s tem pri poizvedovanju z uporabo jezika SQL je rezultat poizvedbe vedno neka tabela (relacija). V jeziku SQL je poizvedovanju namenjen stavek **SELECT**, katerega splošna oblika je najbolj zapletena izmed vseh stavkov tega jezika.

Najpreprostejša oblika stavka SELECT ima obliko

```
SELECT    <seznam atributov>
FROM      <ime tabele>
WHERE     <pogoj>;
```

S tem stavkom iz tabele, katere ime navedemo v določilu **FROM**, izpišemo podatke iz tistih stolpcev, ki jih navedemo v določilu SELECT, in tistih vrstic, ki ustrezajo pogoju, navedenem v določilu **WHERE**. Imena atributov (seveda morajo biti iz izbrane tabele), navedenih v določilu SELECT, morajo biti ločena z vejico.

Na primer,

```
SELECT    IMETP, PRIIMEKTP
FROM      TRG-POTNIK
WHERE     REGIJA = 'Gorenjska';
```

nam izpiše imena in priimke trgovskih potnikov gorenjske regije:

imetp	priimektp
Petra	Cirman
Tine	Cirman

Stolpci so razporejeni, kot določimo v določilu SELECT. Na vrhu tabele se vedno izpišejo tudi imena atributov.

Če spustimo pogoj, potem se izpišejo vse vrstice tabele. Če pa namesto seznama atributov zapišemo znak *, se izpišejo vsi atributi v vrstnem redu, ki je bil uporabljen pri definiranju tabele.

Poizvedba

```
SELECT    *
FROM      KUPEC;
```

povzroči naslednji izpis:

kupec#	nazivkup	naslovkup	telefon	kontakt
12	Atlas d.d.	Mirna 2, Lj	(061)11-22	Janez
13	Janez Aleš	Borbena 223	332-11	Peter, Miro
24	Credo d.o.o.	Pot 12, Mb	(062)33-11	
34	Biro s.p.	Živa 39	632-33	Ana Mohorko
88	Cepki		066-33-22	
89	AA d.o.o.	Prva 1, Zg. Kašelj		
198	Argonavti	Čista 92a	112-112	Sonja Kovač

Za numerične atribute lahko poleg enačaja v pogoju uporabimo tudi neenačaj:

```
SELECT    NAZIVIZD
FROM      IZDELEK
WHERE     CENA < 1000;
```

izpiše nazive izdelkov, ki so cenejši od 1000 tolarjev:

nazivizd
Sol morska
Kisle kumarice 500 g
Jabolčni sok - šesterček

Z uporabo operatorja LIKE primerjamo vrednosti tekstovnih atributov z nekim vzorcem. S stavkom

```
SELECT    NAZIVKUP, NASLOVKUP
FROM      KUPEC
WHERE     NAZIVKUP LIKE 'A%';
```

zahtevamo izpis nazivov in naslovov kupcev, katerih naziv se začne s črko A:

nazivkup	naslovkup
Atlas d.d.	Mirna 2, Lj
Argonavti	Čista 92a
AA d.o.o.	Prva 1, Zg. Kašelj

V vzorcu lahko uporabimo znak %, znak za podčrtavanje _ ter znake, ki so tudi sicer dovoljeni v takih atributih. Znaka % in _ imata naslednji pomen:

%	Označuje niz nič ali več (katerihkoli) znakov.
_	Označuje natanko en (katerikoli) znak.

Tako, na primer, stavek

```
SELECT   IMETP, PRIIMEKTP, REGIJA
FROM     TRG-POTNIK
WHERE    PRIIMEKTP LIKE '%a_';
```

izbere trgovske potnike, katerih predzadnja črka priimka je *a*:

imetp	priimektp	regija
Petra	Cirman	Gorenjska
Jože	Novak	Ljubljanska
Tine	Cirman	Gorenjska

Iskalni pogoj v določilu WHERE je lahko enostaven, kar že poznamo, ali pa sestavljen. Pri sestavljenem pogoju uporabimo več enostavnih pogojev, da sestavimo kompleksnejši pogoj. Če imamo dva enostavna pogoja <pogoj1> in <pogoj2>, potem lahko sestavimo naslednje pogoje:

<pogoj1> **AND** <pogoj2>	Logični IN.	Izbrane bodo vse vrstice, ki izpolnjujejo tako <pogoj1> kot <pogoj2>.
<pogoj1> **OR** <pogoj2>	Logični (inkluzivni) ALI.	Izbrane bodo vse vrstice, ki izpolnjujejo ali <pogoj1> ali <pogoj2> (ali oba).
NOT <pogoj1>	Logična negacija.	Izbrane bodo vse vrstice, ki ne izpolnjujejo pogoja <pogoj1>.

Z več enostavnimi pogoji lahko torej ob uporabi logičnih operatorjev AND, OR in NOT sestavimo še zahtevnejše pogoje.

Nekaj primerov:

```
SELECT    IZDELEK#, NAZIVIZD
FROM      IZDELEK
WHERE     ME = 'kg' AND CENA > 1000;
```

izdelek#	nazivizd
753-443	Sir Gauda

Poizvedba

```
SELECT      NAZIVIZD
FROM        IZDELEK
WHERE       CENA > 100 AND NOT ME = 'kos';
```

izpiše nazive izdelkov, ki so dražji od 100 tolarjev na enoto in katerih merska enota ni kos.

nazivizd
Sir Gauda
Jabolčni sok - šesterček

Namesto stavka

```
SELECT    NAZIVIZD, ME
FROM      IZDELEK
WHERE     CENA >= 100 AND CENA <= 1000;
```

ki izpiše nazive in merske enote za izdelke, katerih cena je med 100 in 1000, lahko zapišemo tudi:

```
SELECT    NAZIVIZD, ME
FROM      IZDELEK
WHERE     CENA BETWEEN 100 AND 1000;
```

Ta zapis določila WHERE zahteva isto, namreč da je cena med (BETWEEN) 40000 in (AND) 80000.

Do sedaj smo v vseh primerih izbirali podatke (hkrati) iz ene same tabele. S stavkom SELECT pa lahko izbiramo tudi podatke **iz več tabel** hkrati.

```
SELECT    <seznam atributov>
FROM      <seznam tabel>
WHERE     <pogoj>;
```

Tudi v seznamu tabel morajo biti imena tabel ločena z vejico.

V določilu WHERE navedemo poleg iskalnih pogojev tudi povezovalne pogoje, ki povedo, kako so tabele med seboj povezane. Z drugimi besedami, s povezovalnim pogojem povemo, katera dva atributa iz nekega para tabel, navedenih v FROM določilu, sta skupna obema tabelama.

Oglejmo si primer. Za vsako naročilo želimo izpisati njegovo šifro, datum naročila ter naziv in naslov kupca. Očitno se podatki nahajajo v dveh tabelah (NAROČILO in KUPCI). Če bi zapisali poizvedbo

```
SELECT    NAROČILO#, DATUM, NAZIVKUP, NASLOVKUP
FROM      NAROČILO, KUPEC
```

bi dobili odgovor (prikazan je le del tabele)

NAROČILO	DATUM	NAZIVKUP	NASLOVKUP
423	9.7.99	Atlas d.d.	Mirna 2, Lj
424	9.7.99	Atlas d.d.	Mirna 2, Lj
425	12.7.99	Atlas d.d.	Mirna 2, Lj
426	12.7.99	Atlas d.d.	Mirna 2, Lj
429	13.7.99	Atlas d.d.	Mirna 2, Lj
423	9.7.99	Biro s.p.	Živa 39
424	9.7.99	Biro s.p.	Živa 39
425	12.7.99	Biro s.p.	Živa 39
426	12.7.99	Biro s.p.	Živa 39
429	13.7.99	Biro s.p.	Živa 39
423	9.7.99	Credo d.o.o.	Pot 12, Mb
424	9.7.99	Credo d.o.o.	Pot 12, Mb
425	12.7.99	Credo d.o.o.	Pot 12, Mb
426	12.7.99	Credo d.o.o.	Pot 12, Mb
...	...	...	...

Odgovor so očitno izbrani stolpci za kartezični produkt dveh tabel, kar ni tisto, kar bi si želeli. Razlog je v tem, da nismo povedali, kako sta tabeli povezani, oziroma, da stolpec KUPEC# v tabeli NAROČILO pomeni isto kot stolpec KUPEC# v tabeli KUPCI.

Stavek SELECT dopolnimo s povezovalnim pogojem

```
SELECT    NAROČILO#, DATUM, NAZIVKUP, NASLOVKUP
FROM      NAROČILO, KUPEC
WHERE     NAROČILO.KUPEC# = KUPEC.KUPEC#
```

in dobimo

naročilo#	datum	nazivkup	naslovkup
429	13.7.99	Janez Aleš	Borbena 223
423	9.7.99	Credo d.o.o.	Pot 12, Mb
425	12.7.99	Credo d.o.o.	Pot 12, Mb
426	12.7.99	Biro s.p.	Živa 39
424	9.7.99	Argonavti	Čista 92a

kar je pravilno. Iskalnega pogoja v tem stavku nismo imeli.

Ker se isto ime atributa (KUPEC#) pojavi v dveh tabelah, moramo, da ne bi prišlo do dvoumnosti, ime atributa razširiti še z imenom tabele, iz katere prihaja. Ime tabele zapišemo pred imenom atributa in ju povežemo s piko

<ime tabele>.<ime atributa>

Na ta način lahko natančneje opredelimo atribut vedno, obvezno pa je le takrat, kadar v istem stavku uporabljamo dva atributa z enakim imenom iz dveh tabel. Opozorimo še na to, da iz enakih imen ne moremo sklepati na isti pomen atributov in obratno, v dveh tabelah imamo lahko atributa z različnima imenoma, ki pa pomenita isto. Tako bi lahko v relacijski shemi, ki jo uporabljamo, poimenovali atribute tudi takole.

NAROČILO (ŠIFRA DATUM, KUPEC#, TP#, POPUST)
IZDELEK (ŠIFRA, NAZIVIZD, ME, CENA)
POSTAVKA (ŠIFRAIZD, ŠIFRANAR, KOLIČINA)
KUPEC (ŠIFRA, NAZIVKUP, NASLOVKUP, TELEFON, KONTAKT)
TRG-POTNIK (ŠIFRA, IMETP, PRIIMEKTP, REGIJA)

Očitno je, da imajo atributi z imenom ŠIFRA v različnih tabelah različen pomen. Prav tako pa ima, na primer, atribut TP# isti pomen kot atribut ŠIFRA v tabeli TRG-POTNIK.

Poglejmo zdaj še primer, ko poleg povezovalnih pogojev uporabimo še iskalne pogoje. Za vsa naročila trgovskih potnikov iz gorenjske regije želimo izpisati, datum naročila ter kateri izdelki (nazivi) so bili naročeni.

```
SELECT   NAROČILO.NAROČILO#, DATUM, NAZIVIZD
FROM     NAROČILO, TRG-POTNIK, POSTAVKA, IZDELEK
WHERE    NAROČILO.TP# = TRG-POTNIK.TP#
AND      NAROČILO.NAROČILO# = POSTAVKA.NAROČILO#
AND      POSTAVKA.IZDELEK# = IZDELEK.IZDELEK#
AND      REGIJA = 'Štajerska';
```

Čeprav v izpisu nastopajo le atributi iz tabel NAROČILO in IZDELEK, pa v stavku potrebujemo tudi tabelo TRG-POTNIK, saj v njej preverjamo, iz katere

regije je trgovski potnik, in tabelo POSTAVKA, saj je le v njej informacija, kateri izdelki se nahajajo na posameznem naročilu[37].

naročilo#	datum	nazivizd
423	9.7.99	Sol morska
423	9.7.99	Kisle kumarice 500 g
425	12.7.99	Kisle kumarice 500 g

V splošnem so rezultat poizvedbe vse vrstice, ki ustrezajo iskalnim in povezovalnim pogojem, tudi če so vrednosti vseh atributov iz določila SELECT za nekatere pare vrstic enake. Če želimo izpisati vse regije, ki jih pokrivajo trgovski potniki, to lahko storimo s stavkom

SELECT REGIJA
FROM TRG-POTNIK;

ki pa nam izpiše regije za vse trgovske potnike, torej se nekatere regije ponavljajo. Če se želimo izogniti podvajanjem, to lahko dosežemo z uporabo določila DISTINCT. Za zgornji primer torej zapišemo stavek

SELECT DISTINCT REGIJA
FROM TRG-POTNIK;

Poglejmo si, kakšen je izpis v enem in kakšen v drugem primeru:

brez uporabe dol. DISTINCT

regija
Štajerska
Gorenjska
Primorska in Notranjska
Dolenjska
Ljubljanska
Gorenjska

z uporabo določila DISTINCT

regija
Dolenjska
Gorenjska
Ljubljanska
Primorska in Notranjska
Štajerska

Opozorimo pa naj, da lahko uporaba določila DISTINCT precej podaljša poizvedovanje, posebej pri večjih tabelah in zahtevnejših poizvedbah.

---

37 Bralcu, ki ima težave z razumevanjem tega, da potrebujemo za odgovor tudi tabelo POSTAVKA, priporočamo, da se postavi v vlogo SUPB, si pogleda le podatke iz ostalih treh tabel in skuša odgovoriti na postavljeno vprašanje.

V aplikacijah pogosto potrebujemo **agregatne (sumarne) funkcije**, katerih rezultat je zbirna informacija o določeni skupini vrstic. Primer sumarne poizvedbe je, kolikšna je bila do sedaj največja količina naročenega (določenega) izdelka, skupna vrednost naročil nekega kupca ... SQL ima vgrajene naslednje agregatne funkcije: **COUNT** (število), **SUM** (vsota), **MAX** (največja vrednost), **MIN** (najmanjša vrednost) in **AVG** (povprečna vrednost). Agregatno funkcijo uporabimo tako, da v SELECT določilu poizvedbe zapišemo ime agregatne funkcije, v oklepaju pa kot parameter navedemo ime atributa, za katerega želimo izračunati vrednost te agregatne funkcije. Na primer,

```
SELECT   AVG(KOLIČINA) AS POVP, MAX(KOLIČINA) AS MAX
FROM     POSTAVKA
WHERE    IZDELEK# = '321-432';
```

povp	max
20	33

nam vrne povprečno in največjo naročeno količino izdelka s šifro 321-432. Z določilom **AS** smo opredelili imena stolpcev izračunanih vrednosti pri izpisu.

Če želimo zvedeti, koliko (funkcija COUNT) naročil smo dobili od podjetja Credo d.o.o., potem poizvedbo zapišemo takole:

```
SELECT   COUNT(*) AS ŠTEVILO
FROM     NAROČILO, KUPEC
WHERE    NAROČILO.KUPEC# = KUPEC.KUPEC#
         AND NAZIVKUP = 'Credo d.o.o.';
```

število
2

Tu znak zvezdica (*) pomeni, da štejemo vrstice tabele, ki je rezultat povezovalnih in iskalnih pogojev. Namesto zvezdice bi lahko zapisali tudi, na primer, ime atributa NAROČILO#.

Naslednja možnost, ki jo ponuja SQL, je **grupiranje** ali združevanje vrstic v skupine na osnovi enakih vrednosti nekega atributa. Potem lahko za vsako od teh skupin vprašamo po vrednosti agregatne funkcije. Poizvedba

```
SELECT    REGIJA, COUNT(*) AS ŠTEVILO
FROM      TRG-POTNIK
GROUP BY  REGIJA;
```

regija	število
Dolenjska	1
Gorenjska	2
Ljubljanska	1
Primorska in Notranjska	1
Štajerska	1

nam izpiše število trgovskih potnikov za vsako regijo. Določilo poizvedbe GROUP BY pove na osnovi vrednosti katerega atributa, naj se vrstice grupirajo. V konkretnem primeru sodijo v isto skupino vsi trgovski potniki, ki imajo enako vrednost atributa REGIJA, torej ki delajo v isti regiji.

V določilu poizvedbe SELECT vedno zapišemo le imena tistih atributov, po katerih grupiramo, ter sumarne funkcije, ki naj se izpišejo za vsako grupo.

Pogosto je koristna tudi možnost uporabe aritmetičnih operatorjev:

+	seštevanje,
-	odštevanje,
*	množenje ter
/	deljenje.

S stavkom

```
SELECT    NAZIVIZD, CENA*0.1
FROM      IZDELEK;
```

za vsak izdelek izpišemo, za koliko tolarjev bi se podražil, če bi vse izdelke podražili za 10 %.

Če zberemo dosedanje znanje, lahko zapišemo nekoliko bolj zapleteno poizvedbo, s katero želimo za vsako naročilo izpis šifre in skupne vrednosti naročila. Podatke, ki jih potrebujemo, najdemo v treh tabelah: NAROČILO, IZDELEK in POSTAVKA. Ker nas zanimajo sumarni podatki (vrednost naročila) glede na naročila, bomo združevali po atributu NAROČILO#.

```
SELECT      NAROČILO.NAROČILO#,
            SUM(CENA*(100-POPUST)/100*KOLIČINA) AS ZNESEK
FROM        NAROČILO, IZDELEK, POSTAVKA
WHERE       NAROČILO.NAROČILO# = POSTAVKA.NAROČILO#
AND         POSTAVKA.IZDELEK# = IZDELEK.IZDELEK#
GROUP BY    NAROČILO.NAROČILO#;
```

naročilo#	znesek
423	
424	317,52
425	4992
426	144593,535

Zahtevnejšega bralca bo utegnilo zanimati, zakaj ni izpisan znesek za naročilo številka 423. Odgovor mu bo razkril pogled na podatke v tabeli naročila (glej sliko 39) in preprost razmislek.

Vrstice se praviloma izpišejo v vrstnem redu, v kakršnem so shranjene v tabeli. Mogoč pa je tudi izpis, pri katerem so vrstice urejene po **naraščajočem** (določilo **ASC**) ali **padajočem vrstnem redu** (določilo **DESC**) vrednosti nekega atributa. Če želimo tak izpis, dodamo še **ORDER BY** določilo poizvedbe, kjer navedemo ime atributa, po katerem naj bodo vrstice urejene:

```
SELECT      NAZIVIZD, ME, CENA
FROM        IZDELEK
WHERE       NOT ME = 'kos'
ORDER BY    CENA ASC;
```

NAZIVIZD	ME	CENA
Sol morska	kg	44,10 SIT
Jabolčni sok - šesterček	paket	480,00 SIT
Sir Gauda	kg	1.156,50 SIT

oziroma

```
SELECT      NAZIVIZD, ME, CENA
FROM        IZDELEK
WHERE       NOT ME = 'kos'
ORDER BY    CENA DESC;
```

nazivizd	me	cena
Sir Gauda	kg	1.156,50 SIT
Jabolčni sok - šesterček	paket	480,00 SIT
Sol morska	kg	44,10 SIT

Razširitve jezika SQL

Videli smo, da je uporaba jezika SQL precej intuitivna in enostavna. Že z osnovno obliko stavka SELECT lahko poiščemo odgovor na marsikatero vprašanje. Žal pa gredo na račun enostavne uporabe omejene možnosti poizvedovanja, saj s stavkom SELECT ni mogoče izraziti vseh poizvedb, ki bi nas utegnile zanimati. Zato ima mnogo SUPB možnost kombiniranja jezika SQL s postopkovnimi jeziki 3. generacije (npr. Java, Visual Basic, C++ in podobno) ali pa je kar jezik SQL razširjen s postopkovnimi konstrukti, kot so prireditveni stavki, pogojno izvajanje dela kode (IF), ponavljanje (FOR) in podobno.

Izmenjava podatkov

Poudarili smo že (glej podpoglavje Opisovanje podatkov) pomen, ki ga ima opisovanje podatkov pri izmenjavi le-teh med različnimi sistemi. Pri tem gre lahko za sisteme znotraj organizacije, najpogosteje pa za izmenjavo podatkov pri medorganizacijskem poslovanju.

V primeru, ko si organizacije med seboj izmenjujejo papirne dokumente, je običajno, da gre za obrazce, na katerih se nahajajo opisi podatkov poleg prostora, namenjenega za vpis le-teh. Podobno moramo pri računalniški izmenjavi dokumentov poskrbeti za opise podatkov, sicer sistem, ki dobi podatke, teh ne more pravilno interpretirati. Na primer, če posredujemo partnerski organizaciji naročilo v obliki

423,9.7.99,123-456,44,10,10,321-432,kos,192,00,1,3

bo program, ki obdeluje naročila, razbral pomen posameznega podatka le, če smo se predhodno *natančno* dogovorili o strukturi podatkov o naročilih. Težave narastejo s številom partnerjev: morda se bomo z vsakim partnerjem posebej dogovorili o strukturi podatkov za naročila, kar pomeni za nas obilo dela, če pa je

naša organizacija dovolj močna, bomo vsem partnerjem vsilili enako strukturo podatkov.

V izogib takšnim težavam so se v preteklosti uveljavili standardi EDI, s katerimi so bile na enoten način opredeljene strukture posameznih vrst dokumentov, danes pa se uveljavlja mnogo bolj fleksibilen standard XML[38] (angl. Extensible Markup Language), ki omogoča opis podatkov. Prednosti jezika XML se posebej pokažejo pri izmenjavi podatkov med sistemi z različnimi operacijskimi sistemi in strojno opremo.

V jeziku XML z značkami opišemo pomen posameznega podatkovnega elementa ali skupine elementov. Značko, ki označuje začetek podatkovnega elementa, tvori ime elementa, zapisano med znakoma < in >, konec elementa pa označimo na enak način, le da pred ime (vendar za znakom <) postavimo še znak /.

Tako bi lahko podatke o naročilu, na primer, opisali takole:

```
<naročilo>
        <šifranaročila>423</šifranaročila>
        <naročnik>
                <šifranaročnika>24</šifranaročila>
                <naročnik>Credo d.o.o.</naročnik>
        </naročnik>
        <postavka>
                <izdelek>
                        <šifraizdelka>123-456</šifraizdelka>
                        <nazivizdelka>Sol morska<nazivizdelka>
                </izdelek>
                <količina>10</količina>
        </postavka>
        <postavka>
                <izdelek>
                        <šifraizdelka>321-432</šifraizdelka>
                        <nazivizdelka>Kisle kumarice 500 g<nazivizdelka>
                </izdelek>
                <količina>1</količina>
        </postavka>
</naročilo>
```

38 XML je pravzaprav metajezk, to je jezik, s katerim opisujemo jezike za opis dokumentov, grafike, finančnih informacij in podobno.

Vprašanja za preverjanje razumevanja

- Kaj so podatkovne baze in kakšne so njihove značilnosti?
- Kakšen je odnos med različnimi vrstami podatkovnih virov, ki ste jih spoznali v prvem poglavju (operativni, podatkovna skladišča...) in podatkovnimi bazami?
- Kakšne so naloge skrbnika podatkovne baze?
- Za kaj je pri podatkovnih bazah odgovoren uporabnik?
- Poiščite čimveč spletnih strani, ki predstavljajo sisteme za upravljanje podatkovnih baz, in si oglejte značilnosti le-teh. Kateri bi bili po vašem mnenju lahko kriteriji, po katerih bi izbrali ustrezen sistem za upravljanje podatkovnih baz?
- Razložite vse tri vrste nadzora celovitosti na primerih, ki jih poznate.
- V MS Accessu izdelajte podatkovno bazo, v kateri boste beležili podatke o izposoji CD plošč iz vaše domače zbirke. Definirajte glavne ključe. Razmislite, ali ste bazo načrtovali tako, da ne bo prihajalo do težav pri delu (anomalij).
- V podatkovni bazi iz prejšnje naloge poiščite odgovore na vprašanja:
  Izpišite seznam vseh plošč slovenskih izvajalcev.
  Za vsakega prijatelja ugotovite, koliko plošč si je izposodil.
  Katerih plošč še niste posodili nikomur?
- Katere dimenzije in mere menite, da bi bile potrebne v večdimenzionalni podatkovni bazi, namenjeni spremljanju zalog v skladišču?
- Katere so značilnosti jezika XML?

Ključni pojmi

anomalija
celovitost
hierarhični model
jezik SQL
jezik XML
mrežni model
načrtovanje
neodvisnost
objektni model
orodja CASE
podatkovna baza
podatkovni model
podvajanje podatkov
relacijski model
sistem za upravljanje podatkovne baze
skrbnik podatkovne baze
stavek SELECT
večdimenzionalni model
vzdrževanje

4. POGLAVJE

Informacijska podpora managementu

Cilji poglavja

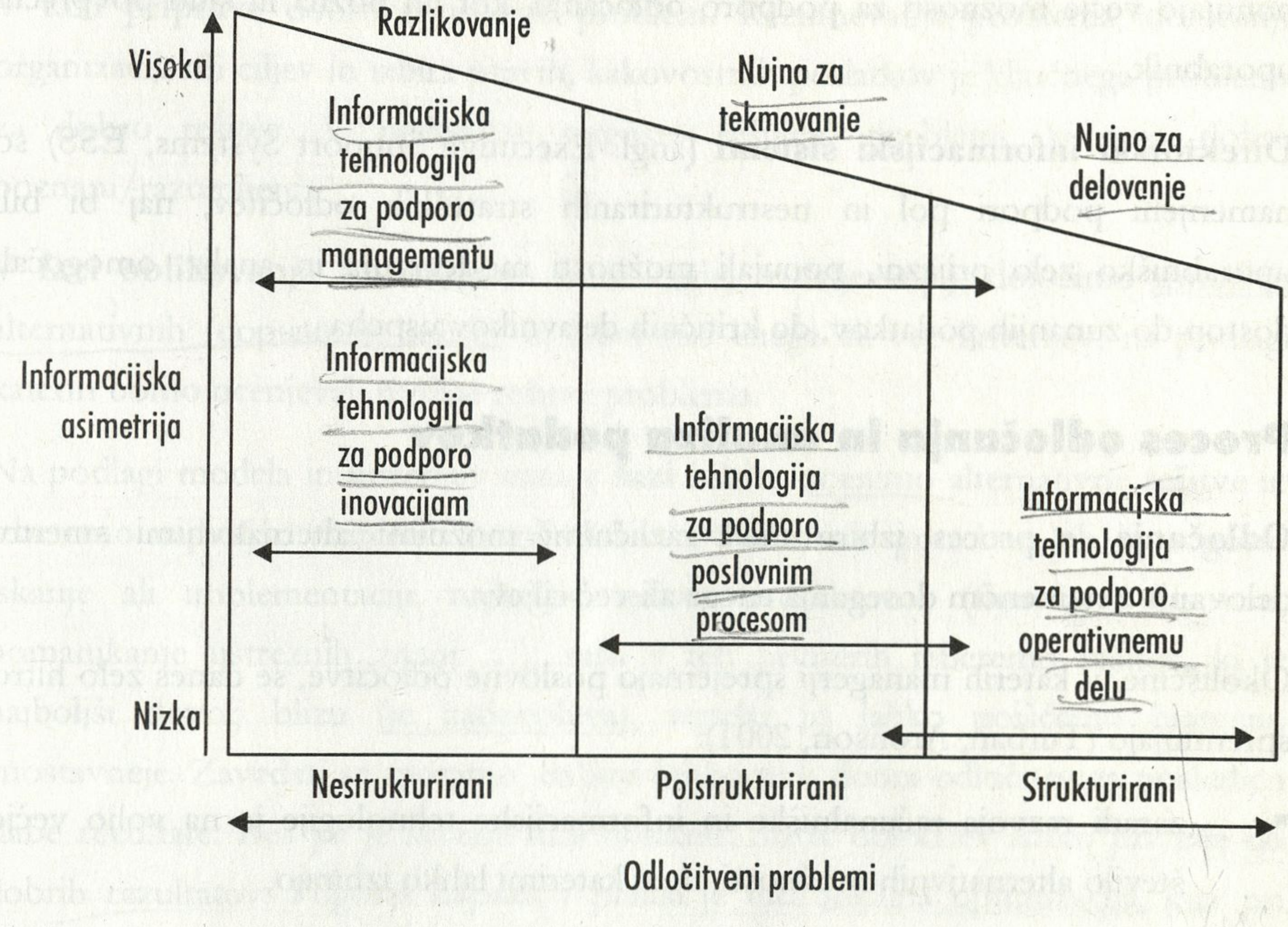
V tem poglavju boste spoznali:

- Uporaba katerih informacijskih tehnologij praviloma bolj prispevaj k dvigu nivoja informacijske asimetrije?
- Katere so faze sistematičnega odločitvenega procesa, katere informacijske tehnologije uporabljamo v posameznih fazah in kakšno vlogo imajo pri poslovnem odločanju modeli?
- Na kakšna vprašanja nam pomagajo iskati odgovore sistemi za podporo odločanju?
- Zakaj je uporaba programskih paketov za delo s preglednicami še vedno najbolj pogosto uporabljeno informacijsko orodje za podporo odločanju?
- Kaj je poslovno obveščanje in kako je povezano z informacijsko demokratizacijo?
- Kaj je sprotna analitična obdelava podatkov, katere podatkovne vire uporabljamo zanjo in kakšne so značilnosti orodij?
- Kaj imata skupnega rudarjenje v podatkih in upravljanje znanja?
- Kako lahko poslovni portali pomagajo reševati problem preobilja informacij, s katerim se danes srečujejo odločevalci.

Managerji morajo danes pri ocenjevanju investicij v informacijsko tehnologijo upoštevati nivo informacijske asimetrije (ta ustreza donosu na vlaganja v informatiko), ki ga sistem lahko zagotovi. Slika 42 prikazuje, kako lahko izbira različnih tehnologij vpliva na nivo informacijske asimetrije. Na desni strani so prikazane investicije v informatizacijo operativnega dela (plače, računovodstvo, finančno poročanje, kartične obdelave ...), ki so nujni za delovanje organizacije. Tu so odločitve najbolj strukturirane in je informacijska podpora zato relativno enostavna, hkrati pa v tem delu dosežemo zelo nizek (ali nikakršen) nivo informacijske asimetrije. V srednjem delu so odločitve manj strukturirane, zato je neposredna uporaba izkušenj drugih (tekmecev) težka. Uspešna informatizacija poslovnih procesov pa je pogoj za uspešno tekmovanje. Primeri informacijske tehnologije, ki se uporabljajo za podporo poslovnim procesom, so celovite programske rešitve (ERP sistemi), sistemi za upravljanje delovnih tokov (angl. Workflow Systems), sistemi za upravljanje zalog ... Na levem delu slike 42 pa so predstavljena vlaganja v tehnologije in uporabo le-teh, ki organizacijam omogočajo razlikovanje. Primeri tovrstnih tehnologij so sistemi za podporo odločanju (angl. Decision Support Systems – DSS), orodja za sprotno analitično obdelavo podatkov, orodja za rudarjenje v podatkih, sistemi za podporo odločanju v skupinah (angl. Group Decision Support Systems), orodja za modeliranje in simulacijo (npr. poslovnih procesov), sistemi za podporo skupinskemu delu (angl. Groupware) in podobno. V tem poglavju se posvečamo tehnologijam, ki omogočajo doseganje višjega nivoja informacijske asimetrije, to so predvsem tehnologije za podporo managementu.

Informacijska podpora managementu pojmujemo kot uporabo tehnologije (samostojnih orodij ali v kombinaciji z drugo informacijsko tehnologijo) za podporo opravilom managementa v splošnem, še posebej pri odločanju (Turban, Aronson, 2001).

Managerski informacijski sistemi (tudi upravljalni informacijski sistemi, angl. Management Information Systems, **MIS**) so nastali kot posledica ugotovitev, da dotedanji operativni sistemi ne ustrezajo managementu. Zagotavljajo predvsem sumarne podatke in prikaz izjem, kar zadostuje predvsem za rutinske, strukturirane odločitve na taktičnem nivoju, kjer so pravila in informacijski tokovi vnaprej znani.



Slika 42: Uporaba tehnologij in informacijska asimetrija (Marchand, Kettinger, Rollins, 2001)

Leta 1971 sta raziskovalca Gorry in Scott Morton ugotavljala, da odločitvene probleme lahko v grobem razdelimo na strukturirane, polstrukturirane in nestrukturirane. Informacijsko so bili (z managerskimi informacijskimi sistemi) podprti predvsem prvi, ni pa bilo ustrezne podpore za manj strukturirane odločitve, ki pa lahko ključno vplivajo na poslovanje. To je vodilo v razvoj **sistemov za podporo odločanju** (angl. Decision Support Systems, **DSS**), ki so namenjeni predvsem podpori polstrukturiranih ad hoc problemov srednjega managementa. Njihov namen ni avtomatizacija odločitvenih procesov ali vsiljevanje rešitev. Po več kot tridesetih letih razvoja obetajočih in naprednih sistemov za podporo odločanju pa je v praksi še vedno najpogosteje uporabljena pojavna oblika možnost kaj-če (angl. what-if) analiz v preglednicah. Uporabi preglednic pri poslovnem odločanju posvečamo v nadaljevanju posebno pozornost tudi zato, ker jih zelo pogosto najdemo v delovnih okoljih managerjev, hkrati pa

ponujajo večje možnosti za podporo odločanju, kot jih pozna in sluti povprečni uporabnik.

Direktorski informacijski sistemi (angl. Executive Support Systems, **ESS**) so namenjeni podpori pol in nestrukturiranih strateških odločitev, naj bi bili uporabniško zelo prijazni, ponujali možnosti modeliranja in analiz, omogočali dostop do zunanjih podatkov, do kritičnih dejavnikov uspeha …

Proces odločanja in analiza podatkov

Odločanje je proces izbire med različnimi možnimi alternativnimi smermi delovanja z namenom doseganja enega ali več ciljev.

Okoliščine, v katerih managerji sprejemajo poslovne odločitve, se danes zelo hitro spreminjajo (Turban, Aronson, 2001):

- zaradi razvoja računalniške in informacijske tehnologije je na voljo večje število alternativnih možnosti, med katerimi lahko izbirajo,
- tekmovalnost se povečuje, kar povzroča večje stroške pri napačnih odločitvah,
- globalizacija, potrošništvo in vladne intervencije povzročajo večjo negotovost glede prihodnjih okoliščin poslovanja,
- hitre spremembe zahtevajo hitrejše odločitve.

Iz naštetih razlogov se *zgolj* pristopi k odločanju, ki so bili v navadi v preteklosti (poskušanje, učenje na napakah, uporaba intuicije), ne obnesejo več. Zato se morajo managerji naučiti uporabljati sistematične pristope ter nove tehnologije in tehnike, ki so danes na voljo.

Sistematičen odločitveni proces je sosledje aktivnosti, ki jih lahko v grobem uvrstimo v 3 faze (glej sliko 43):

- faza priprave,
- faza oblikovanja ter
- faza izbire.

V **fazi priprave** dobro raziščemo problem. Razumevanje problema, določanje organizacijskih ciljev in izbira pravih, kakovostnih podatkov je ključnega problema za dobro rešitev. V praksi se pogosto rešujejo problemi, ki niso dobro poznani/razumljeni.

V **fazi oblikovanja** formuliramo model (glej nadaljevanje), določimo množico alternativnih (dopustnih) rešitev in izberemo enega ali več kriterijev, na podlagi katerih bomo ocenjevali možne rešitve problema.

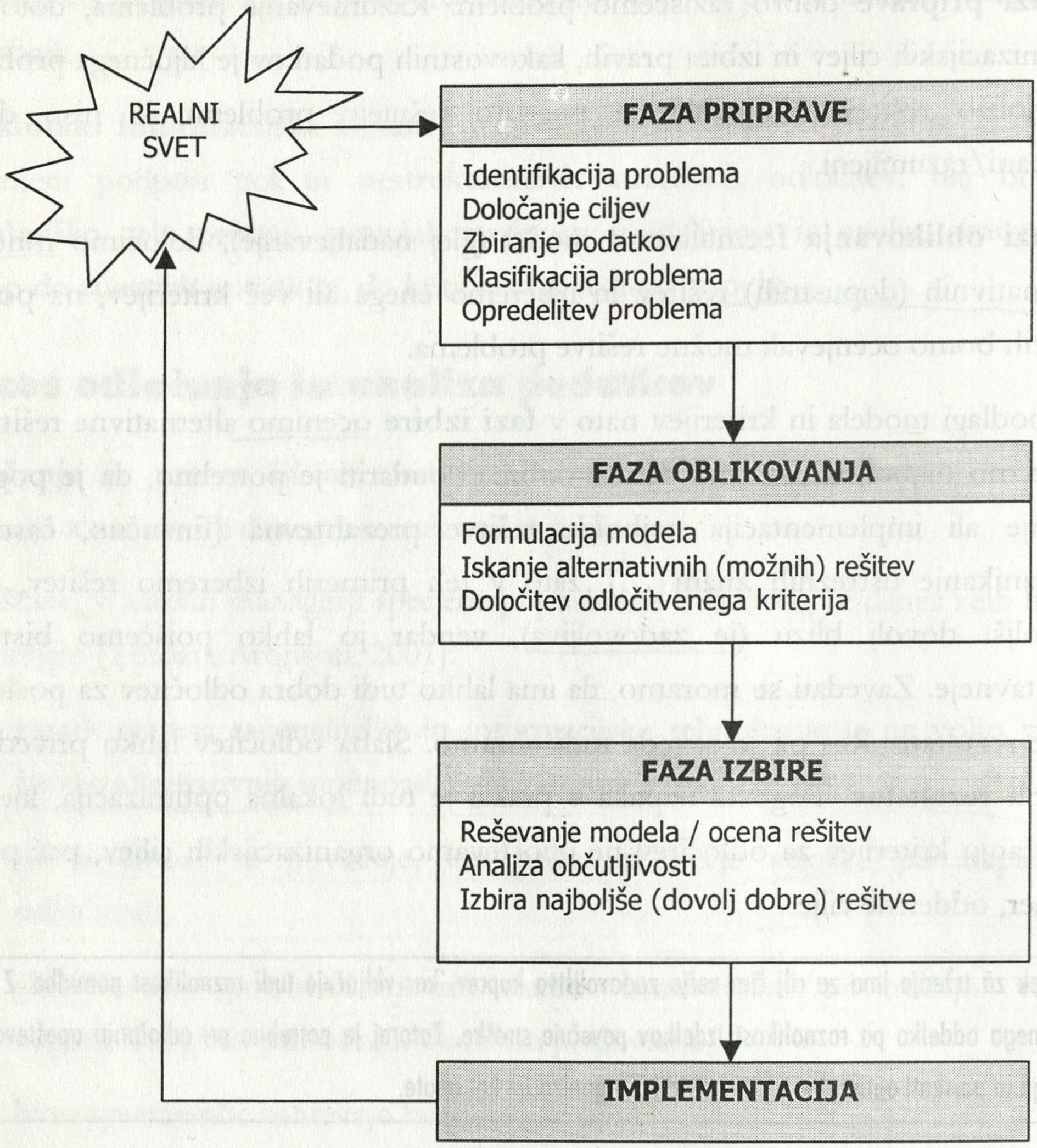
Na podlagi modela in kriterijev nato v **fazi izbire** ocenimo alternativne rešitve in izberemo najboljšo oziroma dovolj dobro. Poudariti je potrebno, da je pogosto iskanje ali implementacija najboljše rešitve prezahtevna (finančno, časovno, pomanjkanje ustreznih znanj ...), zato v teh primerih izberemo rešitev, ki je najboljši dovolj blizu (je zadovoljiva), vendar jo lahko poiščemo bistveno enostavneje. Zavedati se moramo, da ima lahko tudi dobra odločitev za posledico slabe rezultate. Res pa je seveda tudi obratno. Slaba odločitev lahko privede do dobrih rezultatov. Pogosta napaka v praksi je tudi lokalna optimizacija, kjer pri določanju kriterijev za odločitev ne upoštevamo organizacijskih ciljev, pač pa, na primer, oddelčne cilje.

Oddelek za trženje ima za cilj čim večje zadovoljstvo kupcev, kar vključuje tudi raznolikost ponudbe. Z vidika finančnega oddelka pa raznolikost izdelkov povečuje stroške. Zatorej je potrebno pri odločanju upoštevati oba kriterija in poiskati optimalno rešitev z vidika organizacije kot celote.

Sestavni del izbire je analiza občutljivosti, ki pomeni vpliv sprememb v podatkih (vhodni podatki, parametri) na predlagano rešitev. Tovrstna analiza je zelo pomembna predvsem zaradi hitrih sprememb v okolju in omogoča prilagodljivost le-tem, hkrati pa pomeni boljše razumevanje modela.

Praviloma **reševanje problemov** razumemo kot proces odločanja, ki ga sestavljajo zgoraj omenjene tri faze, skupaj s fazo implementacije.

Implementacija pomeni udejanjanje izbrane rešitve. Pomemben del implementacije je tudi merjenje uspešnosti in uporaba izkušenj pri naslednjih poslovnih odločitvah.



Slika 43: Odločitveni proces in reševanje problemov

V literaturi najdemo tudi alternativne modele procesa odločanja (glej Turban, Aronson, 2001), vendar je zgoraj opisani ustrezen za odločitvene probleme, pri katerih uporabljamo matematične modele in si pri tem pomagamo z informacijsko tehnologijo.

Odločitvene probleme lahko *v grobem* razdelimo na strukturirane, polstrukturirane in nestrukturirane. Pri **strukturiranih problemih** so postopki za iskanje najboljših (dovolj dobrih) rešitev znani. Primer strukturiranega problema, kjer poznamo postopek iskanja rešitve, je optimizacija zaloge. Pri **nestrukturiranih problemih** pa je osnova odločanju pogosto kar intuicija. Tovrstni problemi so lahko le deloma

podprti z informacijsko tehnologijo, na primer s tehnologijo za dostop do podatkov, z inteligentnimi ali ekspertnimi sistemi. Primer nestrukturirane odločitve je lahko izbira novega direktorja. **Polstrukturirani problemi** imajo nekatere elemente strukturiranih in nekatere elemente nestrukturiranih problemov.

Problem lahko analiziramo (faza oblikovanja in izbire v odločitvenem procesu) na dva načina: **kvalitativno** ali **kvantitativno**. Pri prvem se opiramo na izkušnje, občutke in je primeren predvsem za manj strukturirane probleme. Tak način odločanja pride v poštev pri relativno enostavnih ali rutinskih problemih. Pri drugem načinu pa odločitev temelji na podatkih, povezanih s problemom. Problem opišemo z matematičnimi izrazi, ki opisujejo cilje, omejitve in povezave med podatki, ki nastopajo v problemu. Z uporabo kvantitativnih metod nato poiščemo rešitev, kar je rezultat analize problema.

Osredotočili se bomo na kvantitativno analizo, pri kateri nam je lahko v veliko pomoč informacijska tehnologija, čeprav se le-ta čedalje pogosteje uporablja tudi pri kvalitativnih metodah (npr. sistemi za podporo skupinskemu delu).

Modeli

Proces kvantitativne analize obsega več korakov:

1. Razviti moramo model, ki opisuje problem.
2. Pripraviti moramo podatke, ki jih potrebujemo v modelu.
3. Poiščemo rešitev modela.
4. Analiziramo občutljivost.
5. Predlagamo rešitev oziroma sestavimo poročilo o naših ugotovitvah.

Model prikazuje realne sisteme na poenostavljen način. Prikazuje le osnovne sestavne dele sistema in povezave med njimi. Navadno uporabljamo tak model, ki prikazuje tiste elemente, ki nas zanimajo v analizi sistema. Uporaba modelov ima vrsto prednosti:

- Zaradi take poenostavitve sistema je analiza le-tega enostavnejša.
- Med prednosti modeliranja lahko štejemo tudi možnost, da simuliramo obnašanje sistema glede na različne odločitve, oziroma različne vrednosti posameznih parametrov, česar v realnem sistemu ne bi mogli.
- S preizkušanjem alternativnih možnosti na modelu ne vplivamo in ne motimo vsakodnevnega delovanja realnega sistema.
- Stroški analize modela so nižji od stroškov preizkušanja alternativ v realnem sistemu.
- Zaradi vse večje negotovosti v poslovanju lahko na modelu analiziramo tudi tveganja pri določenih ravnanjih.
- Modeli povečujejo razumevanje delovanja sistema in učenje.

Modeli so lahko različni. V avtomobilski tovarni pred serijsko izdelavo novega avta izdelajo več modelov, na katerih preizkušajo, kakšno bo obnašanje pravega avta. To je fizični model. Arhitekturni načrti in zemljevidi spadajo med grafične modele. Model je lahko tudi opis z besedami. Velikokrat pa uporabljamo **matematične modele**, na primer

$$\text{dobiček} = \text{prihodki} - \text{stroški}$$

Skupino takih zvez med spremenljivkami, ki opisujejo poslovni sistem ali njegov del, imenujemo tudi poslovni model. V nadaljevanju nas bodo zanimali predvsem taki modeli.

Poslovni modeli

Čeprav se lahko poslovni modeli zelo razlikujejo od primera do primera, jih lahko vendarle razdelimo v nekaj večjih skupin. V literaturi lahko najdemo naslednje večje skupine modelov, ki jih uporabljamo v sistemih za podporo odločanja:

- modeli za napovedovanje,
- modeli vrst,
- modeli za optimizacijo,

- odločitvene tabele in drevesa,
- vodenje projektov,
- vodenje zalog …

Pri napovedovanju želimo glede na znane podatke o pojavu v preteklosti napovedati, kakšen bo pojav v prihodnosti. Za napovedovanje lahko uporabljamo več modelov, med njimi regresijo, drseča povprečja ali glajenje.

Modele vrst lahko uporabljamo povsod, kjer imamo opravka s strežbo strank, na primer v bankah, restavracijah ali pri delitvi računalniških virov. S temi modeli lahko ugotavljamo, kakšen je pričakovan čas čakanja stranke, kakšen je pretok strank (število postreženih na časovno enoto) …

Tretja večja skupina modelov so modeli za optimizacijo. Glede na vrsto problema se lahko odločamo med različnimi modeli (linearni ali nelinearni program, model najkrajše poti, model najkrajšega vpetega drevesa, model maksimalnega pretoka), za vsakega izmed modelov pa poznamo enega ali več algoritmov za razreševanje tega problema. Pogosto alternativ v poslovnih odločitvenih problemih ni mogoče oceniti zgolj z enim preprostim ciljem kot je npr. največji možni dobiček. V tem primeru govorimo o večkriterijskem odločanju, kjer moramo pretvoriti različne, včasih tudi nasprotujoče si, kriterije v eno samo mero. Na primer, managerji morajo ustreči delničarjem, hkrati pa povečati zadovoljstvo zaposlenih s povečanjem njihovih prejemkov. V teh primerih lahko uporabimo sisteme, ki podpirajo večkriterijsko odločanje. Znan je analitični hierarhični postopek (angl. Analytical Hierarchy Process), ki uporablja kvalitativne in kvantitativne kriterije za reševanje večkriterijskih odločitvenih problemov.

Odločitvene tabele in drevesa nam pomagajo pri iskanju najboljše strategije v primerih, ko imamo na voljo več odločitvenih pravil in ko v problemu nastopa nedoločenost (glej podpoglavje Priprava podatkov).

Tudi pri vodenju projektov si lahko pomagamo s kvantitativnimi metodami, na primer PERT in CPM. S temi modeli si pomagamo pri načrtovanju, določanju urnikov in nadzorovanju projektov.

Modele, ki jih uporabljamo pri vodenju zalog, bi lahko šteli kar med optimizacijske modele, saj je njihov cilj zmanjšati (minimizirati) stroške zalog, pri čemer moramo seveda zadovoljiti povpraševanje.

Za konkreten **problem** izberemo torej **ustrezen model**, za katerega imamo lahko potem več **algoritmov za reševanje**. Recimo, da želimo izbrati tako kombinacijo proizvedene količine dveh izdelkov, da bo dobiček karseda velik, ob določenih omejitvah proizvodnih in drugih zmogljivosti. Potem bo morda za ta problem primeren model linearni program. Za reševanje modelov te vrste poznamo več algoritmov, na primer Simplex. To pomeni, da se pri istem modelu lahko odločamo med več algoritmi reševanja. Če bi se v zgornjem primeru odločili, da bomo problem rešili z linearnim programiranjem v Excelu, potem smo s tem izbrali tudi algoritem reševanja, saj nam ta programski paket ponuja le en algoritem za reševanje linearnih programov.

Ideja neodvisnosti podatkov, ki jo poznamo iz podatkovnih baz, je našla svoje mesto tudi na področju sistemov za podporo odločanja. Tako govorimo tukaj o **neodvisnosti med modelom in algoritmom za reševanje**. To pomeni, da za različne modele lahko uporabljamo isti algoritem za reševanje ali za isti model več algoritmov za reševanje. Podobno lahko govorimo o **neodvisnosti med modelom in množico podatkov**. V tem primeru lahko isti model uporabljamo na več množicah podatkov ali obratno.

Priprava podatkov

Modeli praviloma zahtevajo določeno število **vhodnih podatkov**. Vhodne podatke, na vrednosti katerih ne moremo vplivati, imenujemo **nenadzorovani podatki**. V primeru odločanja o naložbi je nenadzorovan podatek lahko obrestna mera, saj njene vrednosti ne moremo spreminjati. Očitno je seveda lahko isti podatek v nekem drugem okolju ali le v drugem modelu **nadzorovan**. Nadzorovanim podatkom rečemo tudi **odločitvene spremenljivke**, saj so to odločitvene alternative, ki smo jih določili v fazi oblikovanja. Z drugimi besedami, želimo določiti čimboljše vrednosti odločitvenih spremenljivk glede na izbrani kriterij.

Izdelek A prodajamo po ceni 500 denarnih enot za kos. Fiksni stroški pri proizvodnji tega izdelka znašajo 300.000 denarnih enot, variabilni pa 200 denarnih enot za kos. Zanima nas, kakšna je najmanjša količina izdelkov A, ki jo moramo proizvesti, da bomo imeli dobiček. To količino izdelkov imenujemo prelomna točka.

Poglejmo, kakšen bi lahko bil model za zgornji problem. Kriterij, po katerem ocenjujemo možne rešitve (količino izdelkov), je v tem primeru dobiček. Poskusimo izraziti dobiček z vhodnimi podatki:

dobiček = prihodki - stroški
prihodki = cena*količina
stroški = fiksni_stroški + variabilni_stroški * količina

Torej je lahko model za naš problem naslednji:

dobiček = cena*količina - (fiksni_stroški + variabilni_stroški * količina)

Kot vidimo, so vhodni podatki: cena, količina, fiksni stroški in variabilni stroški. Od teh je le količina nadzorovan vhodni podatek, saj drugih (v okviru te odločitve) ne moremo spreminjati. Opozorimo na to, da bi lahko bil v nekem drugem procesu odločanja nadzorovan podatek, na primer cena, po kateri prodajamo izdelek.

Zgodi se, da v času odločanja ne poznamo vrednosti nekega nenadzorovanega podatka. Recimo, da za model potrebujemo kot vhodni podatek povpraševanje po izdelku, ki ga šele razvijamo. Če so vsi nenadzorovani vhodni podatki poznani in se ne morejo spreminjati, potem je **model deterministično**, odločanje poteka v pogojih **gotovosti** (angl. ceratinty). Če mora odločevalec analizirati problem za različne vrednosti nekega parametra, pri čemer pozna za te vrednosti verjetnostno porazdelitev, govorimo o **tveganju** (angl. risk), če pa teh verjetnosti ne pozna oziroma ne more oceniti, pa govorimo o **negotovosti** (angl. unceratinity).

Boljše poznavanje problema

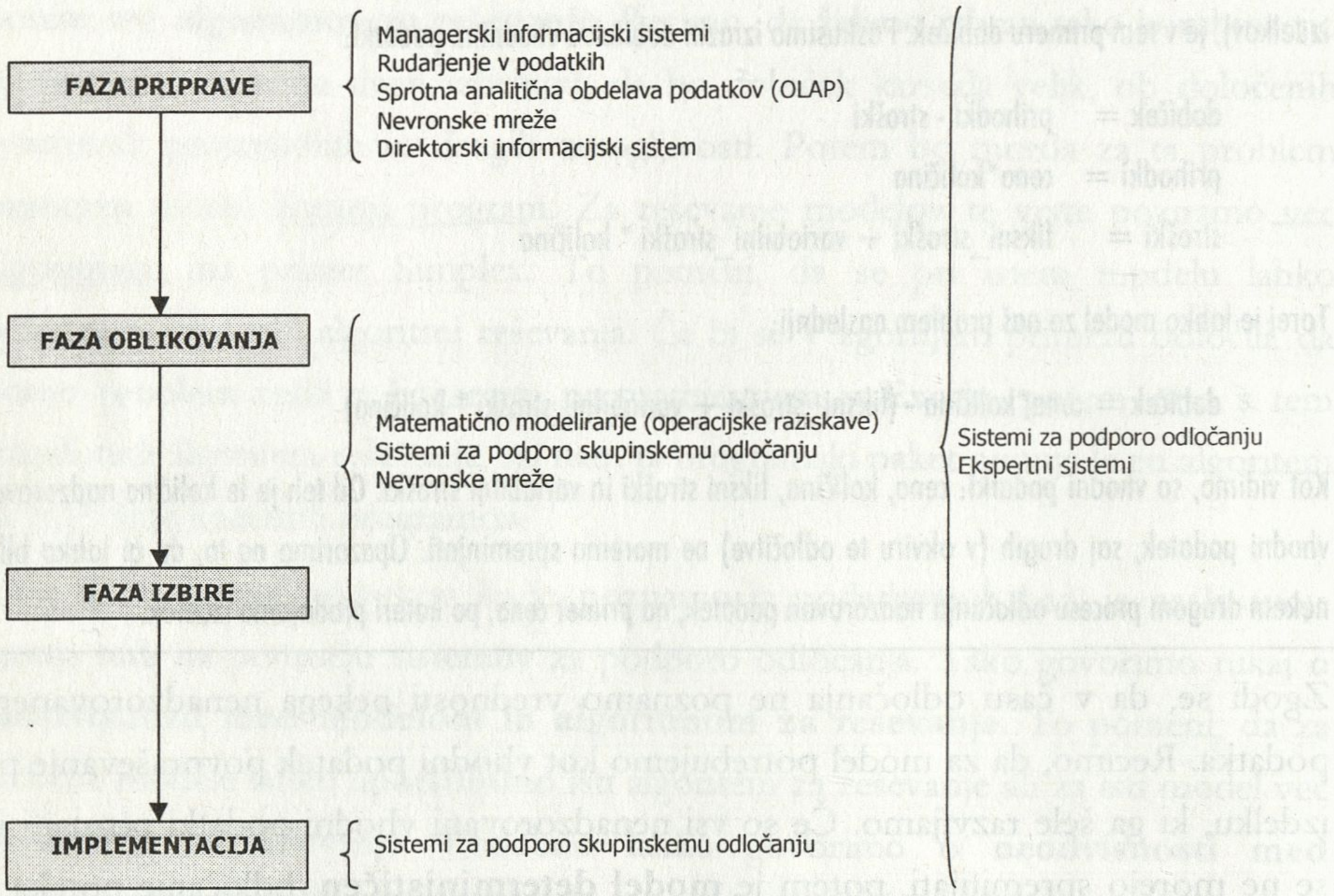
POPOLNO POZNAVANJE PROBLEMA, GOTOVOST	TVEGANJE	NEPOZNAVANJE PROBLEMA NEGOTOVOST

Slika 44: Odločanje in poznavanje problema

Priprava vhodnih podatkov za model je zelo pomemben del analize problema, saj tudi tu velja splošno pravilo v informatiki: *GIGO (Garbage In Garbage Out) - smeti noter, smeti ven.*

Informacijska tehnologija v procesu odločanja

Na sliki 45 so prikazane različne informacijske tehnologije, ki se uporabljajo za podporo v različnih fazah odločitvenega procesa.



Slika 45: Uporaba tehnologij za podporo odločanju (Turban, Aronson, 2001)

S sprotnim spremljanjem notranjih in zunanjih informacij lahko pravočasno zaznamo problem oziroma poslovno priložnost, kar omogoča identifikacijo odločitvenega problema. Prav tako lahko različne tehnologije (managerski informacijski sistem, direktorski informacijski sistem, OLAP, rudarjenje v podatkih ...) omogočajo pridobivanje podatkov v pripravljalni fazi. V fazi oblikovanja in odločitve uporabljamo tehnologije, ki omogočajo izdelavo modela ali kvalitativno analizo problema. Mogoče je uporabiti standardne modele, manj strukturirane probleme pa lahko podpremo s sistemi za podporo odločanju, ki omogočajo oblikovanje prilagodljivih modelov (npr. programi za delo s preglednicami). Iskanje alternativnih rešitev lahko podpremo npr. z orodji za skupinsko delo ali ekspertnimi sistemi. Raziskave so pokazale, da je informacijska podpora enako

pomembna tudi v fazi implementacije za potrebe komunikacije, razlag in utemeljitev ter reševanja problemov pri uvajanju odločitev.

4.1 Sistemi za podporo odločanju

Razumevanje pojma sistemi za podporo odločanju je tako v strokovni literaturi kakor tudi v praksi zelo raznoliko. Najpogosteje se sistemi za podporo odločanju povezujejo s polstrukturiranimi problemi. Zato v nadaljevanju privzemimo naslednjo opredelitev: Sistemi za podporo odločanju so računalniški sistemi, ki podpirajo proces odločanja managerjev predvsem pri polstrukturiranih poslovnih odločitvenih problemih (Turban, Aronson, 2001). Podpirajo lahko le posamezno fazo, lahko pa celoten odločitveni proces, včasih tudi s povezovanjem z drugimi vrstami informacijskih sistemov. Vedeti moramo, da so programske rešitve, ki jih pri tem uporabljamo, v splošnem le del sistemov za podporo odločanja. Res pa je, da so danes mnogi programski paketi tako izpopolnjeni, da jih managerji pogosto uporabljajo kar samostojno.

Sistemi za podporo odločanju nam omogočajo na podlagi zbranih podatkov o sistemu odgovarjati na dileme, ki se pojavljajo pri odločanju. Na primer:

- Kakšne bodo posledice, če sprejmemo neko poslovno odločitev? (kaj-če analiza, angl. what-if)

  Na primer, kako se bo spremenil dobiček, če se odločimo, da v časopisu objavimo tri dodatne oglase? Če naj odgovorimo na to vprašanje, moramo seveda poznati povezave med dobičkom, prihodkom, stroški, reklamnimi sporočili, prodajo... Množica zvez med temi spremenljivkami je model za ta problem.

- Kako naj spremenimo vrednosti določenih parametrov, če želimo doseči nek cilj? (doseganje cilja, angl. goal-seeking)

  V primeru objavljanja oglasov se lahko vprašamo, koliko dodatnih sporočil moramo objaviti, da se bo dobiček povečal za določeno vrednost.

- Katera odločitev je najboljša? (optimizacija, angl. optimization)

Zdaj se lahko odločamo med objavljanjem oglasov v časopisu in na televiziji, kjer so objave seveda dražje, imajo pa močnejši učinek. Omejeni smo tudi z denarjem, ki je na voljo za oglaševanje. Koliko sporočil naj objavimo v katerem mediju, da bo dobiček največji, če naj ne prekoračimo zneska, ki nam je na voljo?

- Zakaj ima neka spremenljivka vrednost, kot jo ima? (zakaj, angl. why)

  Na to vprašanje odgovarja predvsem statistična analiza, ki išče statistične povezave med spremenljivkami. S tem si lahko pomagamo pri iskanju dejanske odvisnosti med spremenljivkami. Povezavo med objavo oglasa na televiziji in povečano prodajo izdelka lahko ugotovimo le na podlagi statistične analize podatkov o preteklih objavah in prodaji.

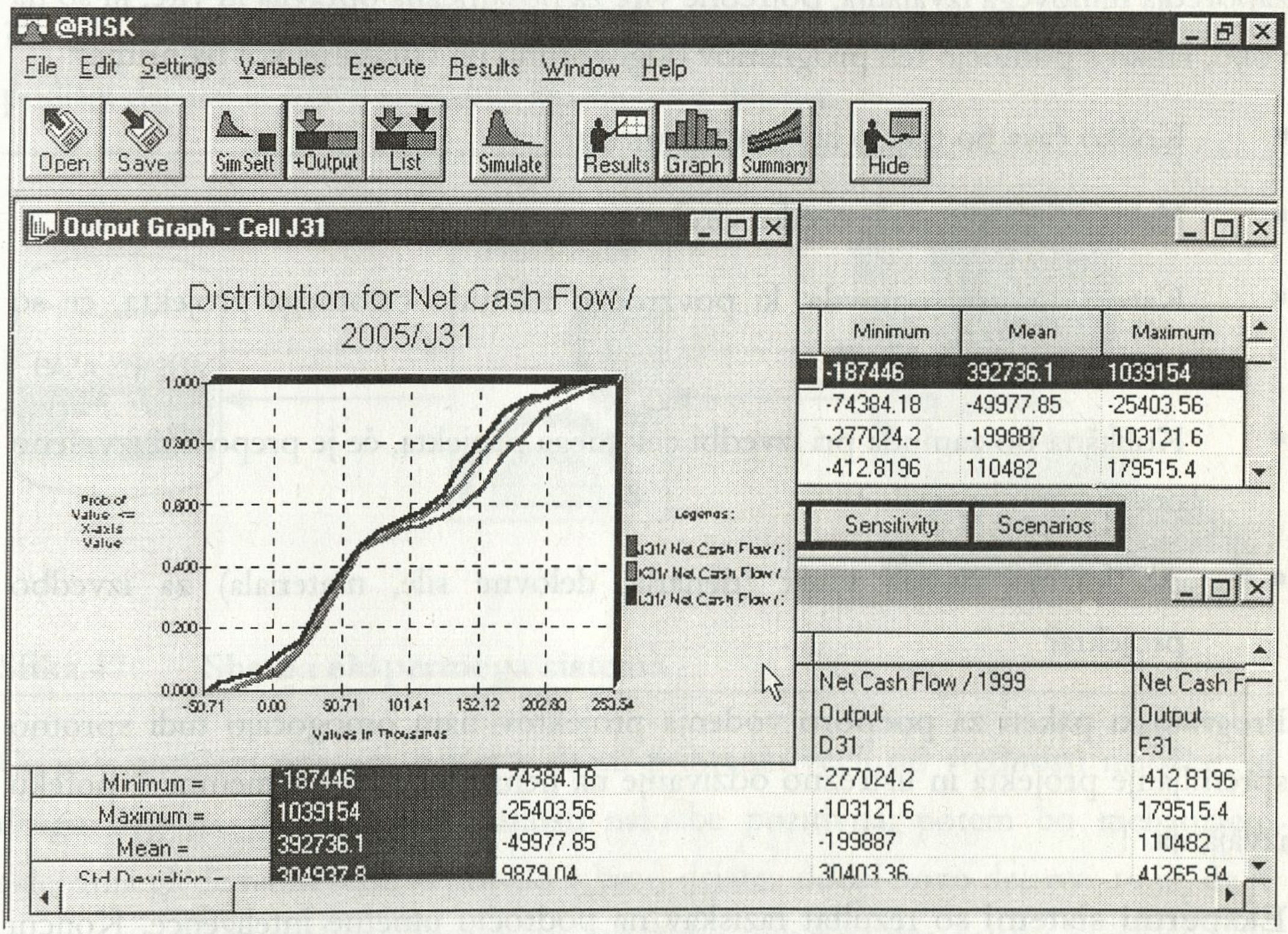
Uporaba računalniške programske in strojne opreme v sistemih za podporo odločanja omogoča hitrejše in kvalitetnejše odločanje. Z vse zmogljivejšo strojno opremo in do uporabnika vse bolj prijaznimi programskimi paketi za podporo odločanja so le-ti postali dostopni skoraj vsakomur.

Med programsko opremo, ki jo uporabljamo za podporo odločanju, največkrat srečamo:

- orodja za izdelavo poročil in poizvedovanje, tudi orodja za sprotno analitično obdelavo podatkov (OLAP);
- programske pakete za rudarjenje v podatkih, na primer SAS, Clementine, MS Analysis Services, Darwin;
- programske pakete za delo s preglednicami, na primer Excel, Quatro Pro, Lotus 1-2-3;
- programske pakete za statistično analizo, na primer SPSS, SAS, StatGraph;
- specializirane programske pakete za modeliranje, na primer IFPS, @Risk; Precision Tree, What's Best!;
- programske pakete za podporo vodenju projektov, na primer MS Project, Super Project;
- ekspertne sisteme, na primer XpertRule KBS.

Programski paketi za statistično analizo pomagajo uporabniku iskati povezave med dvema ali več spremenljivkami. Odgovarjajo torej na vprašanje "zakaj?". Čeprav so ti programski paketi v zadnjem času postali zelo zmogljivi in prijazni do uporabnika, pa je interpretacija rezultatov še vedno na strani uporabnika.

Programski paketi za delo s preglednicami so namenjeni splošnemu analitičnemu modeliranju. V naslednjih poglavjih bo eden izmed njih (MS Excel) orodje, ki ga bomo uporabljali pri razvoju in uporabi modelov.



Slika 46: Zaslon @Risk

Specializirani programski paketi za modeliranje so v večini primerov podobni programskim paketom za delo s preglednicami, pogosto pa so celo njihova nadgradnja (na primer @Risk). Namenjeni so modeliranju na ožjih področjih. Tako je, na primer, IFPS namenjen modeliranju na finančnem področju, @Risk pa tam, kjer se pojavlja v modelih tveganje. Razlika med temi paketi in paketi za delo s preglednicami je predvsem ta, da specializirani paketi ponujajo močnejša orodja za modeliranje na specializiranem področju, seveda pa je omejenost na eno področje

njihova slabost. Ker tudi programski paketi za delo s preglednicami ponujajo vse močnejša orodja, ki so primerna za več področij, se zdi, da bo razvoj v prihodnje vse manj naklonjen samostojnim specializiranim paketom in vse bolj nadgradnjam preglednic za določena področja.

Sistemi za podporo vodenju projektov so v pomoč uporabnikom pri načrtovanju in organizaciji opravil v okviru projektov z upoštevanjem razpoložljivih virov. Glede na dolžine trajanja posameznih opravil, zahtevana zaporedja njihovega izvajanja, potrebne vire za posamezna opravila in vire, ki so na voljo, lahko s pomočjo teh programov odgovorimo na vprašanja kot na primer:

- Koliko časa bo trajalo izvajanje projekta?
- Kako učinkovito razporediti vire?
- Katera so tista opravila, ki povzročijo zamudo celotnega projekta, če so izvršena prepozno?
- Kolikšna bo zamuda pri izvedbi celotnega projekta, če je prepozno izvršeno posamezno opravilo?
- Ali imamo dovolj virov (denarja, delovne sile, materiala) za izvedbo projekta?

Programski paketi za podporo vodenja projektov nam omogočajo tudi sprotno spremljanje projekta in ustrezno odzivanje na nenačrtovane spremembe v poteku izvajanja.

Ekspertni sistemi so rezultat raziskav na področju umetne inteligence. Končni cilj umetne inteligence je razvoj računalnika, ki misli: se uči, logično sklepa in rešuje probleme. Med dosedanjimi rezultati na tem področju se v praksi največ uporabljajo ravno ekspertni sistemi. Na primer, ekspertni sistemi lahko zamenjajo strokovnjaka ali mu pomagajo pri odločitvah v tistih elementih odločanja, ki se na področju, s katerim se ukvarja ekspert, večkrat ponavljajo.

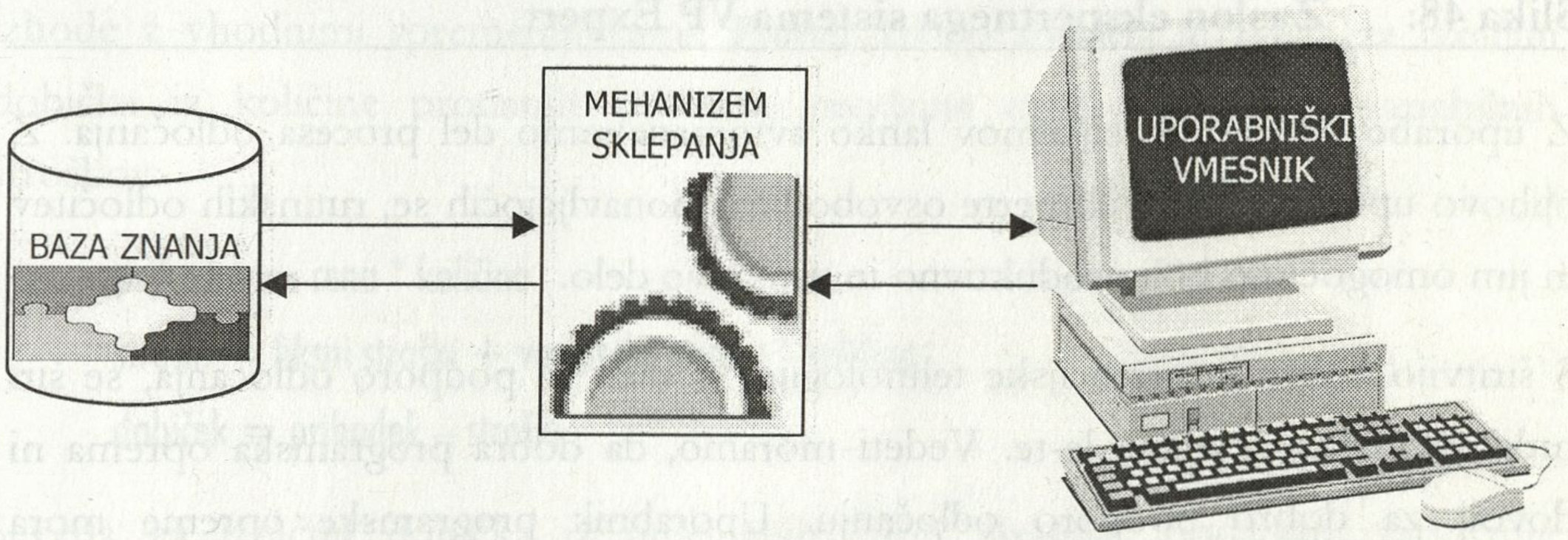
Glede na bolezenske znake pacienta se zdravnik odloči, za katero bolezen naj bi šlo. V mnogih primerih je ta odločitev dokaj rutinska. Seveda lahko ekspertni sistem zdravniku le pomaga pri odločitvah, ne more ga pa zamenjati.

Splošna arhitektura ekspertnih sistemov je naslednja (slika 47):

Ekspertni sistem vsebuje **bazo znanja**. Baza znanja je navadno sestavljena iz podatkovne baze, v kateri so **dejstva o sistemu**, ter **baze pravil sklepanja**. Pravila sklepanja so največkrat oblike ČE-POTEM (IF-THEN), na primer:

ČE prihranki > vrednost naložbe
IN ocena varnosti naložbe = pozitivna
POTEM je pametno naložiti sredstva

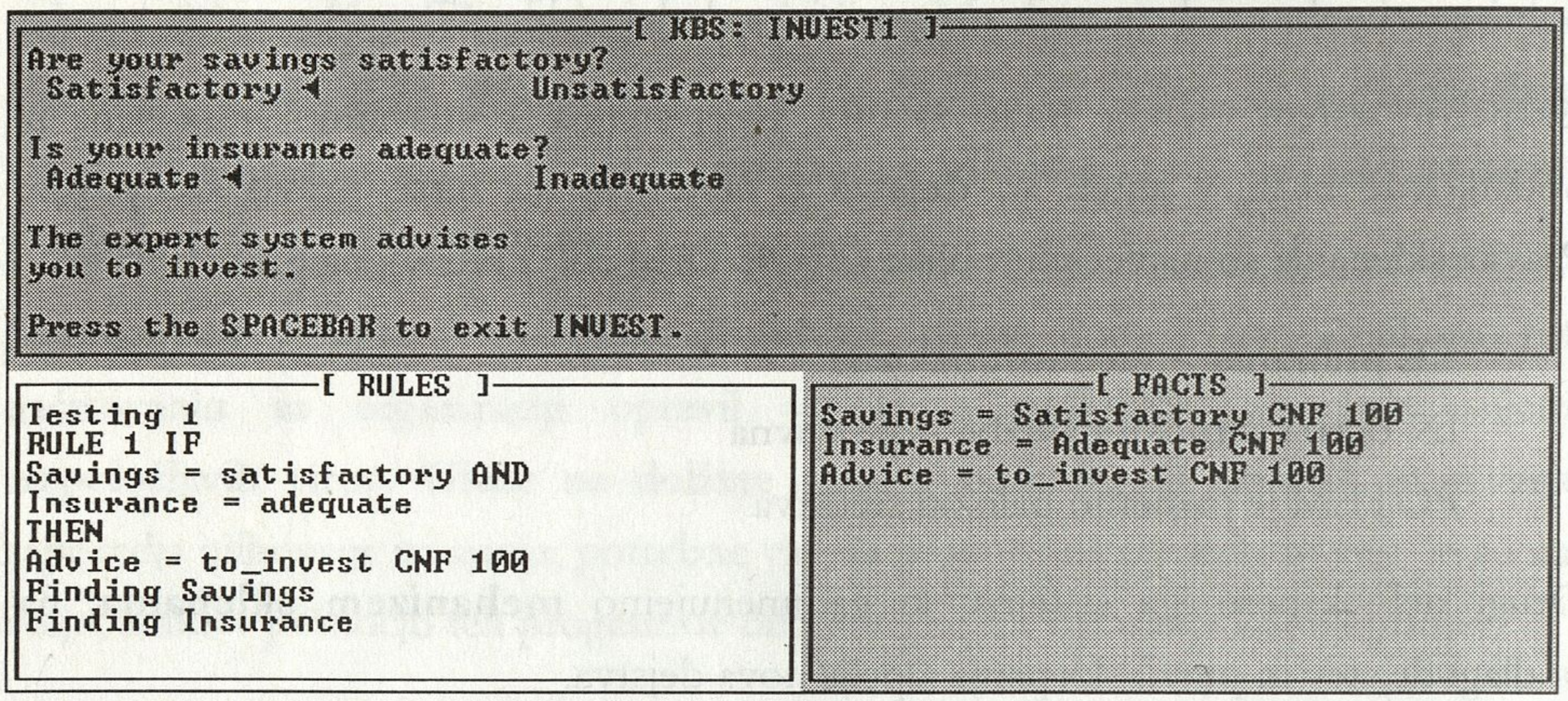
Drugi del ekspertnega sistema, ki ga imenujemo **mehanizem sklepanja**, na podlagi dejstev in pravil sklepanja izpelje nova dejstva.



Slika 47: Shema ekspertnega sistema

Če je v gornjem primeru dejstvo, da so prihranki večji od vrednosti naložbe, in drugo dejstvo, da je ocena varnosti naložbe pozitivna, potem bo mehanizem sklepanja glede na navedeno pravilo v bazo dejstev dodal novo dejstvo, to je, da je smotrno naložiti sredstva. Seveda sta bili lahko tudi prvi dve dejstvi že izpeljani iz nekih drugih dejstev z uporabo nekih drugih pravil sklepanja.

Na sliki 48 vidimo zaslon ekspertnega sistema, ki je razdeljen na tri dele. Zgornji del je uporabniški vmesnik, kar je navadno vse, kar uporabnik ekspertnega sistema vidi. V spodnjem levem delu je vidno pravilo, ki ga trenutno uporabljamo, desno spodaj pa je baza dejstev, ki so trenutno znana o sistemu.



Slika 48: Zaslon ekspertnega sistema VP Expert

Z uporabo ekspertnih sistemov lahko avtomatiziramo del procesa odločanja. Z njihovo uporabo lahko eksperte osvobodimo ponavljajočih se, rutinskih odločitev in jim omogočimo bolj produktivno in zanimivo delo.

S širitvijo dobre informacijske tehnologije, ki služi za podporo odločanja, se širi tudi nekritična uporaba le-te. Vedeti moramo, da dobra programska oprema ni dovolj za dobro podporo odločanju. Uporabnik programske opreme mora pripraviti dober model in vhodne podatke, če želi dobiti kakovostne rezultate, te pa mora znati pravilno interpretirati in naprej uporabiti v procesu odločanja.

4.2 Uporaba programskih paketov za delo s preglednicami

Ceprav na prvi pogled le privlačno orodje za oblikovanje tabel, so programski paketi za delo s preglednicami mnogo več. Večina sodobnih programskih paketov za delo s preglednicami omogoča statistične obdelave podatkov in uporabo preglednice kot preproste podatkovne baze. Seveda je na teh področjih njihova zmogljivost precej manjša od specializiranih statističnih programskih paketov in sistemov za upravljanje podatkovnih baz.

Svojo pravo moč pa programski paketi za delo s preglednicami pokažejo kot orodje za modeliranje. Za nekatera področja sicer obstajajo tudi specializirana orodja za modeliranje, na primer IFPS ali @Risk, ki pa so pogosto rešitve, zgrajene nad programskimi paketi za delo s preglednicami.

V nadaljevanju je podanih nekaj vrst modelov, ki jih lahko uporabljamo pri poslovnem odločanju. Kot primer programskega paketa za delo s preglednicami je uporabljen Microsoft Excel.

Preizkušanje scenarijev in doseganje cilja

V preglednicah najpogosteje oblikujemo matematične modele, ki povezujejo izhode z vhodnimi spremenljivkami. Primer takega modela je zveza za izračun dobička iz količine prodanih izdelkov, prodajne cene, fiksnih in variabilnih stroškov:

prihodek = cena * količina

stroški = fiksni stroški + variabilni stroški * količina

dobiček = prihodek – stroški

Seveda za izračun dobička iz omenjenih štirih vhodnih podatkov zadostuje najpreprostejši kalkulator. Zavedati pa se moramo, da velikost modelov v praksi lahko doseže nekaj deset spremenljivk in nekaj sto povezav (formul). V takih primerih se programski paketi za delo s preglednicami izkažejo pri:

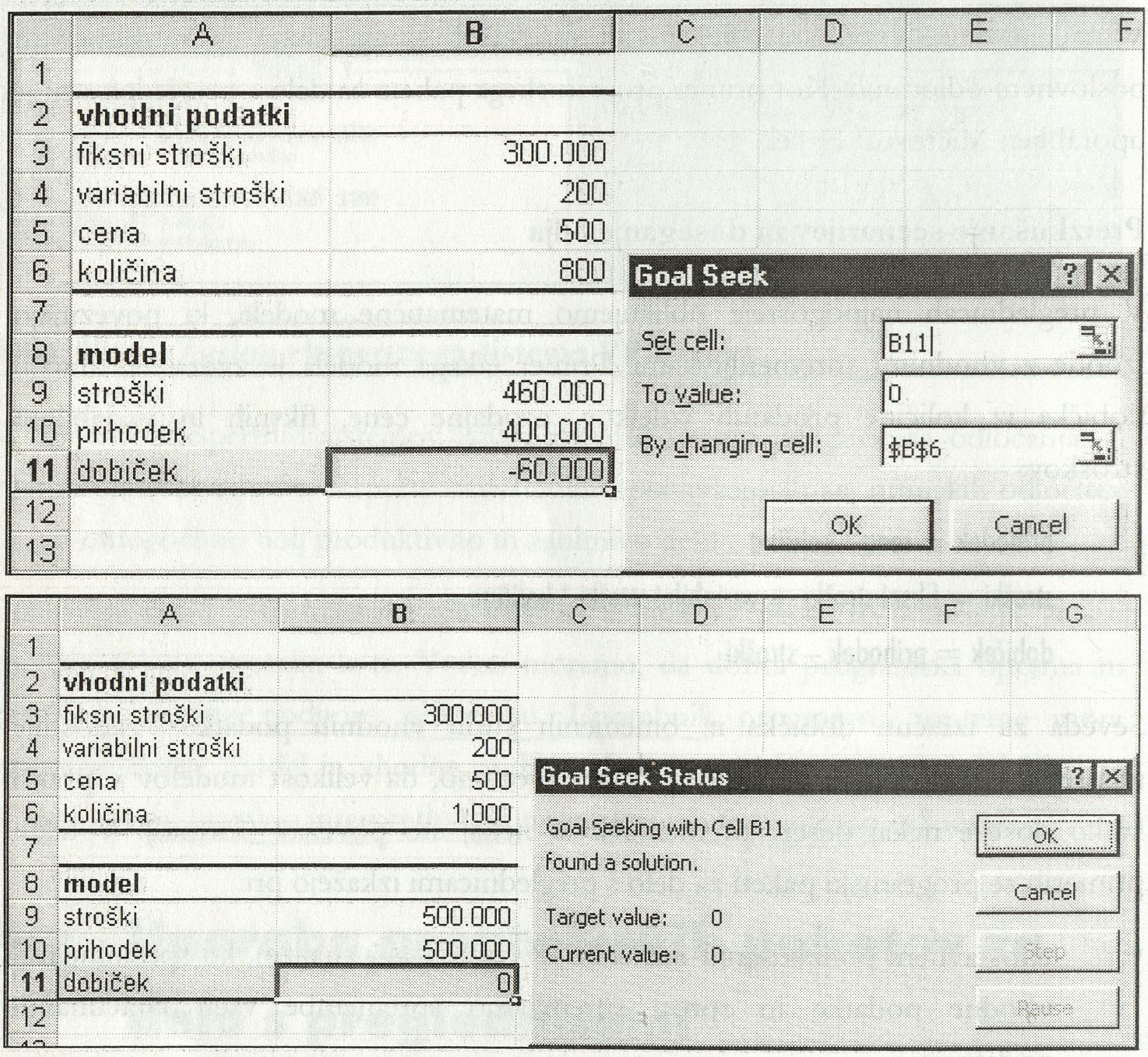
- enostavnem preizkušanju različnih *scenarijev* (*kaj-če* analize), ko spreminjamo vhodne podatke in sproti spremljamo spremembe vseh izračunanih podatkov,
- *doseganju cilja*, to je določitev vrednosti vhodne spremenljivke, da bo izhodna imela neko (vnaprej izbrano) vrednost.

V zgornjem primeru bi lahko preverjali različne scenarije, kaj bi se dogajalo, če spreminjamo ceno izdelka[39]. Pri spremembi cene se takoj preračuna prihodek in

---

39 Opozoriti moramo, da je uporabljeni model zelo poenostavljen, saj je namen le predstavitev možnosti, ki jih ponuja informacijska tehnologija. V modelu npr. ni upoštevano, da sprememba cene praviloma spremeni povpraševanje po izdelku, torej količino prodanih izdelkov.

dobiček. Točko preloma (angl. break-even analysis), to je količino, pri kateri bo dobiček enak 0, bi lahko dobili s preizkušanjem scenarijev, bolj elegantno (in učinkovito pri večjih modelih) pa to naredimo v Excelu z uporabo modula Goal Seeking (glej sliko 49).



Slika 49: Določitev prelomne točke v Excelu

Vrtilne tabele

Vrtilna tabela prikazuje sumarne podatke ene spremenljivke (odvisna spremenljivka) glede na vrednosti drugih dveh (ali več) spremenljivk (neodvisni spremenljivki). Vzemimo dve spremenljivki U in V, ki lahko zavzameta vrednosti u_1, u_2, u_3 in u_4 ter v_1, v_2 in v_3. Zanimajo nas sumarni podatki (na primer vsota) za spremenljivko S. Oblika vrtilne tabele za ta primer je prikazana na sliki 50.

	v_1	v_2	v_3
u_1	s_{11}	s_{12}	s_{13}
u_2	s_{21}	s_{22}	s_{23}
u_3	s_{31}	s_{32}	s_{33}
u_4	s_{41}	s_{42}	s_{43}

Slika 50: Skica dvodimenzionalne vrtilne tabele

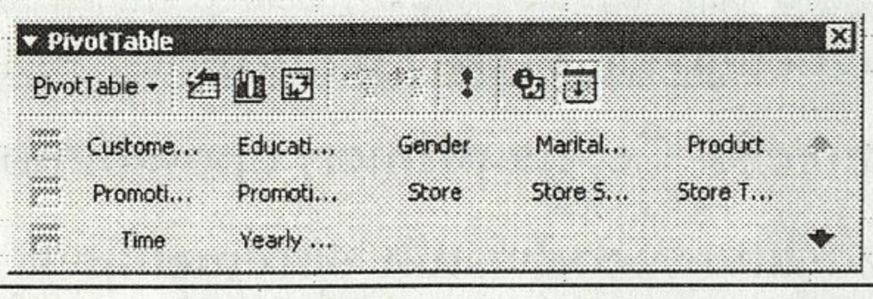
Pri tem s_{ij} pomeni vsoto vrednosti spremenljivke S za vse enote, pri katerih ima spremenljivka U vrednost u_i, spremenljivka V pa vrednost v_j. Poleg vsote lahko uporabimo tudi druge sumarne funkcije. Če, na primer, uporabimo funkcijo štetja (Count), dobimo frekvenčne porazdelitve neodvisnih spremenljivk. Vrtilno tabelo lahko posplošimo tudi na več dimenzij, ko namesto dveh (U in V) uporabimo več neodvisnih spremenljivk.

Če z vrtilno tabelo nismo zadovoljni, jo lahko zelo enostavno spremenimo, na primer, zamenjamo vrstice in stolpce tabele.

Po krajšem "igranju" s tem orodjem bo tudi še neizkušeni analitik ugotovil, da gre kljub preprosti uporabi za zmogljivo in fleksibilno orodje, ki omogoča prikaz poljubnega prereza podatkov.

Vrtilno tabelo v Excelu povežemo tudi z večdimenzionalno bazo podatkov (npr. MS Analysis Services), s tem pa postane vrtilna tabela v Excelu preprosto orodje za analize OLAP (glej Sprotna analitična obdelava podatkov v nadaljevanju). Na sliki 50 je prikazan primer vrtilne tabele, ki prikazuje dobiček glede na državo (angl. Country, State), izobrazbo (angl. Education Level) in promocijski medij (angl. Promotion Media). Takšno poročilo je izdelano na enak način kot običajna vrtilna tabela, le da za podatkovni vir izberemo večdimenzionalno kocko. V primeru uporabe vrtilne tabele za analize OLAP se podatki v vrtilni tabeli sproti (glede na nastavitev) spreminjajo v skladu s spremembami v podatkovnem viru.

	A	B	C	D	E	F	G	H	I
1									
2	Promotion Media	Bulk Mail							
3									
4	Profit			Education Level					
5	**Country**	**State Province**	**City**	Bachelors Degree	Graduate Degree	High School Degree	Partial College	Partial High School	Grand Total
6	USA	CA		857,633	622,6478	1286,5439	970,1866	326,7418	4063,7531
7		WA	Ballard	17,1426		9,3692		18,0824	44,5942
8			Bremerton	54,0185	53,9912	36,6356	169,4009	45,6904	359,7366
9			Burien	25,3848	6,408		21,2874	25,2158	78,296
10			Edmonds				13,3348		13,3348
11			Everett		34,527	6,286			40,813
12			Issaquah		6,7624		33,8406		40,603
13			Kirkland	20,8468		42,4891	14,4999		77,8358
14			Lynnwood			30,4962	30,517		61,0132
15			Marysville				20,1751		20,1751
16			Port Orchard	25,5114	59,035	109,4208	55,0413		249,0085
17			Redmond		15,4665				15,4665
18			Renton	37,8772		2,3424		2,5806	42,8002
19			Seattle			16,8957	57,2648		74,1605
20			Spokane	196,4411		70,749	136,5629	22,7749	426,5279
21		WA Total *		377,2224	176,1901	324,684	551,9247	114,3441	1544,3653
22	USA Total *			1234,8554	798,8379	1611,2279	1522,1113	441,0859	5608,1184
23	Grand Total *			1234,8554	798,8379	1611,2279	1522,1113	441,0859	5608,1184
24									
25									
26									
27									
28									
29									
30									
31									
32									



Slika 51: Primer sprotne analitične obdelave podatkov (OLAP) v Excelu

Časovne vrste in napovedovanje

Časovna vrsta je zaporedje vrednosti neke spremenljivke, ki so bile ugotovljene v zaporednih časovnih točkah ali v nekem časovnem obdobju, na primer, zaporedje vrednosti dnevne prodaje, zaporedje letnih količin proizvodnje jekla v neki državi, gibanje temperature, rodnost v neki državi. Pogosto nas zanima, kako se bo pojav obnašal v prihodnosti, oziroma kakšne bodo vrednosti opazovane spremenljivke v prihodnje. Za napovedovanje sicer lahko uporabimo tudi kvalitativne metode (napovedi ekspertov), pa obstaja tudi vrsta metod za napovedovanje na podlagi podatkov o obnašanju pojava v preteklosti. Excel ponuja (neposredno) za napovedovanje le omejene možnosti[40], vendar lahko s pridom izkoristimo nekatere njegove druge elemente.

40 Gre za možnost uporabe regresijske enačbe in za drseča povprečja, ki jih lahko prikažemo na grafih.

Na sliki 52 so v stolpcih C, E in G preglednice zaporedoma prikazane uporabe metod drseča povprečja, posplošena drseča povprečja in eksponentno glajenje[41]. V pripadajočih desnih stolpcih (D, F in H) pa je za vsako metodo prikazan še izračun povprečne kvadratne napake.

	A	B	C	D	E	F	G	H
1							alfa	0,4
2		dejanske vrednosti	drseča povprečja	PKN	posplošena drseča povprečja	PKN	eksponentno glajenje	PKN
3	1	22					=B3	
4	2	28					=alfa*B4+(1-alfa)*G3	=(B4-G4)^2
5	3	29					=alfa*B5+(1-alfa)*G4	=(B5-G5)^2
6	4	24	=AVERAGE(B3:B5)	=(B6-C6)^2	=0,5*B5+0,3*B4+0,2*B3	=(B6-E6)^2	=alfa*B6+(1-alfa)*G5	=(B6-G6)^2
7	5	27	=AVERAGE(B4:B6)	=(B7-C7)^2	=0,5*B6+0,3*B5+0,2*B4	=(B7-E7)^2	=alfa*B7+(1-alfa)*G6	=(B7-G7)^2
8	6	23	=AVERAGE(B5:B7)	=(B8-C8)^2	=0,5*B7+0,3*B6+0,2*B5	=(B8-E8)^2	=alfa*B8+(1-alfa)*G7	=(B8-G8)^2
9	7	25	=AVERAGE(B6:B8)	=(B9-C9)^2	=0,5*B8+0,3*B7+0,2*B6	=(B9-E9)^2	=alfa*B9+(1-alfa)*G8	=(B9-G9)^2
10	8	28	=AVERAGE(B7:B9)	=(B10-C10)^2	=0,5*B9+0,3*B8+0,2*B7	=(B10-E10)^2	=alfa*B10+(1-alfa)*G9	=(B10-G10)^2
11	9	22	=AVERAGE(B8:B10)	=(B11-C11)^2	=0,5*B10+0,3*B9+0,2*B8	=(B11-E11)^2	=alfa*B11+(1-alfa)*G10	=(B11-G11)^2
12	10	27	=AVERAGE(B9:B11)	=(B12-C12)^2	=0,5*B11+0,3*B10+0,2*B9	=(B12-E12)^2	=alfa*B12+(1-alfa)*G11	=(B12-G12)^2
13	11	24	=AVERAGE(B10:B12)	=(B13-C13)^2	=0,5*B12+0,3*B11+0,2*B10	=(B13-E13)^2	=alfa*B13+(1-alfa)*G12	=(B13-G13)^2
14	12	29	=AVERAGE(B11:B13)	=(B14-C14)^2	=0,5*B13+0,3*B12+0,2*B11	=(B14-E14)^2	=alfa*B14+(1-alfa)*G13	=(B14-G14)^2
15	13	38	**=AVERAGE(B12:B14)**	=AVERAGE(D6:D14)	**=0,5*B14+0,3*B13+0,2*B12**	=AVERAGE(F6:F14)	**=alfa*B15+(1-alfa)*G14**	=AVERAGE(H4:H14)

Slika 52: Uporaba metod drseča povprečja, posplošena drseča povprečja in eksponentno glajenje

Nekatera orodja imajo že vgrajene module, ki računajo iz danih časovnih vrst napovedi po različnih metodah. Gre predvsem za statistične programske pakete, pa tudi pri orodjih OLAP postaja možnost napovedovanja čedalje bolj pogosta. Obstaja pa tudi možnost nadgradnje preglednice MS Excel z orodjem Forecast Pro, ki uporablja vgrajen ekspertni sistem za določitev metode, ki bo za konkreten primer zagotovila najboljšo napoved. Tako orodje je namenjeno tudi analitikom, ki se na metode napovedovanja ne spoznajo najbolje.

Optimizacija

Z optimizacijo želimo določiti vrednosti nadzorovanih odločitvenih spremenljivk, tako da bo neka od njih odvisna količina imela maksimalno (največjo) vrednost ali minimalno (najmanjšo) vrednost. Pri tem moramo pogosto upoštevati še omejitve in zahteve, ki določajo, kakšne smejo biti vrednosti nadzorovanih spremenljivk.

Pri poslovnih odločitvah mnogokrat nastopajo na primer takšnile primeri:

41 Za opis metod glej npr. (Winston, 1991)

- Vodja trženja mora odločiti, koliko denarja naj podjetje vloži v različne oglaševalske medije, da bo učinek največji.
- Koliko katerega izdelka naj izdeluje podjetje, da bo dobiček največji? Pri tem je podjetje omejeno s povpraševanjem kupcev po posameznih izdelkih, s proizvodno zmogljivostjo in/ali drugimi omejitvenimi danostmi.
- Podjetje isti izdelek izdeluje v več obratih po Sloveniji. Kupci izdelka se prav tako nahajajo na različnih lokacijah v Sloveniji. Podjetje mora zadostiti povpraševanju kupcev, želi pa minimizirati transportne stroške. Koliko naj dostavi izdelkov iz posameznega obrata posameznemu kupcu?

Če želimo reševati probleme optimizacije, moramo najprej dobro formulirati problem:

- določiti nadzorovane odločitvene spremenljivke,
- določiti omejitve in
- določiti **kriterijsko funkcijo**, to je funkcijo odločitvenih spremenljivk, ki jo želimo maksimizirati ali minimizirati.

Podjetje izdeluje dva izdelka I1 in I2. Dobiček pri 1 kg izdelka I1 je 25 denarnih enot, pri 1 kg izdelka I2 pa 30 denarnih enot. Obrat, v katerem izdelujejo oba izdelka, ima tri oddelke (A, B in C). Proizvodni proces zahteva za oba izdelka določeno število delovnih ur na vsakem izmed oddelkov. Število potrebnih delovnih ur za en kilogram izdelka je naslednje:

	I1	I2
A	1,50	3,00
B	2,00	1,00
C	0,25	0,25

V prihodnjem mesecu bodo imeli na voljo 450 delovnih ur na oddelku A, 350 na oddelku B in 50 na oddelku C.

Koliko katerega izdelka naj izdelajo v prihodnjem mesecu, da bo dobiček največji?

V tem primeru so kriterijska funkcija in vse omejitve *linearne funkcije odločitvenih spremenljivk*. Takemu optimizacijskemu modelu pravimo **linearni program**, reševanju problema pa **linearno programiranje**. Za linearne optimizacijske probleme poznamo več algoritmov za reševanje. Programski paket Excel ima

vgrajen modul Solver, ki omogoča tudi reševanje linearnih optimizacijskih problemov[42].

V praksi pa pogosto nastopajo tudi optimizacijski problemi, ki **niso linearni**. Excel omogoča njihovo reševanje na povsem enak način, kot velja za linearne probleme, le posebej moramo povedati, da gre za nelinearni problem. Čeprav je tudi reševanje nelinearnih optimizacijskih problemov tako enostavno, pa moramo tukaj še posebej upoštevati dejstvo, da je programski paket le orodje in je interpretacija rezultatov še vedno naloga uporabnika. Pri nelinearnih problemih ima lahko kriterijska funkcija lokalne ekstreme, ki pa so lahko precej slabši od optimalne rešitve problema. Različne začetne vrednosti odločitvenih spremenljivk lahko dajo različne rešitve. Poleg tega problema so možni še drugi, ki so povezani z numeričnim postopkom reševanja problema, ki pa jih tu ne omenjamo, saj želimo bralca le opozoriti, da ne sme slepo verjeti rezultatom, ki jih dobi kot rešitev. Seveda pa so mu lahko ti rezultati v veliko pomoč pri iskanju prave rešitve.

Na preprostem primeru pokažemo, da imajo lahko različne začetne vrednosti odločitvenih spremenljivk za posledico različne rešitve, ki jih ponudi programski paket.

V problemu nastopata dve odločitveni spremenljivki X in Y. Kriterijska funkcija je sicer linearna:

$$f(X,Y) = X$$

vendar so omejitve nelinearne:

$$(X-1)^2+(Y+2)^2 \leq 16$$

$$X^2+Y^2 \geq 13$$

kar pomeni, da problem ni linearen. Povejmo še, da želimo poiskati najmanjšo (minimizirati) vrednost kriterijske funkcije.

Obe začetni vrednosti smo torej prvič postavili na nič. Oglejmo si rezultat:

---

42 Za zahtevnejše probleme linearnega programiranja priporočamo uporabo specializiranih paketov za to področje, kot na primer LINDO.

	A	B	C	D	E	F
1	**odločitvene spremenljivke**					
2	X	3,40				
3	Y	1,20				
4						
5	**kriterijska funkcija**					
6	kf	3,40				
7						
8	**omejitve**					
9	om1	16,00				
10	om2	13,00				

Solver Results

Solver found a solution. All constraints and optimality conditions are satisfied.

Keep Solver Solution

Restore Original Values

OK | Cancel | Save Scena

Rešitev je točka (17/5,6/5), kjer je vrednost kriterijske funkcije enaka 3,4. Excel pravi, da je zadoščeno vsem omejitvam in zahtevam po optimalnosti. Poskusimo zdaj še z začetnima vrednostma X=0, Y=-1.

	A	B	C	D	E	F
1	**odločitvene spremenljivke**					
2	X	-3,00				
3	Y	-2,00				
4						
5	**kriterijska funkcija**					
6	kf	-3,00				
7						
8	**omejitve**					
9	om1	16,00				
10	om2	13,00				

Solver Results

Solver found a solution. All constraints and optimality conditions are satisfied.

Keep Solver Solution

Restore Original Values

OK | Cancel | Save Scena

Dobili smo drugo rešitev, ki prav tako zadošča omejitvam. Vrednost kriterijske funkcije je manjša, kar pomeni, da je bila prva rešitev le lokalni minimum. Ta rešitev (-3,-2) je res globalni minimum, česar pa nepozoren uporabnik Excela ne bi ugotovil.

Monte Carlo

Razmišljamo o nakupu zemljišča, ki je danes vredno 1 milijon denarnih enot. Lastnik nam ponuja možnost, da lahko ob plačilu 50.000 denarnih enot čez šest mesecev isto zemljišče kupimo za 1,1 milijon denarnih enot, ne glede na to, kakšna bo takrat njegova tržna vrednost. Seveda čez pol leta nismo obvezni kupiti zemljišča. Predpostavimo še to, da je ta nakup špekulativne narave in nameravamo zemljišče prodati takoj naprej ter pri tem zaslužiti.

Možnosti, da po preteku določenega časovnega obdobja nekaj lahko kupimo po zagotovljeni ceni, pravimo **opcija**. Za opcijo moramo seveda nekaj plačati, nekakšno zavarovalnino (v gornjem primeru 50.000 denarnih enot). Ta znesek pa se imenuje **cena opcije**.

Vrnimo se k primeru nakupa zemljišča. Najprej moramo razlikovati nakup opcije od nakupa samega zemljišča. Z nakupom opcije smo pridobili *pravico*, da čez pol leta kupimo zemljišče po danes določeni ceni.

Pred nami sta torej dve odločitvi. Danes se moramo odločiti, ali naj kupimo opcijo ali ne. Če bomo danes opcijo kupili, se bomo morali čez šest mesecev odločiti, ali naj zemljišče res kupimo.

Najprej razmislimo slednjo odločitev. Ker se bomo o nakupu zemljišča po ceni 1,1 milijon odločali le, če bomo opcijo kupili, lahko pri odločitvi o tem, ali se zemljišče splača kupiti, zanemarimo ceno opcije, saj bo ta takrat že plačana in je v nobenem primeru ne bomo mogli dobiti nazaj. Kdaj se nam torej splača kupiti zemljišče? Če bo tržna cena zemljišča čez šest mesecev manjša od 1,1 milijona, se nam zemljišča vsekakor ne splača kupiti po tej ceni, saj bomo poleg cene opcije utrpeli še dodatno izgubo. Če pa bo tržna cena večja od tega zneska, potem se nam nakup v vsakem primeru izplača, saj bomo lahko zemljišče prodali po tej ceni, ki je višja od cene, ki jo bomo mi plačali. Seveda, če bo tržna cena le malo višja od 1,1 milijona, še ne bomo imeli dobička, saj ne bomo pokrili cene opcije, vendar bomo z nakupom malo zmanjšali izgubo.

Druga odločitev, ali naj kupimo opcijo, pa je težja. Težavnost odločitve je v tem, da danes, ko se moramo odločiti, ne vemo, kolikšna bo tržna cena zemljišča čez pol leta. Tako tudi ne moremo vedeti, ali se opcijo splača kupiti.

V primeru, ko ne poznamo vrednosti neke spremenljivke, ki jo potrebujemo za odločanje, govorimo o **negotovosti**. V tem primeru lahko poskušamo čim bolje uganiti ceno (angl. best guess), vendar se lahko veliko bolje odločamo, če poznamo porazdelitev vrednosti spremenljivke. Z drugimi besedami, vprašamo se, kako verjetno je, da bo cena enaka neki vrednosti X.

Tako porazdelitev seveda v praksi dobimo na podlagi izkušenj in znanja strokovnjakov, kar pomeni, da je tudi to le predvidevanje, vendar nam rezultati

analize na podlagi porazdelitve dajejo mnogo boljše možnosti za odločanje kot pri ugibanju ene same vrednosti.

Recimo, da za naš primer pričakujemo, da je verjetnostna porazdelitev za ceno normalna porazdelitev s pričakovano vrednostjo 1 milijon in standardnim odklonom 0,15 milijona. Torej s približno 68 % verjetnostjo pričakujemo, da bo cena zemljišča med 0,85 milijona in 1,15 milijona[43]. Kako si lahko pomagamo s podatki o porazdelitvi?

Če naključno generiramo dovolj veliko število tržnih cen, kakršne naj bi bile čez 6 mesecev, lahko za vsako od njih izračunamo, kakšen bi bil dobiček (izguba) in nato izračunamo pričakovan dobiček. Seveda morajo naključno generirane cene ustrezati zahtevani porazdelitvi. S široko rabo računalnikov so se take vrste simulacij dejansko začele uporabljati v procesih odločanja, saj nam računalnik omogoča hitro generiranje velikega števila primerov in obdelavo le-teh. Zavedati se moramo, da so zaporedja števil, ki jih generiramo z računalnikom kot slučajna zaporedja, le navidezno slučajna, saj je računalnik deterministična naprava. Vendar nam v večini primerov to zadostuje.

Kadar pri reševanju problemov uporabljamo slučajnost, govorimo o **metodi Monte Carlo**.

Poglejmo končno, kako si z Excelom pomagamo pri odločanju o tem, ali naj kupimo opcijo v primeru nakupa zemljišča.

Najprej zapišimo v preglednico vrednosti, ki jih poznamo: cena zemljišča danes, ponujena cena čez šest mesecev, če kupimo opcijo, in cena opcije.

Če želimo izračunati pričakovan dobiček, potrebujemo še tržno ceno čez šest mesecev. Rekli smo, da bomo naključno generirali veliko število cen, ki ustrezajo naši predpostavki o porazdelitvi; Uporabimo generator naključnih števil (angl. Random Number Generator) v Excelu.

---

43 Gre za lastnost normalne porazdelitve, da je verjetnost, da je vrednost naključne spremenljivke med $\mu-\sigma$ in $\mu+\sigma$ približno enaka 68 %, kjer je μ povprečna vrednost, σ pa standardni odklon.

Zdaj lahko za vsako ceno izračunamo dobiček. Pri računanju dobička ne smemo pozabiti, da je vrednost denarne enote danes drugačna, kot bo vrednost ene denarne enote čez pol leta. Zato moramo pri računanju dobička vse zneske preračunati na sedanjo vrednost (neto sedanja vrednost). Če je polletna obrestna mera enaka *r*, potem dobimo današnjo vrednost nekega zneska X, ki ga plačamo čez pol leta tako, da X delimo z 1+*r*. Vnesimo v preglednico še vrednost za *r*, ki naj bo v našem primeru 7,5 %.

Če bo tržna cena manjša od 1,1 milijona, potem je dobiček -50.000 denarnih enot, saj potem zemljišča ne bomo prodali in je celoten denarni tok le cena opcije. Sicer dobiček izračunamo tako, da razliko med tržno ceno in 1,1 milijona, kolikor bomo mi plačali za zemljišče, preračunamo na sedanjo vrednost, od katere odštejemo še ceno opcije.

če tržna cena > 1.100.000 potem
  dobiček = (tržna cena - 1.100.000)/(1+r) - 50.000
sicer
  dobiček = -50.000

V Excelu lahko iz tržne vrednosti po zgornjem postopku izračunamo dobiček z uporabo vgrajene funkcije IF. Ta ima tri parametre, ki so lahko izrazi. Prvi parameter je logični izraz - pogoj. Če je njegova vrednost RES (TRUE), potem vrne funkcija vrednost drugega parametra - izraza, sicer pa ovrednoti tretji izraz in vrne njegovo vrednost. Torej za izračun dobička uporabimo naslednjo formulo:

= IF(TC>PC ; (TC-1100000)/(1+R)-CO ; -CO)

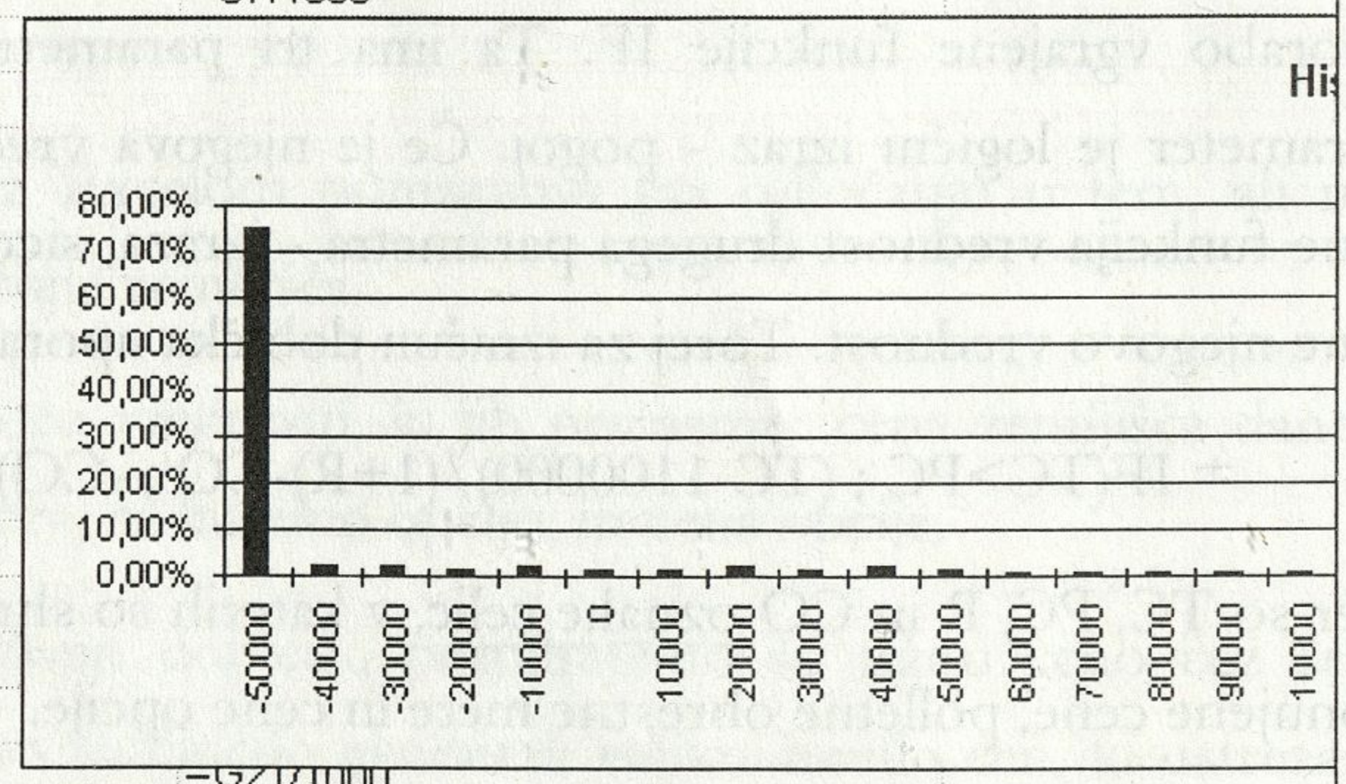
kjer so TC, PC, R in CO oznake celic, v katerih so shranjene vrednosti tržne cene, ponujene cene, polletne obrestne mere in cene opcije.

	A	B	C
1		co	50000
2		r	0,075
3		pc	1100000
4			
5	**generirane cene**		**dobiček**
6	899391,582403041		=IF(A6>pc;(A6-pc)/(1+r_)-co;-co)
7	888970,990014059		=IF(A7>pc;(A7-pc)/(1+r_)-co;-co)
8	711436,885263538		=IF(A8>pc;(A8-pc)/(1+r_)-co;-co)
9	907895,75576855		=IF(A9>pc;(A9-pc)/(1+r_)-co;-co)
10	1228349,56325823		=IF(A10>pc;(A10-pc)/(1+r_)-co;-co)
11	710225,438320776		=IF(A11>pc;(A11-pc)/(1+r_)-co;-co)
12	936689,960122749		=IF(A12>pc;(A12-pc)/(1+r_)-co;-co)
13	1162648,6891837		=IF(A13>pc;(A13-pc)/(1+r_)-co;-co)

Slika 53: Model za izračun dobička

Če tak izračun naredimo za vse generirane tržne cene, dobimo porazdelitev dobička, kot kaže slika 54.

dobiček	*frekvence*	*relativne frekvence*
-50000	751	=G6/1000
-40000	21	=G7/1000
-30000	22	
-20000	17	
-10000	23	
0	13	
10000	14	
20000	25	
30000	16	
40000	20	
50000	11	
60000	7	
70000	8	
80000	7	
90000	4	
100000	7	=G21/1000
110000	4	=G22/1000



Slika 54: Porazdelitev dobička

Iz nje vidimo, da lahko z več kot 70 % verjetnostjo pričakujemo izgubo v znesku 50.000 denarnih enot. Iz opisnih statistik za dobiček lahko vidimo, da so možni tudi zelo veliki dobički (234.984,74 denarnih enot), vendar iz frekvenčne porazdelitve razberemo, da je verjetnost za to zelo majhna.

Iz opisnih statistik tudi razberemo, da je povprečen dobiček -30.609,94 denarnih enot, kar pomeni, da lahko pričakujemo izgubo, če se odločimo za nakup opcije. Po taki ceni torej opcije ne bomo kupili.

Ali lahko tudi odgovorimo na vprašanje, koliko smo pripravljeni plačati za opcijo? Ker smo dobičke izračunali tako, da smo na koncu vedno odšteli ceno opcije (dobiček je linearna funkcija cene opcije s smernim koeficientom -1), je očitno, da sprememba cene opcije za eno denarno enoto povzroči spremembo pričakovanega dobička prav tako za eno denarno enoto. Če želimo, da je pričakovani dobiček vsaj 0, moramo ceno opcije znižati za 30.610 denarnih enot, torej na 19.390 denarnih enot.

Znižajmo zdaj ceno opcije na 15.000 denarnih enot. Pričakovani dobiček bo zdaj pozitiven (4.390 denarnih enot), vendar bo še vedno možnost izgube, čeprav manjša. Zdaj se moramo sami odločiti za nakup opcije, glede na to, koliko smo pripravljeni tvegati.

To je bil le en primer uporabe metode Monte Carlo, ki pa je uporabna povsod, kjer imamo opravka z naključnostjo oziroma nedoločenostjo. V Excelu lahko generiramo kar nekaj znanih porazdelitev, še boljšo podporo pri modelih, ki uporabljajo naključno generirana števila, pa ponuja njegova nadgradnja, to je program @Risk.

Naj ob tem primeru spet opozorimo na to, da so nam ti modeli in programska oprema le v oporo pri odločanju. V tem primeru vidimo, da moramo sami določiti (na podlagi izkušenj, znanja in občutka) porazdelitev tržnih cen. Prav tako se moramo v primeru dovolj ugodne cene opcije sami odločiti, ali smo pripravljeni tvegati.

4.3 Poslovno obveščanje

Z izrazom **poslovno obveščanje** (angl. Business Intelligence) razumemo (Inmon, Imhoff, Sousa, 1996 in Gartner Group 1999) vse sisteme, ki omogočajo uporabnikom analizo podatkov z namenom razumevanja delovanja organizacije in posledic sprejetih odločitev. V večini primerov gre za elemente sistemov za

podporo managementu na različnih nivojih. Razvoj poslovne inteligence je tesno povezan z **informacijsko demokratizacijo**, ki omogoča čedalje večjemu številu uporabnikom možnost dostopa do podatkov in njihovo analizo. Po drugi strani pa tudi pri strateških odločitvah vse pogosteje ne zadošča le presoja na podlagi izkušenj, temveč je za doseganje konkurenčnih prednosti nujna analiza velikih količin podatkov, kar je postalo mogoče z razvojem zmogljive strojne in programske računalniške opreme, nastankom sodobnih integriranih podatkovnih virov, npr. podatkovnih skladišč, in navsezadnje z dovolj velikimi količinami zbranih podatkov v digitalni obliki.

Informacijska tehnologija za poslovno inteligenco ponuja danes raznolike možnosti: od orodij za poizvedovanje po podatkovnih virih, preko orodij za sprotno analitično obdelavo podatkov do orodij za rudarjenje v podatkih in specialnih orodij za analizo.

Business Objects je ena od bolj znanih rešitev na področju poslovnega obveščanja. Uporabnikom ne zadoščajo več poročila, ki so jih uporabljali v preteklosti, pač pa potrebujejo boljši dostop do raznovrstnih podatkov: uporabniki želijo tvoriti lastna poročila, izvajati ad hoc poizvedbe, analizirati podatke ... Zato rešitev Business Objects temelji na t.i. "semantični plasti", ki uporabnikom omogoči poslovni pogled na podatke tako, da tehnološke pojme preslika v poslovne, razumljive uporabniku. S tem uporabnikom omogoča preprosto poizvedovanje, analiziranje in poročanje.

Sprotna analitična obdelava podatkov

V začetnih fazah razvoja sistemov za podporo odločanja, managerskih in direktorskih informacijskih sistemov je bil poudarek na zagotavljanju zadostne količine podatkov in informacij za odločanje (Sauter, 1997). Žal je bilo v tem času na voljo le malo podatkov v digitalni obliki, ki bi bili primerni za računalniško obdelavo, programi za njihovo obdelavo in strojna oprema pa nista bila dovolj zmogljiva. Z naraščanjem količine podatkov v računalniški obliki pa se je problem zagotavljanja zadostne *količine* podatkov prelevil v problem zagotavljanja *pravih* podatkov, to je podatkov, ki jih odločevalec v resnici potrebuje. Odločevalci se torej srečujejo z vse več podatki, ki jih za odločitve ne potrebujejo in so včasih celo zavajajoči. Managerji so dobivali na mize obsežna vnaprej pripravljena poročila, izpise iz operativnih podatkovnih baz, ki so večinoma zaradi nepreglednosti in

neprilagojenosti dejanskim potrebam obležali neuporabljeni. Po drugi strani pa so managerji potrebovali mnogo poročil, ki jih zaradi raznolikosti odločitvenih situacij, s katerimi se srečujejo, ni bilo moč vnaprej pripraviti. Jezik za poizvedovanje po podatkovnih bazah SQL je sicer postajal zaradi grafičnih uporabniških vmesnikov in programov, ki so prevajali naravnemu podoben jezik v SQL, enostaven za uporabo. Za odločevalce bi torej jezik SQL lahko bil orodje za iskanje *pravih* podatkov, to je tistih, ki jih res potrebujejo. Težava pa je v tem, ker ima SQL zaradi svoje narave preprostega jezika le omejene zmožnosti in z njim ne moremo poiskati odgovorov na vsa poslovna vprašanja. Zaradi povečane "lakote" po podatkih tudi informacijske službe niso uspele zadostiti vsem zahtevam po pripravi *ad hoc* poročil oziroma pogledov na podatke.

Odgovor na te težave je **sprotna analitična obdelava podatkov** (angl. On-Line Analytical Processing – OLAP[44]), ki omogoča neposreden (angl. on-line) dostop do podatkovnih virov in izdelavo "poljubnih"[45] pogledov na podatke. Z uporabo orodij za sprotno analitično obdelavo podatkov (v nadaljevanju orodij OLAP) je torej managerjem omogočeno:

- da si sami na *enostaven* način pripravijo pogled na podatke, kot ga za dano odločitveno situacijo potrebujejo, ter
- da z enostavnim spreminjanjem pogleda na podatke ugotavljajo, kateri podatki so zanimivi in relevantni za sprejemanje poslovnih odločitev.

OLAP torej zagotavlja predvsem veliko prilagodljivost in samostojnost pri dostopu do podatkov, vendar je izjemno pomemben predpogoj ustrezno pripravljen podatkovni vir in enostavna uporaba orodij.

---

44 V nadaljevanju uporabljamo zaradi uveljavljenosti v Sloveniji kratico OLAP, čeprav izhaja iz angleškega izraza za sprotno analitično obdelavo podatkov.

45 Poljubnost je seveda omejena na eni strani s podatki, ki jih vir vsebuje, in njihovo organizacijo, na drugi strani pa s pravicami posameznega odločevalca do posameznih pogledov na podatke.

V Merkurju že nekaj let uporabljajo t.i. komercialno analitski sistem (KAS), katerega namen je omogočiti analitikom in vsem, ki morajo vsakodnevno sprejemati odločitve na področju prodaje in upravljanja z blagom, pridobivanje informacij, ki jih potrebujejo pri svojem delu. Nekatera pogosto uporabljena poročila so že vnaprej pripravljena in uporabniki samo dostopajo do njih oziroma imajo vanje vpogled. Uporabniki pa lahko tudi samostojno izdelajo poročila, ki jih potrebujejo, in sicer v obliki, v kakršni jih potrebujejo. Vsako novo pripravljeno poročilo pa lahko tisti, ki ga je pripravil, ponudi v uporabo tudi ostalim uporabnikom. V prvi fazi so se osredotočili na podatke o prodaji ter zalogah in jih povezali s podatki o potrošnikih in poslovnih partnerjih. Sistem uporabljajo na različnih področjih v podjetju: od trženja do odnosov z dobavitelji. KAS uporabljajo pri analizah, ki so povezane z upravljanjem strukture prodaje v maloprodajni verigi, z učinkovitostjo prodajnih akcij, z upravljanjem asortimana ipd. Gre za orodje, ki daje odgovore na vprašanja kdo, kaj in zakaj v realnem času. S pomočjo tega orodja podatke iz podatkovnega skladišča spreminjajo v strateške informacije. V prihodnosti se nameravajo lotiti tudi kompleksnejših analiz v smeri iskanj odgovorov na vprašanja tipa kaj-če. KAS je zgrajen na rešitvi Microstrategy.

(Pripravil Boris Moškotelec, univ.dipl.ekon., pomočnik direktorja Marketinga, Merkur d.d.)

Podatkovni viri

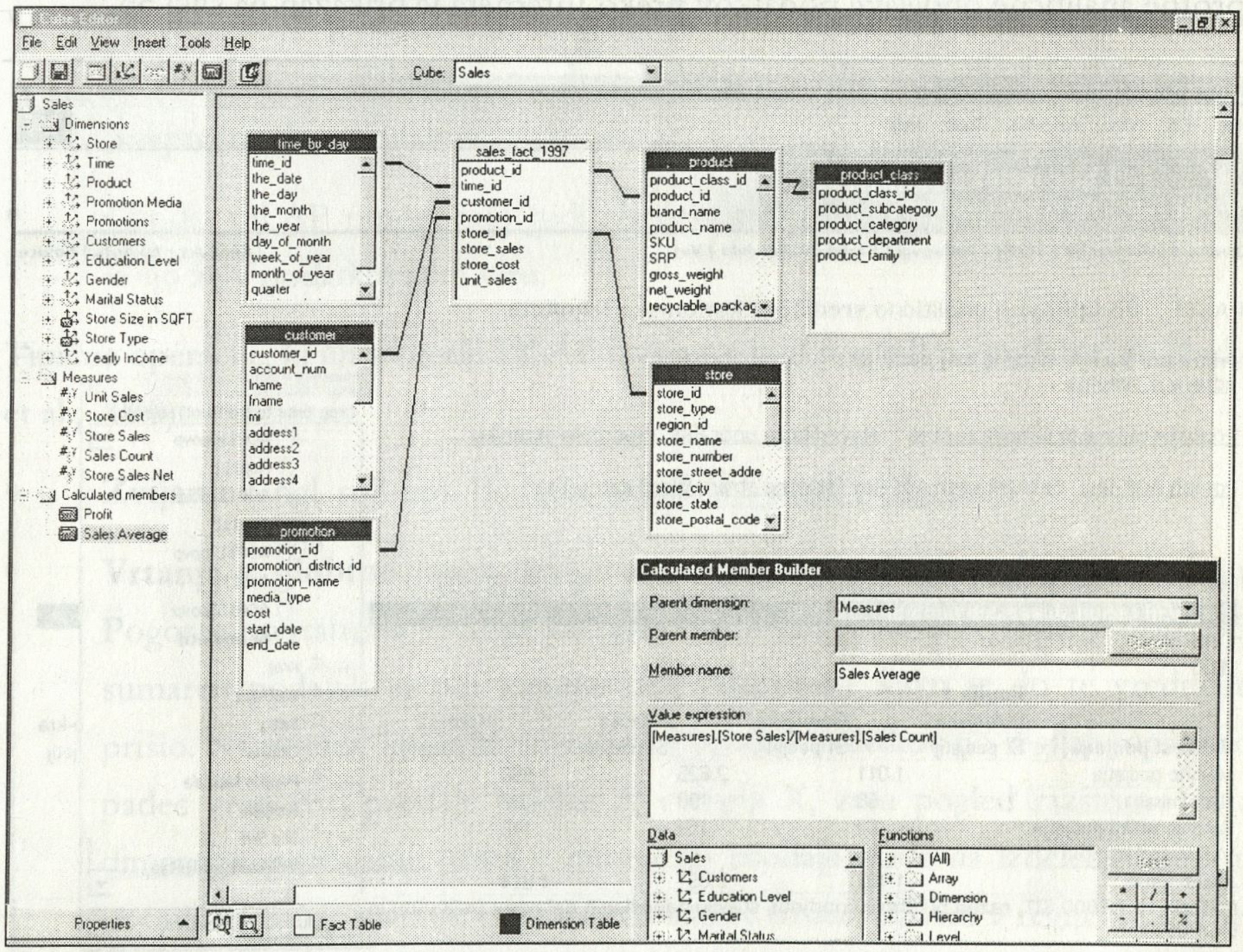
Že v začetku razvoja OLAP je bilo ugotovljeno, da skupni podatkovni viri za operativne potrebe, ki jih za razliko od OLAP imenujemo **sprotna transakcijska obdelava podatkov** (angl. on-line transaction processing – OLTP), niso ustrezni zaradi različnih razlogov (glej tudi poglavje Podatkovni viri v organizaciji):

- značilnosti obremenitev sistemov so bistveno drugačne,
- podatkovni model je v operativnih sistemih bistveno prezapleten, da bi lahko tak podatkovni vir uporabniki uporabljali samostojno,
- podatki so neintegrirani v različnih operativnih podatkovnih virih in drugo.

Iz povedanega je očitno, da je najprimernejši podatkovni vir za OLAP področno podatkovno skladišče, saj vsebuje integrirane podatke, podatkovni model pa je prilagojen potrebam analitika in praviloma večdimenzionalen ter s tem preprost za razumevanje. Redkeje se orodja OLAP uporabljajo neposredno nad podatkovnim skladiščem, predvsem zaradi zapletenosti modela in velike količine podatkov, ki ne omogoča učinkovitega dela.

Večdimenzionalni pogled na podatke, ki ga omogočajo orodja OLAP, lahko zagotovimo na različen načine, najpogosteje pa srečamo:

- ROLAP ali relacijski OLAP, kjer so sistemi za upravljanje relacijskih podatkovnih baz prilagojeni tako, da omogočajo učinkovito delo z velikimi količinami podatkov, s posebnimi orodji, ki podatke preslikajo v večdimenzionalno kocko, pa skrijemo zapletenost relacijskega modela (glej sliko 41),
- MOLAP ali večdimenzionalni OLAP (angl. Multidimensional OLAP) pa temelji na uporabi sistema za upravljanje večdimenzionalnih podatkovnih baz, torej so podatki shranjeni tako, kakor jih vidi uporabnik.



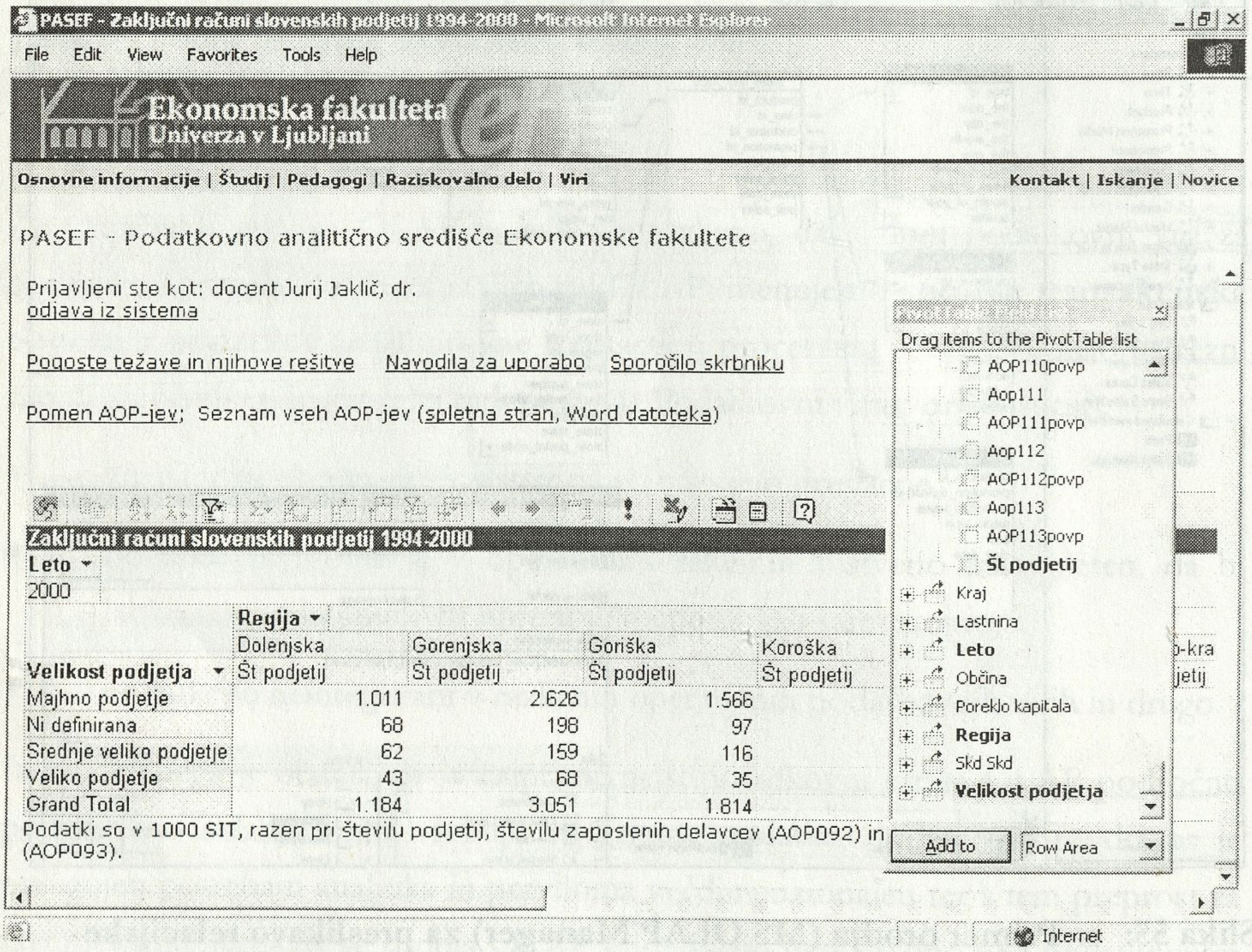
Slika 55: Primer orodja (MS OLAP Manager) za preslikavo relacijske podatkovne baze v večdimenzionalno kocko

Orodja OLAP

UTD2 - S2T4 TE2G EQ8U BB8A

Orodja OLAP omogočajo večdimenzionalen pogled na podatke, njihovi bistveni značilnosti pa sta preprosta uporaba in prilagodljivost pogleda na podatke.

Preprosta uporaba pomeni, da lahko z nekaj kliki miške dobimo poljuben prerez podatkov, praviloma v obliki vrtilne tabele (glej podpoglavje Vrtilne tabele v nadaljevanju). Zaželeno je tudi, da je delovno okolje orodja tako, v katerem se uporabnik že dobro znajde. Tako je, na primer, programski paket za delo s preglednicami MS Excel, ki ga uporabljajo mnogi analitiki, mogoče uporabiti tudi kot orodje OLAP. Orodja OLAP vedno bolj pogosto srečujemo tudi kot del spletnih aplikacij, ki jih uporabljamo v standardnih spletnih brskalnikih. Primer sprotne analitične obdelave podatkov preko Interneta je prikazan na sliki 56.



Slika 56: Uporaba sprotne anlitične obdelave podatkov preko Interneta

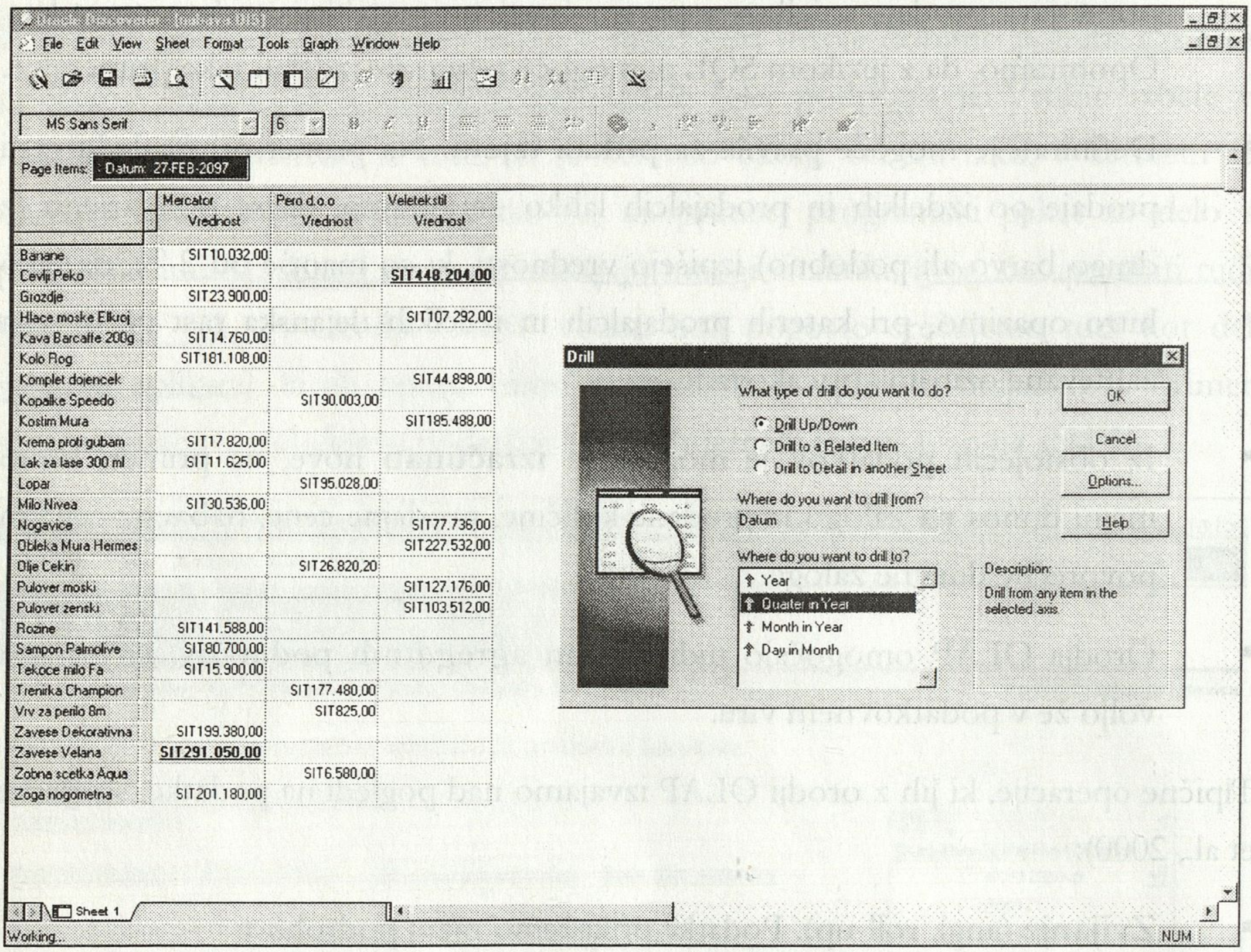
Pri izdelavi pogledov na podatke naj bi orodja OLAP ponujala predvsem naslednje možnosti:

- Enostavno izdelavo pogledov na podatke in prikaz le-teh v obliki **grafov** različnih tipov.
- Posebej pomembna je tudi možnost izdelave **primerjav**, na primer primerjava prodaje izdelkov v prvem četrtletju preteklega in letošnjega leta. Opomnimo, da z jezikom SQL ni mogoče *neposredno* izdelati take primerjave.
- Definirati je mogoče pravila za **prikaz izjem**. Na primer, v pregledu rasti prodaje po izdelkih in prodajalcih lahko definiramo, da se poudarjeno (z drugo barvo ali podobno) izpišejo vrednosti, ki so manjše od 2 %, da tako hitro opazimo, pri katerih prodajalcih in izdelkih dejanska rast ne dosega zahtevane oziroma pričakovane.
- Iz obstoječih podatkov je mogoče je **izračunati** nove, na primer, bruto mejni donos na zalogo iz prodane količine, prodajne cene, nabavne cene in povprečne dnevne zaloge.
- Orodja OLAP omogočajo tudi izračun **agregatnih podatkov**, ki niso na voljo že v podatkovnem viru.

Tipične operacije, ki jih z orodji OLAP izvajamo nad pogledi na podatke, so (Jarke et al., 2000):

- **Zvijanje** (angl. roll-up). Podatke prikažemo manj podrobno.
- **Vrtanje v globino** (angl. drill-down). Podatke prikažemo bolj podrobno. Pogosto uporabimo vrtanje, ko opazimo v tabeli zanimiv (npr. izstopajoč) sumaren podatek in nas zanima bolj podrobno, kako je do te vrednosti prišlo. Na primer, ko prikazujemo rast prodajo izdelkov po regijah, opazimo padec vrednosti prodaje izdelka A v regiji X, zato pogled razširimo še z dimenzijo prodajalne (torej v dimenzijo prodaja glede na izdelek, regijo in trgovino, da ugotovimo, če je morda katera izmed trgovin v regiji X posebej odgovorna za padec prodaje v tej regiji. Ta operacija je značilna pri iskanju odgovora na vprašanje "Zakaj?".
- **Rezanje** (angl. slice and dice). Naredimo izbor podatkov, prikažemo podkocko. Na primer, v pogledu prodaje izdelkov po regijah izberemo samo izdelke določene kategorije.

- **Vrtenje** (angl. pivot). Obračamo pogled na podatke, na primer, pogled prodaja po regijah spremenimo v pogled prodaja po izdelkih in nato v pogled prodaja po izdelkih znotraj posamezne regije.



Slika 57: Prikaz izjem in možnost vrtanja v globino v orodju Oracle Discoverer

Rudarjenje v podatkih

Čeprav je sprotna analitična obdelava zelo uporabna v mnogih primerih, ko nas zanima vpogled v podatke in želimo odkrivati zanimive informacije, pa ne omogoča *avtomatičnega* odkrivanja znanja (zanimivih informacij, vzorcev, pravil ...) v velikih količinah podatkov. Prav to pa nam omogoča **rudarjenje v podatkih** (angl. Data Mining), ki ga lahko definiramo kot proces odkrivanja vzorcev in povezav v podatkih z namenom boljših poslovnih odločitev.

Čeprav so tehnike, ki se uporabljajo pri rudarjenju v podatkih, znane že od prej, pa je razvoj rudarjenja pogojen z razvojem zmogljive računalniške opreme, saj je za

rudarjenje značilno, da iskanje pravil in vzorcev poteka avtomatično v veliki množici podatkov. Namen uporabe rudarjenja je prav razumevanje ogromnega števila podatkov, ki so bili zbrani v letih od začetkov računalniške obdelave podatkov v organizacijah.

Ker so uporabniki orodij za rudarjenje praviloma (poslovni) analitiki, je pomembna zahteva za ta orodja, da so preprosta za uporabo in da ni potrebno posebno poglobljeno znanje tehnik rudarjenja. Uporabniki torej praviloma *niso* profesionalni statistiki, matematiki, programerji in podobno.

Rudarjenje v podatkih se ponavadi omenja v zvezi s poslovno uporabo (glej npr. definicijo zgoraj), a je njegova uporaba mnogo širša in obsega, na primer:

- medicinsko diagnostiko,
- odkrivanje prevar (npr. davčne napovedi),
- nadzor onesnaževanja v elektrarnah,
- prijaznejša uporaba spleta (v veliki množici informacij na spletu dobite izbor tistih, ki vas utegnejo zanimati glede na vaše dosedanje obiske spletnih strani),
- kriminalistiko (odkrivanje vzorcev pri kriminalnih dejanjih),
- preverjanje identitete (npr. v računalniških sistemih).

Na poslovnem področju se je rudarjenje najbolj uveljavilo na področju trženja, kjer je postalo zanimivo zaradi bolj poudarjene usmeritve k posameznemu kupcu. Govorimo o **uravnavanju odnosov s kupci** (angl. Customer Relationship Management, **CRM**), ki pomeni usmerjenost na stranke, vzpostavljanje, vzdrževanje, ohranjanje ter izboljševanje odnosov z njimi. Gre za tako poslovno usmeritev podjetja oziroma način razmišljanja, ki postavlja v središče stranko. Najpomembnejša korist koncepta CRM je privabljanje boljših strank in zadržanje obstoječih dobrih oziroma donosnih strank. Seveda moramo v primeru take usmeritve podjetja stranko dobro spoznati in razumeti njeno obnašanje, kar

omogočajo kakovostni podatki o strankah[46] in ustrezni analitični postopki ter orodja. Analitični CRM (govorimo še o operativnem[47] in organizacijskem[48] CRM) je torej področje, kjer se organizacija ukvarja s postopki za doseganje temeljitega vpogleda v potrebe in želje strank, razumevanje njihovega vedenja in predvidevanje njihovih namer. Prav tu je rudarjenje izredno zmogljiva tehnologija, če ga uporabimo pravilno in če imamo dovolj kakovostnih podatkov o strankah.

Najpogosteje se rudarjenje na področju trženja uporablja za:

- neposredno trženje, npr. ponudbe pošiljamo kupcem, od katerih z večjo verjetnostjo pričakujemo odziv,
- izdelavo profilov kupcev – ugotavljamo vzorec obnašanja kupca, na podlagi le-tega pa lahko ustrezno prilagodimo ponudbo,
- segmentacijo – določanje skupin kupcev z enakimi značilnostmi (vzorcem obnašanja),
- iskanje povezav med prodajo izdelkov (analiza nakupne košarice, angl market basket analysis), kar lahko uporabimo npr. za ustrezno razporeditev izdelkov na policah,
- aktiviranju kupca,
- navzkrižnem trženju, to je stimulacija nakupa drugih izdelkov istega podjetja (angl. cross-selling), oz. več istih izdelkov (angl. up-selling), kar lahko nadomešča pridobivanje novih kupcev,
- obdržanju kupca, kar je bistveno ceneje od pridobivanja novega kupca …

Denimo, da pošiljamo reklamno gradivo za nov izdelek. Če ne nameravamo gradiva poslati vsem našim kupcem, imamo dve možnosti: naslove lahko izberemo iz trženjske baze naključno, lahko pa uporabimo rudarjenje. V slednjem primeru si

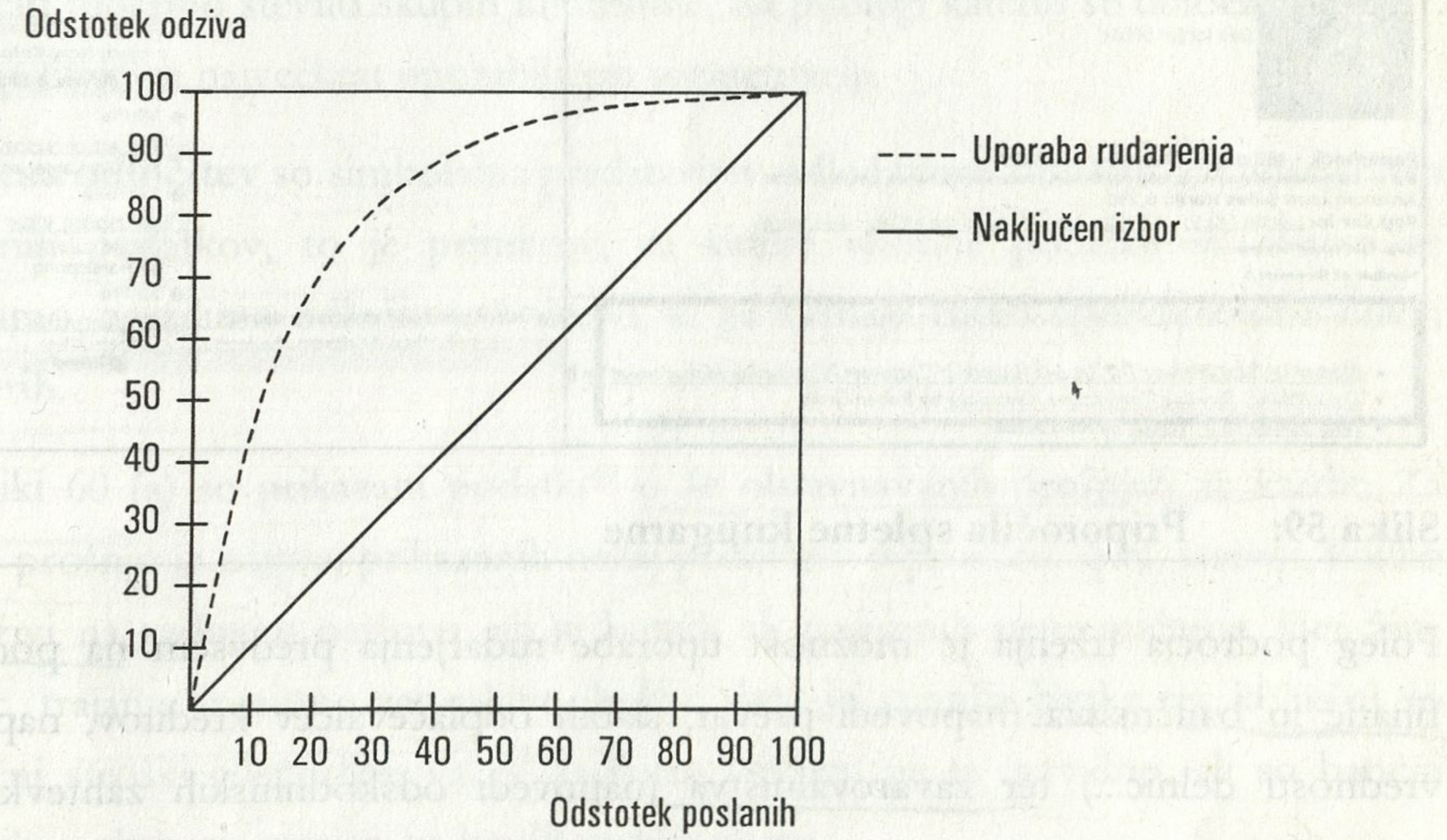
---

46 Prav pridobivanje podatkov o strankah se kaže kot eden najtežjih problemov pri analizi njihovega vedenja.

47 Operativni CRM se nanaša na izvajanje procesov, s katerimi se podjetje povezuje s strankami. Zanj je značilna prenova in informatizacija poslovnih procesov, pri katerih organizacija sodeluje s strankami.

48 Organizacijski CRM se nanaša na komuniciranje med organizacijo in strankami (preko elektronske pošte, obveščanje preko spletnih strani …)

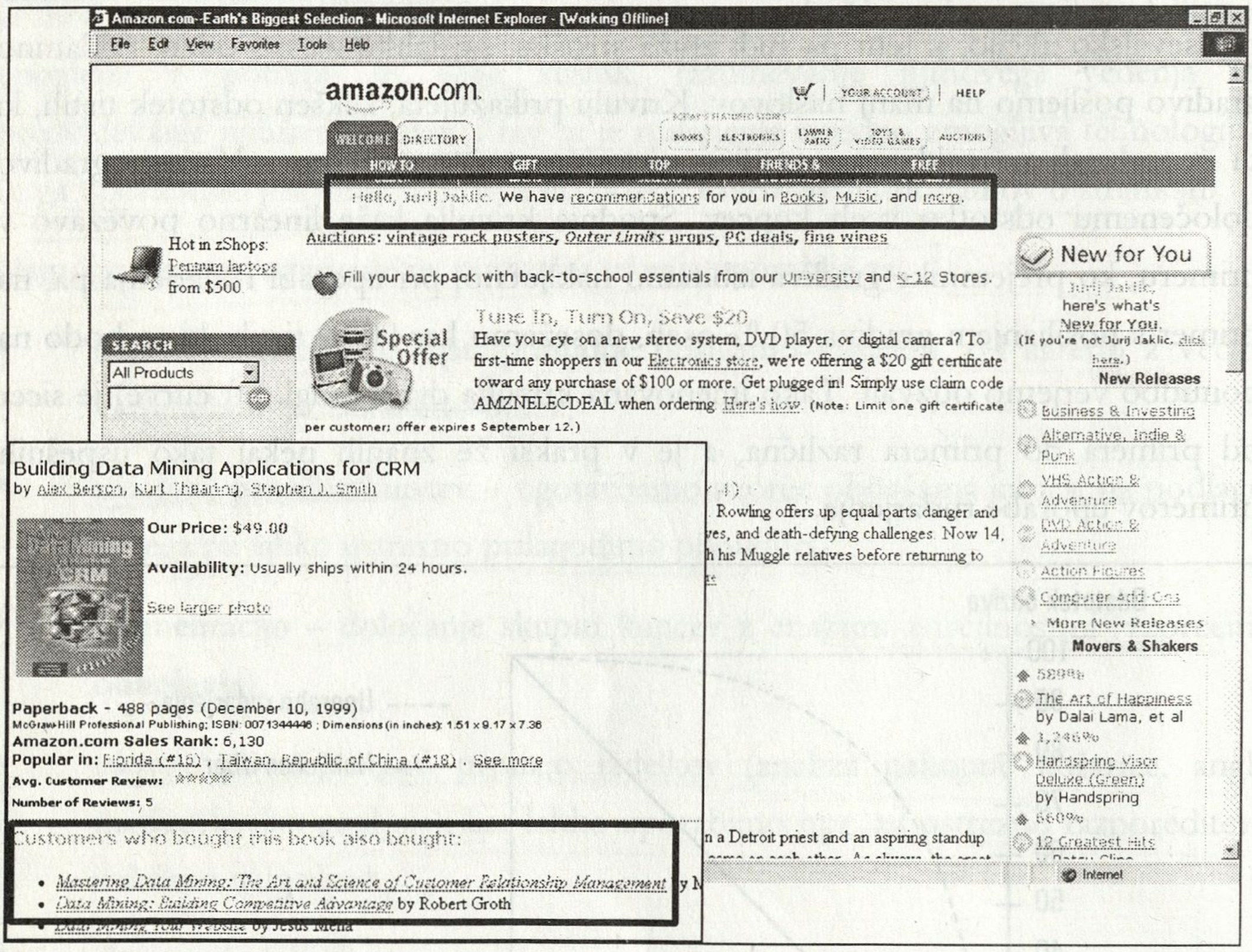
z rudarjenjem pomagamo tako, da na osnovi podatkov o dosedanjih nakupih izberemo tiste kupce, za katere je verjetnost, da se bodo odzvali (kupili nov izdelek) večja. Na sliki 58 je prikazano, kako nam rudarjenje poveča odzivnost na oglaševalsko akcijo, s tem pa tudi zniža stroške, saj lahko za isti odziv reklamno gradivo pošljemo na manj naslovov. Krivulji prikazujeta, kakšen odstotek tistih, ki bi se odzvali na reklamno gradivo, dosežemo, če pošljemo reklamno gradivo določenemu odstotku vseh kupcev. Spodnja krivulja kaže linearno povezavo v primeru, ko prejemnike gradiva izbiramo naključno, pri uporabi rudarjenja pa, na primer s pošiljanjem gradiva 50 % oseb, dosežemo kar 90 % tistih, ki se bodo na ponudbo verjetno odzvali. Tako imenovana krivulja dviga (angl. lift curve) je sicer od primera do primera različna, a je v praksi že znanih nekaj tako uspešnih primerov uporabe rudarjenja.



Slika 58: Povečanje odzivnosti z uporabo rudarjenja

Lep primer uporabe rudarjenja v podatkih so tudi priporočila naslovov, ki utegnejo zanimati obiskovalca spletne knjigarne (glej npr. http://www.amazon.com). Ta priporočila so izdelana za vsakega obiskovalca posebej glede na njegovo predhodno obnašanje, to je sprehajanje po spletnih straneh knjigarne. Druga vrsta priporočil so pri opisu knjige, ker nam se izpiše seznam knjig, za katere so se zanimali obiskovalci, ki so se tudi zanimali za to knjigo. Tovrstna priporočila

pomenijo seveda dodatno vrednost, ki za spletno knjigarno predstavlja konkurenčno prednost.



Slika 59: Priporočila spletne knjigarne

Poleg področja trženja je možnost uporabe rudarjenja predvsem na področju financ in bančništva (napovedi prevar, slabih odplačevalcev kreditov, napovedi vrednosti delnic...) ter zavarovalništva (napovedi odškodninskih zahtevkov in prevar).

Tehnike/algoritmi, ki se najpogosteje uporabljajo pri rudarjenju, so:

- najbližji sosed,
- razvrščanje v skupine,
- drevesa odločitev,
- inducirana pravila,

- nevronske mreže,
- genetski algoritmi …

Najbližji sosed (angl. Nearest Neighbor) napoveduje vrednost spremenljivke za neko (opazovano) enoto tako, da pogleda vrednost iste spremenljivke za tiste enote, ki so "najbližje" opazovani enoti. Predpostavka je, da imajo enote, ki so blizu skupaj, podobne vrednosti spremenljivk. Seveda moramo pri tej tehniki določiti, kaj pomeni oziroma kako merimo "bližino". To je ena najstarejših tehnik, ki se uporablja pri rudarjenju.

Pri **razvrščanju v skupine** (angl. Clustering) gre za razvrstitev enot v skupine tako, da je znotraj skupin dosežena kar največja homogenost (enote so čimbolj skupaj), hkrati pa so skupine med seboj čimbolj heterogene (čimbolj narazen). Določiti moramo število skupin in atribute, na podlagi katerih se določa "bližina". Ta algoritem se največkrat uporablja pri segmentaciji.

Drevesa odločitev so simbolična predstavitev odločitvene situacije. Z rudarjenjem iz danih podatkov, to je primerov, za katere vhodne podatke in rešitve že poznamo, zgradimo odločitveno drevo, ki ga kasneje lahko uporabimo na novih primerih.

Na sliki 60 (a) so prikazani podatki[49] o že obravnavanih prošnjah za kredit. Za vsako prošnjo je najprej prikazanih nekaj podatkov o prosilcu: spol, starost, koliko časa živi na zadnjem naslovu, ali je lastnik ali najemnih nepremičnine, kjer živi, poklic, trajanje trenutne zaposlitve, koliko časa je stranka banke ter kolikšni so mesečni stroški gospodinjstva. V zadnjem stolpcu pa je razvidno, ali so bančni uradniki – eksperti, prošnji za kredit ugodili ali ne.

Z uporabo rudarjenja lahko iz teh podatkov izluščimo odločitvena pravila, ki so jih uradniki uporabljali pri obravnavanju prošnje. Z drugimi besedami, iz podatkov izluščimo znanje ekspertov. S slike 60 (b) vidimo, da je bil pri obravnavi najpomembnejši kriterij, koliko časa je prosilec stranka banke; meja je tri leta. Če je prosilec stranka banke več kot tri leta, je bila prošnja praviloma odobrena. Če pa je

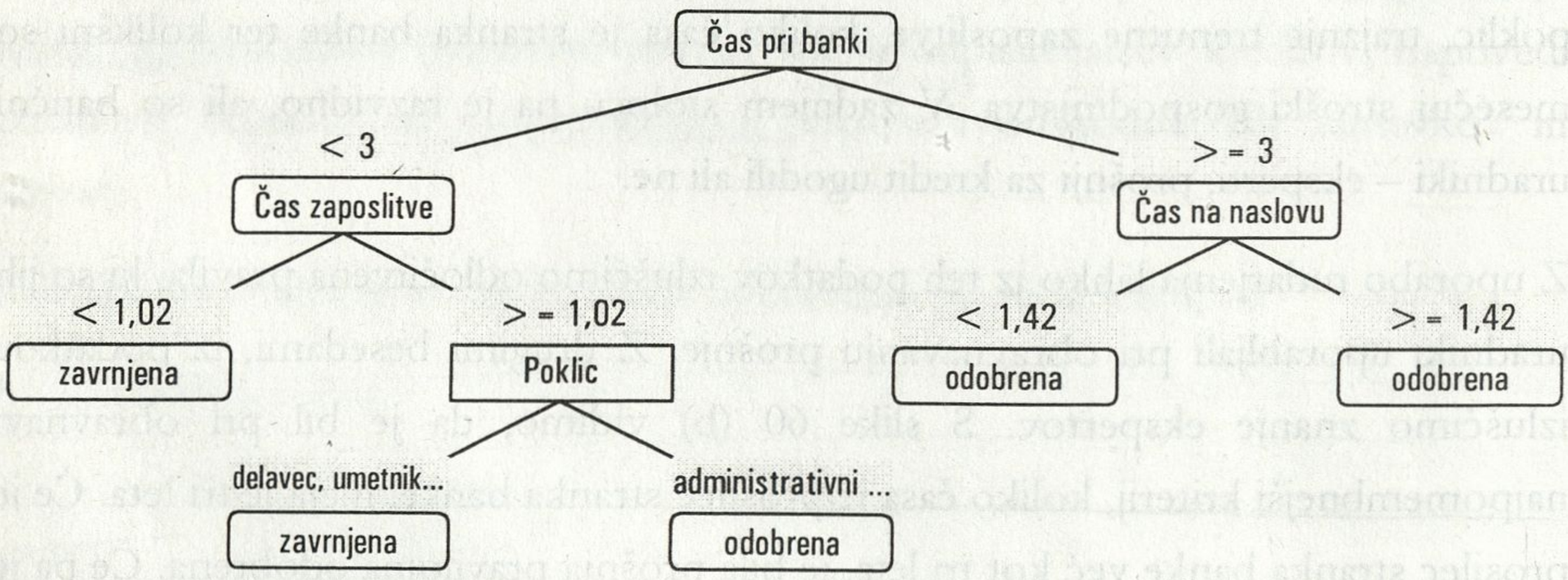
---

49 Primer je povzet po (Al-Attar, 1999).

prosilec pri banki manj kot tri leta, je bil naslednji kriterij trajanje trenutne zaposlitve. In tako dalje. Odločitvena pravila beremo tako, da se sprehodimo od korena drevesa (na sliki na vrhu) do listov, na primer, "Če je prosilec pri banki manj kot 3 leta in traja njegova trenutna zaposlitev manj kot 1,02 leti, je bila prošnja *praviloma* zavrnjena." Odstotek primerov, ki izpolnjujejo pogoje pravila (so pri banki manj kot 3 leta in traja njegova trenutna zaposlitev manj kot 1,02 leti) in rešitev ustreza pravilu (prošnja zavrnjena), imenujemo **zaupanje**. Dobljeno drevo odločanja lahko zdaj uporabimo za (delno) avtomatizacijo obravnave prošenj.

Spol	Starost	Čas na naslovu	Prebivališče	Poklic	Čas zaposlitve	Čas pri banki	Mesečni stroški	Odločitev
M	50	0,5	lastnik	nezaposlen	0	0	145	zavrnjena
M	19	10	najeto	delavec	0,8	0	140	zavrnjena
Ž	52	15	lastnik	pisatelj	5,5	14	000	odobrena
M	22	2,5	najeto	učitelj	2,6	0	000	odobrena
M	29	13	lastnik	voznik	0,5	0	228	zavrnjena
Ž	16	0,3	lastnik	nezaposlen	0	1	160	zavrnjena
M	23	11	lastnik	ekonomist	0,5	1	100	odobrena
Ž	27	3	lastnik	manager	2,8	1	280	zavrnjena
Ž	19	5,4	lastnik	paznik	0,3	0	080	zavrnjena
Ž	27	0,3	lastnik	manager	0,1	1	272	zavrnjena
M	34	4	najeto	paznik	8,5	7	195	odobrena
M	20	1,3	najeto	delavec	0,1	0	140	zavrnjena
M	34	1,3	lastnik	trgovec	0,1	0	440	zavrnjena
...	...	...	...	...	...	...	...	...

(a)



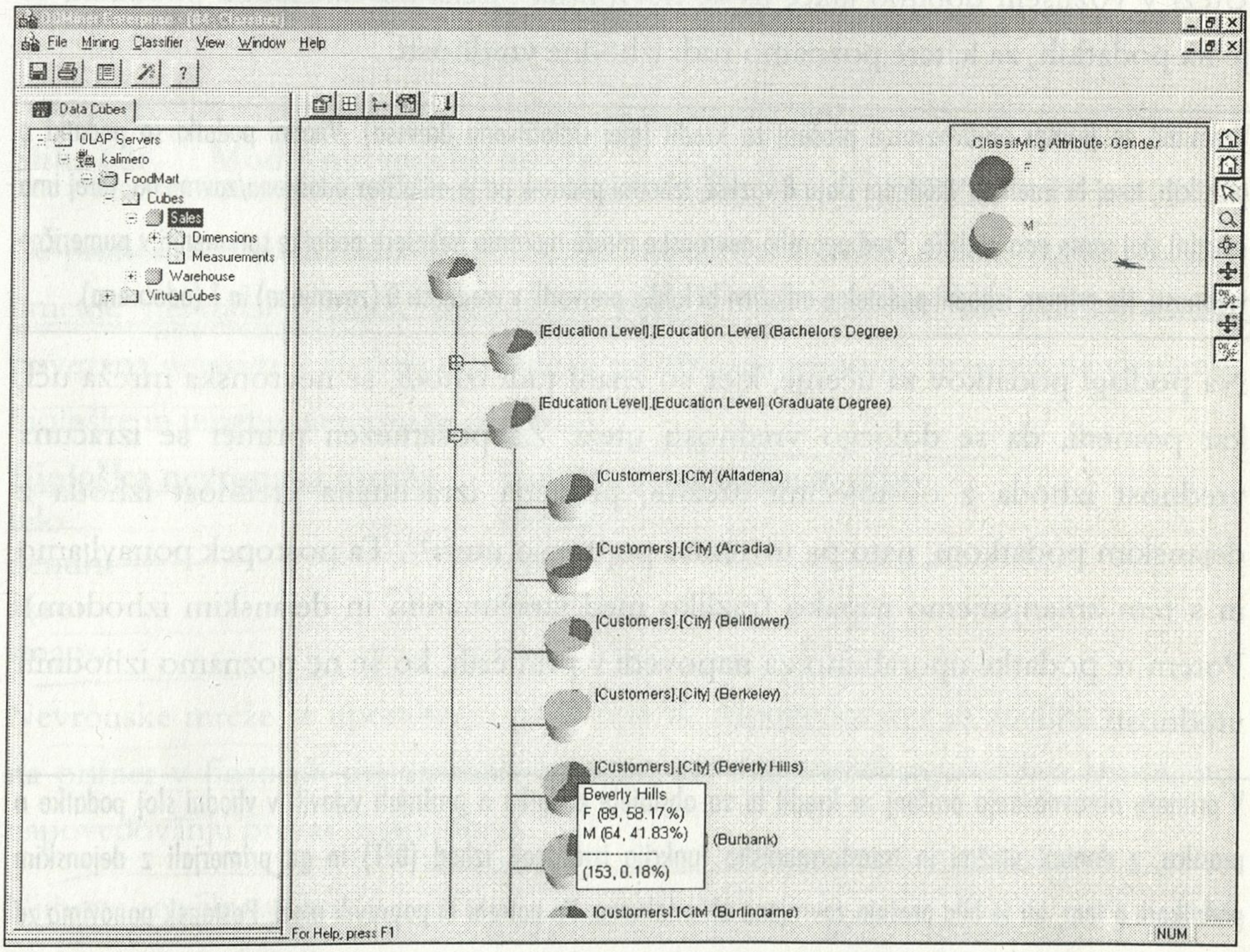
(b)

Slika 60: Podatki (a) in pripadajoče drevo odločanja (b)

Inducirana pravila imajo podoben namen kot odločitvena drevesa, le da so odločitvena pravila prikazana v obliki ČE-POTEM pravil:

ČE A_1 & A_2 & ... & A_n POTEM B_1 & B_2 & ... & B_m

kar preberemo: "Če se zgodi A_1...A_n, potem se v okviru iste transakcije *pogosto* zgodi tudi B_1...B_m". Velja, da lahko vsako odločitveno drevo izrazimo kot niz pravil, ne velja pa obratno. To pomeni, da imajo inducirana pravila nekoliko večjo izrazno moč, vendar imajo odločitvena drevesa prednost z vidika predstavitve.



Slika 61: Primer orodja za rudarjenje v podatkih DB Miner

Inducirana pravila lahko razdelimo na tri vrste, glede na to, ali povezujejo podatke iz iste dimenzije (o večdimenzionalnem modelu glej več v poglavju Podatkovne baze):

znotraj-dimenzionalno pravilo: kupec.država="Slovenija": izdelek.naziv="barva" ⇒ izdelek.naziv="čopič"

med-dimenzionalno pravilo: kupec.država="Slovenija" ⇒ izdelek.kategorija="kava"

hibridno pravilo: kupec.država="Slovenija": izdelek.naziv="barva" ⇒ izdelek.naziv="čopič" & čas = "pomlad"

Nevronske mreže naj bi posnemale delovanje človeških možganov.

Umetne nevronske mreže so množica povezanih vozlišč, ki so razporejeni v več slojev (glej sliko 62). Vozlišča nevronske mreže vsebujejo uteži, ki odražajo relativno moč posameznih vrednosti, ki so vhod v vozlišče. Z uporabo transformacijske funkcije se uteženo povprečje vhodnih podatkov pretvori v izhodni podatek, ki je nato vhod v vozlišča na naslednjem nivoju, razen na izhodnem sloju, kjer je to izhod iz nevronske mreže – rešitev problema.

Uteži v vozliščih dobimo tako, da se nevronska mreža uči na testnih podatkih, to je na podatkih, za katere poznamo tudi izhodne vrednosti.

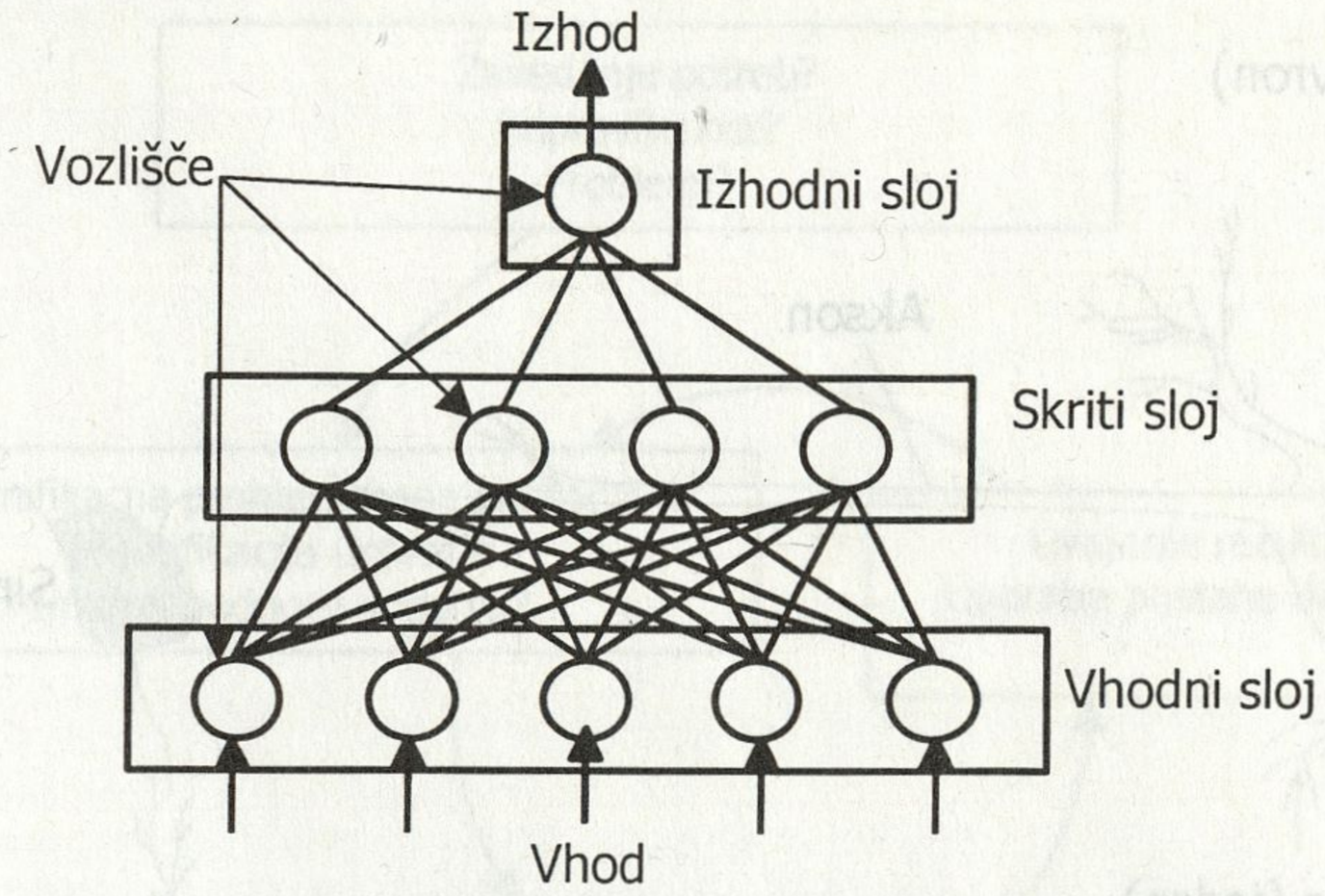
Vzemimo za primer obravnavanje prošenj za kredit (glej Odločitvena drevesa). Vhodni podatki so podatki o prosilcih, torej bi imeli na vhodnem sloju 8 vozlišč, izhodni podatek pa je odločitev odobrena/zavrnjena, torej ima izhodni sloj samo eno vozlišče. Pred uporabo nevronske mreže moramo nekatere podatke spremeniti v numerične vrednosti. Na primer, izhodni podatek o odločitvi bi lahko pretvorili v vrednosti 0 (zavrnjena) in 1 (odobrena).

Na podlagi podatkov za učenje, kjer so znani tudi izhodi, se nevronska mreža uči, kar pomeni, da se določijo vrednosti uteži. Za posamezen primer se izračuna vrednost izhoda z obstoječimi utežmi, primerja izračunana vrednost izhoda z dejanskim podatkom, nato pa ustrezno popravijo uteži[50]. Ta postopek ponavljamo in s tem zmanjšujemo napako (razliko med izračunanim in dejanskim izhodom). Potem te podatke uporabimo za napovedi v primerih, ko še ne poznamo izhodnih vrednosti.

V primeru obravnavanja prošenj za kredit bi za obstoječe podatke o prošnjah vstavili v vhodni sloj podatke o prosilcu, z danimi utežmi in transformacijsko funkcijo izračunali izhod (0/1) in ga primerjali z dejanskim podatkom o tem, ali je bila prošnja zavrnjena ali odobrena. Po potrebi bi popravili uteži. Postopek ponovimo za vse podatke v množici podatkov za učenje.

---

50 Postopkov za popravljanje uteži poznamo veliko, najpogosteje se uporablja vzvratno razširjanje (angl. backpropagation).



Slika 62: Model nevronske mreže

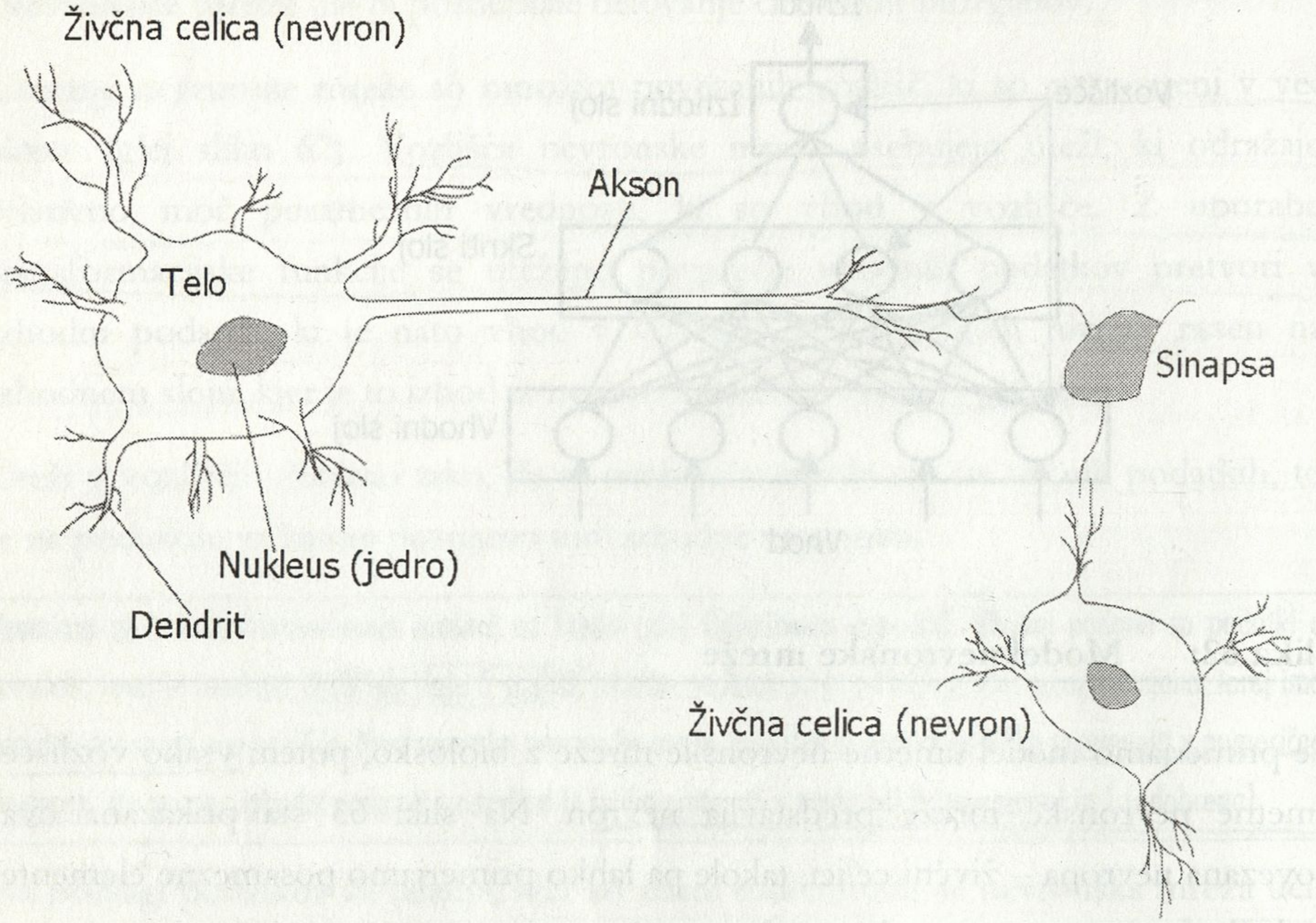
Če primerjamo model umetne nevronske mreže z biološko, potem vsako vozlišče umetne nevronske mreže predstavlja nevron. Na sliki 63 sta prikazana dva povezana nevrona – živčni celici, takole pa lahko primerjamo posamezne elemente biološke in umetne nevronske mreže:

Biološka nevronska mreža	**Umetna nevronska mreža**
telo	vozlišče
dendrit	vhod
akson	izhod
sinapsa	utež

Nevronske mreže se uporabljajo predvsem za napovedovanje in analizo tveganja, na primer v financah pri analizah tveganja, pri napovedovanju odziva kupca, pri napovedovanju prevar in podobno.

Njihove značilnosti so:

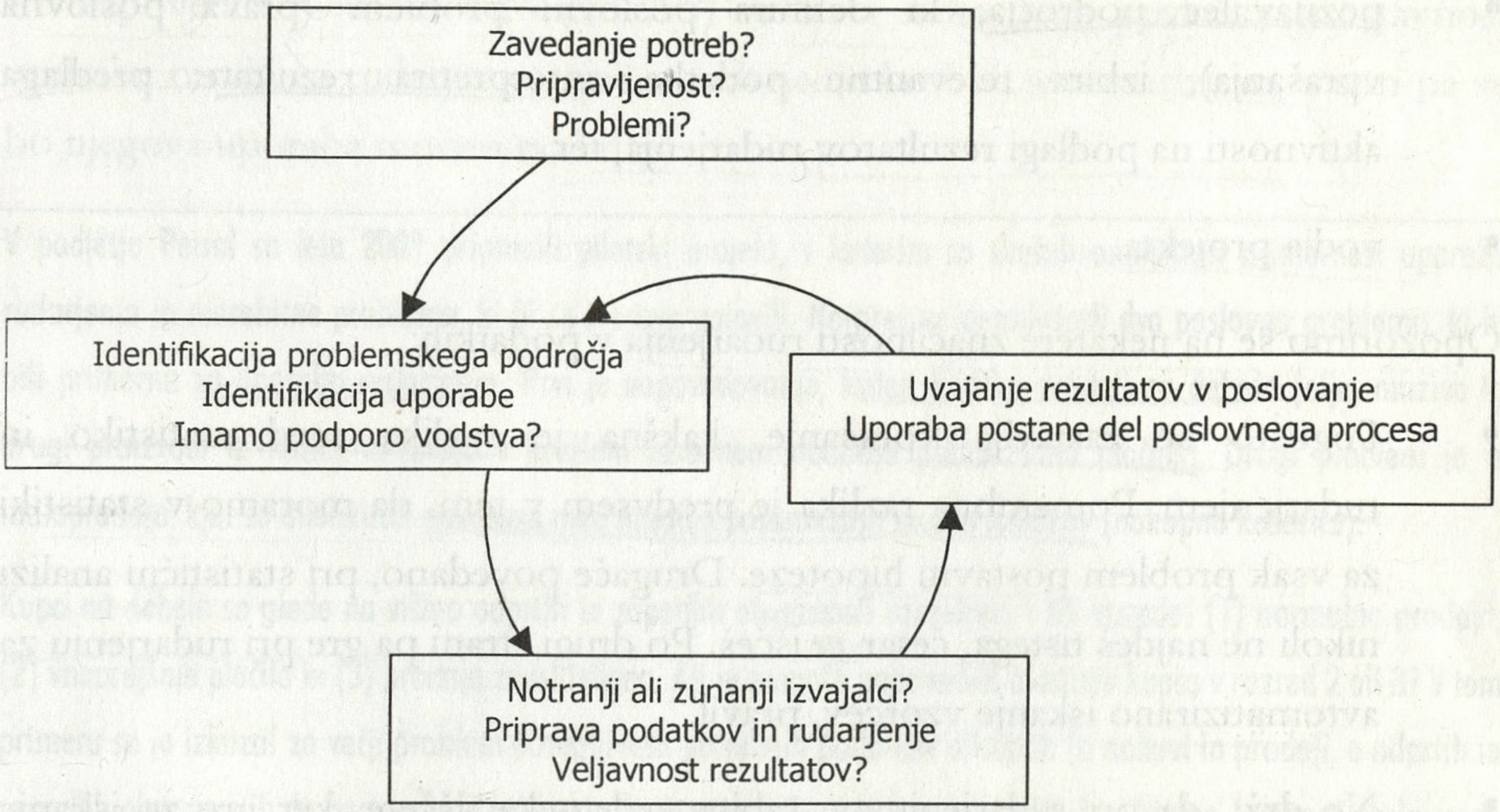
- velika natančnost,
- dolgi časi "učenja"
- vhodni podatki morajo biti numerični (v nasprotnem primeru je prej potrebna pretvorba).



Slika 63: Živčni celici in njuna povezava

Čeprav je ideja, da bi programska oprema posnemala načela biološkega razvoja, že stara, pa se je uporaba **genetskih algoritmov**, ki temeljijo na tej ideji, razširila šele v zadnjih letih. Genetski algoritmi se uporabljajo za optimizacijo, strojno učenje, pri reševanju ekonomskih problemov …

Zavedati se moramo, da rudarjenje v podatkih ne pomeni zgolj uporabe orodja za rudarjenje, temveč ga moramo opazovati skozi celoten proces, kot je prikazan na sliki 64. Zelo pogosta napaka pri rudarjenju je slabo definiranje poslovnega problema, ki ga želimo z rudarjenjem rešiti. V tem primeru je le malo verjetno, da bo rudarjenje v poslovnem smislu uspešno, čeprav dobimo dobre in veljavne rezultate same analize.



Slika 64: Proces rudarjenja v podatkih

Izkušnje praktične uporabe kažejo, da največji del vloženega napora v okviru procesa zbiranje in priprava potrebnih podatkov. Odvisno od kakovosti podatkov in podatkovne arhitekture v organizaciji lahko vloženo delo v to fazo dosega celo 50 % in več dela vloženega v celotni proces. Predpriprava podatkov obsega: združevanje podatkov iz različnih virov, opredelitev izračunanih atributov, obravnava manjkajočih vrednosti, pretvorba nenumeričnih v numerične vrednosti (nekateri algoritmi zahtevajo numerične vrednosti), obravnava nekonsistentnih podatkov in osamelcev, odstranitev šuma, normalizacija podatkov, po potrebi dodajanje novih atributov …

Iz faz procesa je razvidno, da rudarjenja ne more izvajati samo nekdo, ki dobro pozna delo z orodjem za rudarjenje, pač pa gre za interdisciplinarno področje. Zaželeno je, da bi v procesu rudarjenja sodelovali:

- informatik skrbi za pripravo podatkov, za lociranje le-teh, združevanje …,
- podatkovni analitik (npr. statistik) skrbi za primernost metod, za izbor primernih podatkov …,

- poznavalec področja, ki definira poslovni problem (prava poslovna vprašanja), izbira relevantne podatke, interpretira rezultate, predlaga aktivnosti na podlagi rezultatov rudarjenja, ter
- vodja projekta.

Opozorimo še na nekatere značilnosti rudarjenja v podatkih:

- Pogosto se zastavlja vprašanje, kakšna je razlika med statistiko in rudarjenjem. Pomembna razlika je predvsem v tem, da moramo v statistiki za vsak problem postaviti hipoteze. Drugače povedano, pri statistični analizi nikoli ne najdeš tistega, česar *ne* iščeš. Po drugi strani pa gre pri rudarjenju za avtomatizirano iskanje vzorcev, pravil.
- Ne drži, da pri rudarjenju ne rabite vedeti, kaj iščete, ker gre za iskanje neznanih vzorcev. Res je, da programska oprema poišče neznane vzorce, vendar mora analitik najprej formulirati problem (kar ni isto kot postavljanje hipotez; glej prvo točko).
- Najpogostejše napake neizkušenih analitikov so: ponovno odkrivanje znanja, nepoznavanje problemskega področja in neustrezna priprava podatkov.

Na primer, z rudarjenjem so ugotovili, da so krediti praviloma pogosteje odobreni v primeru pisnih vlog kot v primeru telefonskih klicev. Šele, ko so v proces vključili poznavalca področja, je le-ta povedal, da se pogosto zgodi, da pri telefonskih klicih, pri katerih prošnja očitno ne ustreza zahtevam, to prosilcu povedo, ne da bi zabeležili njegov klic – prošnjo. Pri pisnih prošnjah pa vsako zabeležijo, ne glede na to, da morebiti že na prvi pogled ne ustreza.

- Opozorimo še na pravne vidike rudarjenja, saj lahko z rudarjenjem ugotovimo in kasneje uporabimo take informacije, ki bi lahko posegale v pravice posameznika (npr. kupca) do varovanja osebnih podatkov. Področje rudarjenja je posebej problematično zato, ker nikoli ne vemo, kaj bomo z njim odkrili in kako to uporabili (to je pravzaprav bistvo rudarjenja), zakonodaja pa dovoljuje zbiranje podatkov o posamezniku le, če je vnaprej določeno, v kakšne namene bomo podatke uporabljali.

Po napovedih (Gartner Group, 1999) se bo v nekaj letih povečala enostavnost uporabe orodij za rudarjenje, proces rudarjenja bo bolj avtomatiziran, s tem pa se bo njegova uporaba močno razširila.

V podjetju Petrol so leta 2001 pripravili pilotski projekt, s katerim so skušali analizirati primernost uporabe rudarjenja in morebitne probleme, ki bi se pri tem pojavili. Najprej so identificirali dva poslovna problema, ki bi bili primerna za uporabo rudarjenja: Prvi je napovedovanje, kateri kupci v prodaji na debelo (olja, maziva in drugi proizvodi iz nafte) utegnejo v krajšem časovnem obdobju postati slabi plačniki. Drugi problem je iz maloprodaje, kjer so analizirali povezave med prodajo posameznih skupin izdelkov (nakupna košarica).

Kupci na debelo so glede na višino odprtih in zapadlih obveznosti razdeljeni v tri razrede: (1) normalna prodaja, (2) vnaprejšnje plačilo in (3) prodaja zaustavljena. Ali je mogoče napovedati uvrstitev kupca v razred 2 ali 3? V tem primeru se je izkazal za večji problem dosegljivosti potrebnih podatkov o kupcih (o nabavi in prodaji, o odprtih in zapadlih obveznostih do partnerja in partnerja do organizacije, o prometu v dobro in v breme, lastnosti poslovnega partnerja), iz katerih bi sklepali na spremembo uvrstitve v razred. Potrebni podatki so se nahajali v različnih virih, vnos nekaterih podatkov ni bil obvezen, kakovost podatkov je bila slaba, nekatere podatke o partnerjih je bilo potrebno pridobiti iz zunanjih virov. Po pripravi podatkov, oblikovanju modela in uporabi orodja za rudarjenje (IBM Intelligent Miner) so oblikovali listo 70 partnerjev, ki naj bi prišli v razred 2 ali 3, kasneje se je izkazalo, da se jih je od teh dejansko uvrstilo v ta dva razreda 14. Pomembni stranski produkti tega projekta so bili: ozaveščanje in informiranje o sodobnih informacijskih tehnologijah rudarjenja in njihovi uporabnosti, izboljšanje postopka zbiranja podatkov o poslovnih partnerjih in izboljšanje kakovosti samih podatkov.

V drugem primeru, kjer so analizirali račune iz maloprodaje (trgovine HIP-HOP) in odkrivali povezave med prodajo skupin izdelkov, je bilo stanje glede kakovosti podatkov bistveno drugačno, saj imajo vse podatke iz vseh trgovin zbrane v osrednji bazi podatkov. Čeprav so bili podatki kakovostni, je bila potrebna njihova predpriprava za postopek rudarjenja (npr. izločitev "zaokroževanja" kot postavke računov). Projekt je potekal v sodelovanju s tržniki. Pravila, ki so jih z rudarjenjem našli, so tržniki primerjali z intuitivnim poznavanjem. Nekatere odkrite povezave so bile tržnikom že prej znane, odkrite pa so bile tudi nekatere presenetljive povezave, kar je pravzaprav cilj rudarjenja. Na primer, presenetljivo je bilo pravilo, da so "artikle iz skupine hrana za gospodinjstvo kupili kupci, ki so kupili izdelke iz več skupin kot povprečni kupci". Odkrili so tudi močne povezave med nakupom sladkih in slanih izdelkov ter pijačami ter da sodi skupina alkoholnih pijač med skupine z največjim vplivom na prodajo ostalih artiklov.

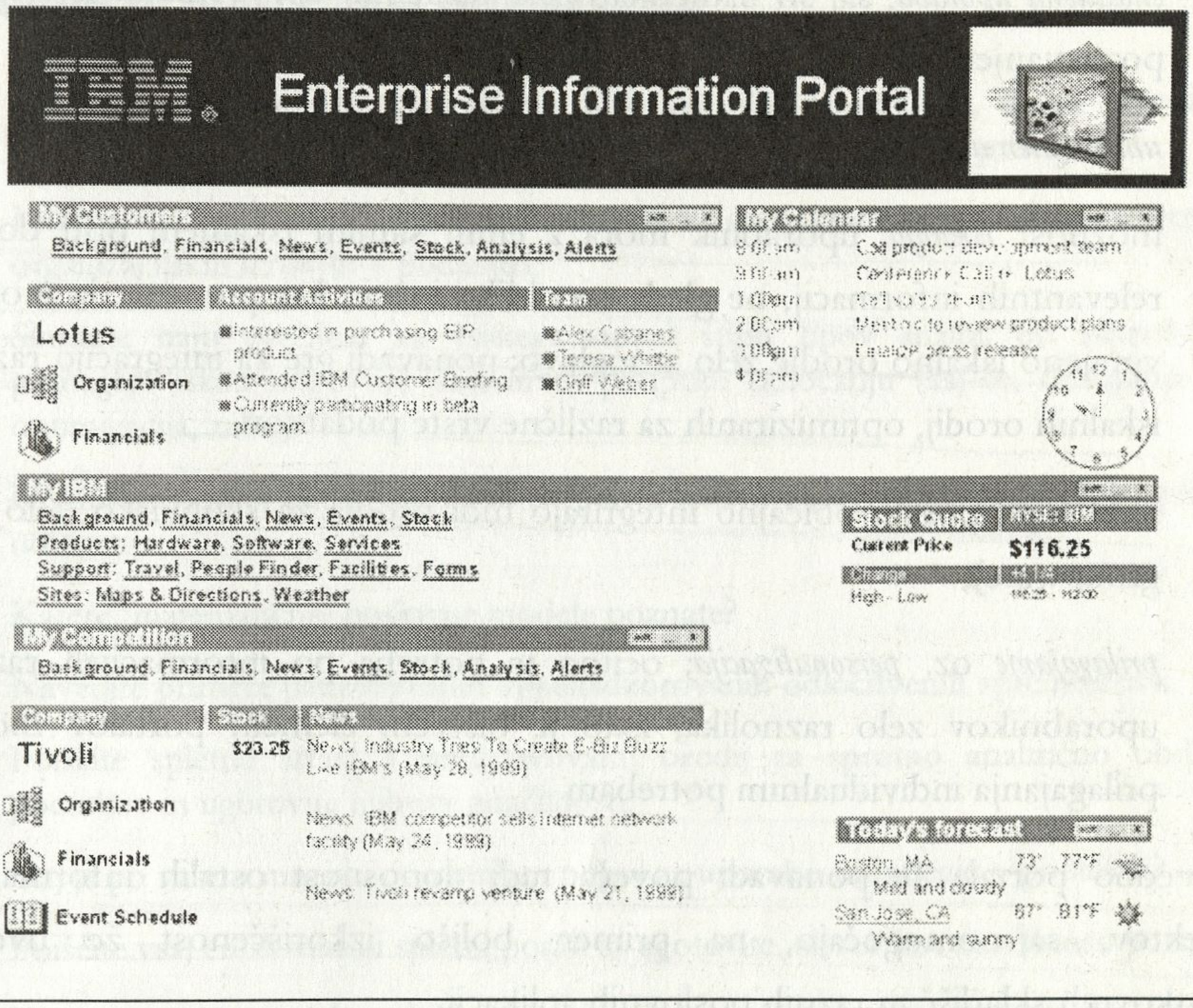
(Pripravil mag. Sandi Pogačnik, Petrol d.d.)

4.4 Poslovni portali

Kljub uvedbi integriranih podatkovnih virov, kot so npr. podatkovna skladišča ali operativne shrambe podatkov, pa se v mnogih organizacijah še vedno dogaja, da uporabniki nimajo učinkovitega dostopa do vseh poslovnih informacij, ki jih potrebujejo za svoje delo. Razlog tiči v tem, da so prej omenjeni integrirani podatkovni viri namenjeni predvsem shranjevanju strukturiranih podatkov, v organizacijah pa obstaja množica nestrukturiranih podatkov, kot so e-pošta, faks, spletni dokumenti, digitalizirane slike, dokumenti, pripravljeni v urejevalnikih besedil, poskenirani dokumenti, zvočni in video dokumenti, predstavitve in drugo. Seveda so tovrstne informacije prav tako pomembne za učinkovito delo in poslovno odločanje kakor informacije, pridobljene iz strukturiranih podatkovnih virov. Ne gre pa pozabiti tudi na zunanje nestrukturirane podatke (npr. grafični prikaz gibanja tečaja delnice konkurenčnega podjetja, vremenska napoved z animirano satelitsko sliko in podobno). Mnogokrat uporabniki izgubljajo mnogo dragocenega časa za iskanje tovrstnih informacij.

Poslovni portal je povezava različnih virov informacij iz poslovnih strukturiranih podatkovnih virov (podatkovno skladišče, področno podatkovno skladišče, operativne podatkovne baze ...), nestrukturiranih podatkovnih virov (sistemov za upravljanje z dokumenti ...), programskih rešitev (OLAP ...) ter zunanjih podatkovnih virov (predvsem z Interneta) v enotno vstopno točko, preko katere do njih dostopamo z uporabo (spletnih) brskalnikov. Gre torej za prikaz za uporabnika bistvenih informacij združenih na enem mestu (enem zaslonu).

Seveda na enem mestu ni mogoče hkrati prikazati *vseh* informacij, ki jih potrebuje uporabnik, zato tudi portali omogočajo vrtanje v globino (glej podpoglavje Sprotna analitična obdelava podatkov). Če je, na primer, na uvodnem zaslonu portala grafično prikazano gibanje prodaje v zadnjih dneh, lahko s klikom na graf pridemo do bolj podrobnih (detajlnih) podatkov o prodaji. Ali, če je prikazan naš današnji urnik, kjer je zabeleženo, da imamo poslovni sestanek, lahko s klikom na ustrezno ikono pridemo do gradiv, ki jih za sestanek potrebujemo.



Slika 65: Primer poslovnega portala

Poslovni portal torej združuje zelo različne koncepte in informacijske tehnologije; za uporabnika postane transparentno, za katero tehnologijo gre in iz katerega podatkovnega vira pridobiva informacije. Uporabniki pričakujejo od portalov predvsem (Hrvatin, 2000):

- enostavno iskanje in dostop do poslovnih informacij, ki jih potrebujejo za učinkovito opravljanje svojega dela,
- smiselno povezovanje informacij v celoto,
- iskanje oseb, ki lahko v določenem primeru ponudijo pomoč,
- boljšo lastno organiziranost,
- možnost povezovanja s sodelavci ter
- povezanost poslovnih procesov.

Poglavitne značilnosti poslovnih portalov so:

- *enostavna uporaba*, saj so namenjeni zelo različnim uporabnikom z različnim poznavanjem informacijske tehnologije,
- *univerzalen dostop do podatkovnih virov*,
- možnost *iskanja*; uporabnik mora z enim samim iskanjem priti do vseh relevantnih informacij, ne glede na obliko in vir, kar pomeni, da mora biti vgrajeno iskalno orodje zelo zmogljivo; ponavadi gre za integracijo različnih iskalnih orodij, optimiziranih za različne vrste podatkov,
- *sodelovanje*; portali običajno integrirajo tudi orodja za skupinsko delo (angl. groupware),
- *prilagajanje* oz. *personalizacija*; očitno je potreba po informacijah različnih uporabnikov zelo raznolika, zato je bistveni element portalov možnost prilagajanja individualnim potrebam.

Z uvedbo portala se ponavadi poveča tudi donosnost ostalih informacijskih projektov, saj omogočajo, na primer, boljšo izkoriščenost že uvedenih podatkovnih skladišč in raznih poslovnih aplikacij.

Poslovne portale lahko, glede na namembnost, razdelimo v tri skupine:

- **portali za zaposlene** (angl, business to employee – B2E) omogočajo dostop do poslovnih informacij podjetja, do poslovnih aplikacij, izboljšuje povezave med zaposlenimi, izboljšuje poslovno odločanje ter dviguje kakovost poslovnih procesov,
- **portali za elektronsko poslovanje** (angl. business to business – B2B) so namenjeni zunanjim partnerjem (angl. business to business), povečujejo učinkovitost v celotni poslovni verigi od dobaviteljev, distributerjev do končnih kupcev,
- **portali za kupce** (angl. business to customer – B2C), katerih namen je kupcem posredovati informacije o izdelkih in storitvah.

Poleg poslovnih portalov so nekoliko dlje znani tudi **spletni portali**, kjer so prav tako na enem mestu zbrane različne informacije od novic do zanimivih spletnih povezav. Tudi za spletne portale velja, da si jih lahko obiskovalec prilagaja oziroma da si določi, katere informacije želi videti.

Vprašanja za preverjanje razumevanja

- Zakaj v organizacijah uporabljamo informacijske tehnologije, ki ne povečujejo bistveno informacijske asimetrije (npr. informatizacija poslovnih procesov)?
- Kakšna je povezava med informacijsko demokratizacijo in spremembami organizacijskih struktur v podjetjih?
- Skušajte najti problem za vsakega izmed štirih tipov analiz, pri katerih nam pomagajo iskati odgovore sistemi za podporo odločanju (kaj-če, doseganje cilja, optimizacija, zakaj).
- Razmislite, kje bi lahko uporabili ekspertni sistem za avtomatizacijo (dela) procesa odločanja.
- Katere (matematične) poslovne modele poznate?
- Navedite primere nadzorovanih in nenadzorovanih odločitvenih spremenljivk.
- Poiščite spletne strani s predstavitvami orodij za sprotno analitično obdelavo podatkov in ugotovite njihove značilnosti.
- Kaj je po vašem mnenju pomemben pogoj za uporabo rudarjenja v podatkih?
- Poiščite vsaj en slovenski spletni portal in ugotovite, ali res ima značilnosti portala.
- Katere vrste modelov lahko rešujemo v programskih paketih za delo s preglednicami?

Ključni pojmi

analiza podatkov
analiza problema
časovna vrsta
direktorski informacijski sistem
donos na vlaganja v informatiko
doseganje cilja
ekspertni sistem
informacijska demokratizacija
informacijska podpora managementu
kaj-če analiza
managerski informacijski sistem
model
modeliranje
napovedovanje
odkrivanje znanja
odločanje
odločitveni proces
OLAP
OLTP
optimizacija
poslovni model
poslovni portal
poslovno obveščanje
preglednica
proces odločanja
rudarjenje v podatkih
sistem za podporo odločanju
sprotna analitična obdelava podatkov
sprotna transakcijska obdelava podatkov
strukturiranost problema
vrtilna tabela

Literatura

[1] Adam N. R., Wortmann J. C.: Security-Control Methods for Statistical Databases: A Comparative Study. ACM Computing Surveys, 21 (1989), 4, str. 515-556.

[2] Al-Attar A.: Data Mining - Beyond Algorithms. Attar Software, 1999. [http://www.attar.com/]

[3] Anderson R. A., Sweeney D. J., Williams T. A.: Quantitative Methods for Business. West Publishing Company, 2000.

[4] Bajec A. et al., ured.: Slovar slovenskega knjižnega jezika. SAZU in ZRC SAZU, Inštitut za slovenski jezik Frana Ramovša, 1994.

[5] Bock D. D., Ryan T.: Accuracy in Modeling with Extended Entity Relationship and Object Oriented Data Models. Journal of Database Management, 4 (1993), 4, str. 30-39.

[6] Burleson D. K.: Practical Application of Object-Oriented Techniques to Relational Databases. John Wiley & Sons, 1994.

[7] Cattell R. G. G., ed.: The Object Data Standard: ODMG 3.0. Morgan Kaufmann Publishers, 2000.

[8] Cattell R. G. G.: Object Data Management. Addison-Wesley Publishing Company, 1994.

[9] Chen P. P.: The Entity-Relationship Approach to Logical Data Base Design. Q.E.D. Information Sciences, Inc., Data Base Monograph Series No. 6, 1977.

[10] Chen P. P.: The Entity-Relationship Model-Toward a Unified View of Data. ACM-TODS 1 (March), 1976, str. 9-36.

[11] CODASYL: Data Base Task Group Report, 1971. Association for Computing Machinery, 1975.

[12] Codd E.F: A Relational Model of Data for Large Shared Data Bases. Communications of the ACM. 13 (June), 1970, str. 77-387.

[13] Delphi Group: Research Services. [www.delphigroup.com/research/], 1999.

[14] Dick Kevin: XML. A Manager's Guide. Addison Wesley, 2000.

[15] Elmasri R., Navathe S. B.: Fundamentals of Database Systems. Addison Wesley Publishing, 2000.

[16] Gartner Group: ITXPO'99 Symposium. October 11-15, 1999.

[17] Grad J., Dacar F.: Podatkovne strukture, osnove baze podatkov in njene uporabe - I. del. Ekonomska fakulteta Borisa Kidriča, 1993.

[18] Grad J., Jaklič J.: Baze podatkov. Ekonomska fakulteta v Ljubljani, 1996.

[19] Gradišar M., Resinovič G.: Informatika v poslovnem okolju. Ekonomska fakulteta, Ljubljana, 2001.

[20] Han J., Kamber M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, 2001.

[21] Hoffer J. A., George J. F., Valachic J. S.: Modern System Analysis and Design. The Benjamin/Cummings Publishing Company, 1996.

[22] Hoffer J. A.: An Empirical Investigation into Individual Differences in Data Models. Proceedings of Third International Information Sytem Conference. Ann Harbour, Mich., 1992.

[23] Hrvatin R.: Poslovni portal – vaša nova poslovna miza. Uporabna informatika, št. 2, letnik 8, 2000, str. 94-98.

[24] Huh S. Y.: Modelbase Construction with Object-Oriented Constructs. Decision Sciences, Vol. 24, No. 2, 409-434, 1993.

[25] Inmon W. H., Imhoff C., Sousa R.: Corporate Information Factory. Wiley Computer Publishing, 1998.

[26] Inmon W. H., Zachman J. A., Geiger J. G.: Data Stores, Data Warehousing and the Zachman Framework: Managing Enterprise Knowledge. McGraw-Hill, 1997.

[27] Inmon W. H.: Building the Data Warehouse. John Wiley & Sons, 1996.

[28] Jarke M., Lenzerini M., Vassiliou Y., Vassiliadis P.: Fundamentals of Data Warehouses, Springer, 2000.

[29] Kennedy R., Lee Y., Van Roy B., Reed C. D., Lippmann R. P.: Solving Data Mining Problems through Pattern Recognition. Prentice Hall, 1998.

[30] Kimball R.: The Data Warehouse Toolkit. John Wiley & Sons, 1996.

[31] Kovačič A.: Informatizacija poslovanja. Ekonomska fakulteta v Ljubljani, 1998.

[32] Kroenke D. M.: Database Processing: Fundamentals, Design, Implementation, 7th Edition. Prentice Hall, 2000.

[33] Marchand D. A., Kettinger W. J., Rollins J. D.: Information Orientation: The Link to Business Performance. Oxford University Press, 2001.

[34] Martin E. W., DeHayes D. W., Hoffer J. A., Perkins W. C.: Managing Information Technology: What Managers Need to Know. Macmillan Publishing Company, 1991.

[35] Martin J.: Principles of Object-Oriented Analysis and Design. Prentice-Hall, 1993.

[36] McFadden R. F., Hoffer A. J.: Data Base Management. The Benjamin/Cummings Publishing Company, 1985.

[37] Navathe S. B.: Evolution of Data Modeling for Databases. Communications of the ACM, 35 (1992), 9, str. 112-123.

[38] Pigford D. V., Baur G.: Expert Systems for Business: Concepts and Applications. Boyd & Frases Publishing Company, 1992.

[39] Prijatelj N.: Matematične strukture I. Partizanska knjiga, 1980.

[40] Rishe N.: Database Design. The Semantic Modeling Approach. McGraw-Hill, 1992.

[41] Sauter V. L.: Decision Support Systems: An Applied Managerial Approach. John Wiley & Sons, 1997.

[42] Schmidt Bob: Data Modeling for Information Professionals. Prentice Hall, 1999.

[43] Sobočan B.: Upravljanje z znanjem. Uporabna informatika, št. 3, letnik 6, 1998, str. 14-19.

[44] Stein R. M.: Object Databases. Byte, 19 (1994), 4, str. 74-84.

[45] Tsichritzis D. C., Lochovsky F. H.: Data Models. Prentice-Hall, 1982.

[46] Turban E., Aronson J. E.: Decision Support Systems and Intelligent Systems, 6th Edition, Prentice Hall, 2001.

[47] Turban E., McLean E., Wetherbe J.: Information Technology for Management: Making Connections for Strategic Advantage, 2nd Edition. John Wiley & Sons, 1999.

[48] Watson Richard T.: Data Management. Databases and Organizactions. John Wiley & Sons, 1999.

[49] Winston W. L.: Operatons Research: Applications and Algorithms. Duxbury Press, 1991.

Stvarno kazalo

- 142
% 135
* 143, 142
/ 142
_ 135
+ 142
1NF *glej normalna forma*
2NF *glej normalna forma*
3NF *glej normalna forma*

A

agregacija 14
    stopnja 14
agregatne funkcije 141
aktivnosti upravljanja podatkov 30-39
algoritem 37, 156
ALTER TABLE 130
analitična obdelava 16
analitični hierarhični postopek 155
analitik 40, 56-57, 84, *glej tudi uporabnik*
analiza 40-41
    kaj-če *glej kaj-če analiza*
    občutljivosti 151, 153
    podatkov 150-159, 177
    povezav 49
    statistična 160
AND 136
anomalije 111-112
    pri brisanju 112
    pri spreminjanju 111
    pri vnosu 111
arhiviranje 38
AS 141
ASC 143
atribut 42, 93, 102
    elementarni 44
    izpeljani 44
    ključni *glej ključ*
    neključni 95-96, 99, 122-123
    sestavljeni 44, 94
    večvrednostni 43, 94, 99, 103, 115-118, 121-122
    vrednost 43
AVG 141
avtomatizacija 79, 149, 164, 190, 196, 197

B

baza znanja 63, 163
BETWEEN 137

C

CASE *glej orodje CASE*
celovitost 23, 33
    referenčna 34
celovitost podatkov 79-81, 128
    eksplicitna 81
    omejitev območja 80
    referenčna 81
CODASYL 93
CONSTRAINT 130
COUNT 141
CREATE TABLE 127-130
CRM *glej uravnavanje odnosov s kupci*

Č

časovna vrsta 168-169

D

DDL *glej jezik za definiranje podatkov*
dedovanje 104
deljenje podatkov 4, 5, 29, 30
dejstvo 163
DELETE 131-132
DESC 143
dimenzija 106
  časovna 18, 106
dimenzijska tabela 107
DISTINCT 140
DML *glej jezik za manipulacijo podatkov*
dodana vrednost 22
dodatna vrednost 188
dokument 64-66
  življenjski cikel 64
dokumentacija
  podatkovne baze 124
  pregled 56
  zbiranje 40
dokumentna baza 64
domena 44
doseganje cilja 159-165
dostopnost 32-33, 84
drevo odločitev *glej odločitveno drevo*
DROP TABLE 130
drseča povprečja 155, 169

E

EDI 15, 23-24, 145
ekspertni sistem 63, 162, 169
eksponentno glajenje 169
ekstranet 15
elektronsko poslovanje 23, 29, 200
entiteta 42, 93, 98
entitetni tip 45, 93, 98
ER diagram 51, 124
ER model *glej model entitet-povezav*

F

fizična predstavitev 101
FOR 144
FROM 133-143
funkcionalna odvisnost 53-55
  analiza 120-123

G

generalizacija 104
generator naključnih števil 174
genetski algoritmi 194
GIGO 157
gotovost 157
graf 183
GROUP BY 142
grupiranje 141, 142

H

hierarhija tipov 104

I

identifikator 13, 16, 19
IF 144
indeks 82-84
induktivno pravilo 191
informacijska asimetrija 2, 148-149
informacijska demokratizacija 178
informacijska podpora managementu 5, 148-200
informacijska usmerjenost 2
informacijsko obnašanje 2-3
informatizacija poslovnih procesov 148
INSERT 130-131
integracija 21
inteligentni agenti 62
intervju 40, 56-57
  posamični 56-57
  skupinski 56-57
intranet 63
izjeme 183
izobraževanje 39, 59, 85

J

jezik
  nepostopkovni 78
  postopkovni 144
  SQL 78, 101, 124, 125-144, 179, 183
  za definiranje podatkov 77, 125
  za manipulacijo podatkov 78, 125
  XML 15, 24, 145

K

kaj-če analiza 149, 159, 165
kardinalnost 48, 50-53
kartezični produkt 138
ključ 46-48
  glavni 47, 94-97, 107, 117, 122, 130
  kandidat za 46, 96
  nadomestni 96
  sestavljeni 47, 96, 99, 113, 117, 122
  tuji 99, 104, 113, 122
kocka 106-107, 167, 181
kriterijska funkcija 170-172
krivulja dviga 187

L

lakota po podatkih 179
LIKE 134
linearni program 170
linearno programiranje 170

M

managerski informacijski sistem 148
MAX 141
mehanizem sklepanja 163
mehki kazalci 104
mera 106
metapodatki 32, 76, 126-127
metoda 102
MIN 141
model 86, 151, 153-156, 165
  deterministično 157
  entitet-povezav 41-53
  hierarhični 87,. 88-92
  matematični 154
  mrežni 87, 92-93
  objektni 87, 102-105
  podatkovni 56-59, 86
  poslovni 154-156
  problemskega področja 41
  procesov 59, 63
  relacijski 87, 93-102, 110
  večdimenzionalni 17, 87, 105-109, 180
  vrst 154
modeliranje 150, 154, 160, 161, 165
  podatkovno *glej podatkovno modeliranje*
MOLAP 181
Monte Carlo 172-177

N

načrtovalec 41, 49, 74, 84
načrtovanje 30-31, 39-59
nadgradnja 37-38
nadtip 104
najbližji sosed 189
napovedovanje 168-169
negotovost 157, 173
neodvisnost
  med modelom in algoritmom 156
  med modelom in podatki 156
  med programi in podatki 76
  od programov 71
neposredno trženje 186
nesreča 37
nevronska mreža 192-194
nivo
  analitični 16
  operativni 12
normalizacija 120-123
normalna forma 121-122
  prva 122
  druga 122
  tretja 122
NOT 130, 136
NULL 43, 46, 111, 112, 130, 131

obnavljanje 37-38, 82
obračun uporabe 37
odločanje 150
  večkritertijsko 155
odločitvena tabela 155
odločitvene spremenljivke 156, 170
odločitveno drevo 155, 189-190
odvisnost
  delna 122
  tranzitivna 122
ograjevanje 104
okolje pajkove mreže 8
OLAP *glej sprotna analitična obdelava podatkov*
omejitev 79-81, 130, 170
opcija 173-177
  cena 173
operativna shramba podatkov 21
operativni nivo 12-13
opis 22-24, 32
opredelitev 23, 32
optimizacija 154, 155, 159-160, 169-172
  linearna *glej linearni program*
  lokalna 151
  nelinearna 171-172
OR 136
ORDER BY 143
orodje
  CASE 123-124
  OLAP 160, 167, 182-184
  za skupinsko delo 158, 200
oznaka 23

P

pajkova mreža *glej okolje pajkove mreže*
Paretovo pravilo *glej pravilo 80/20*
personalizacija 200
podatki
  agregatni 183
  ažurnost 85
  brisanje 131-132
  celovitost *glej celovitost podatkov*
  detajlni 12
  izpeljani (izračunani) 114-115, 183
  kakovost 22, 33-34
  modeliranje *glej podatkovno modeliranje*
  nadzorovani 156, 169
  nekonsistentni 6
  nenadzorovani 156, 157
  opisovanje 22-24
  organizacija 29, 32-33
  podvajanje *glej podvajanje podatkov*
  poizvedovanje 133-144
  popolnost 85
  priprava 156
  proaktivna uporaba 3
  referenčni 14
  spreminjanje 132
  sumarni 19, 38, 166
  surovi detajlni 12
  točnost 85
  tok 21-22
  vhodni 156
  vnos 130-131
  zaklepanje 5
  zaznavanje 28
  zbiranje 28-29
  zgodovinski 13-14, 18, 38
  zgodovinski referenčni 14
  značilnosti uporabe 16-17
  zunanji 15-16
podatkovna arhitektura 10
podatkovna baza 70, 70-85
  definiranje 127-130
  integrirana 6, 70
  načrtovalec 74
  načrtovanje 74, 85-124
  operativna 198
  relacijska 100, 106, 124
  skrbnik *glej skrbnik podatkovne baze*
  tvorba 74, 75
  uporaba 71-74
  vzdrževanje 75, 79-84

podatkovna preobremenjenost 29
podatkovni slovar 32, 76
podatkovni vir 4
  integrirani 9, 18-21
  problemi upravljanja 4-9
  zunanji 15-16, 198
podatkovni zapis 87, 94
  podvajanje 90
podatkovno modeliranje 56-59
podatkovno skladišče 18-20, 105, 180, 198
področno podatkovno skladišče 20, 105, 180, 198
podtip 104
podvajanje podatkov 70, 90-92, 98, 110, 111
  odprava 118-120, 120-123
pogled na podatke 123, 179-184
pogojno izvajanje 144
poimenovanje 22
poizvedba 133
poizvedovanje 101, 133-144
polje 94
ponavljanje 144
portal
  poslovni 64, 198-200
  spletni 200
  za elektronsko poslovanje 200
  za kupce 200
  za zaposlene 200
poslovna uporabnost 14
poslovni portal *glej portal*
poslovno obveščanje 177
poslovno področje 18
poslovno pravilo 40, 48, 74, 124
posplošena drseča povprečja 169
poškodovanje 37
povezava 48, 160-161, 184, 194
  med entitetnimi tipi 49-51
  med relacijami (tabelami) 97-100, 117
povezovalni pogoj 138-140
pravica uporabe 23, 32
pravilo 80/20 59
pravilo sklepanja 163
pravilo
  poslovno *glej poslovno pravilo*
  induktivno *glej induktivno pravilo*
preglednica 149, 164-177
prelomna točka 166
pridobivanje 31
prilagajanje 200
PRIMARY KEY 130
primerjava 183
problem 151, 156
  analiza 153
  kvalitativna analiza 153
  kvantitativna analiza 153
  strukturiranost 149, 152-153
problemsko področje 40-42, 48
proces odločanja 150-156
profil kupca 186

R

računalniška izmenjava podatkov 15
razred 103
razvrščanje v skupine 189
relacija 93
reševanje problemov 151
rezanje 183
rezervna kopija 37
RIP *glej računalniška izmenjava podatkov*
ROLAP 181
rudarjenje v podatkih 62, 160, 178, 184-197
  proces 194-195

S

segmentacija 189
SELECT 133-144
semantika 111
SEQUEL 125
SET 132
shema 87
  kakovost 110-113
  relacijska 97, 105, 110-113, 124
  večdimenzionalna 106-109
  zvezdasta 108
sistem za podporo odločanju 149, 159-177
sistem za upravljanje podatkovnih baz 63, 75-84
  hierarhičnih 91
  mrežnih 93
  objektnih 103
  relacijskih 101-102
  uporaba 77-79
  večdimenzionalnih 106
sistem za upravljanje delovnih tokov 65, 148
sistem za upravljanje dokumentov 63, 64-66
sklepanje 163
skrbnik podatkovne baze 4, 72, 75, 81, 84
sloj
  transformacijski 22
  integracijski 22
specializacija 104
sprotna analitična obdelava podatkov 17, 148, 160, 167-168, 178-184
sprotna transakcijska obdelava 180
SQL *glej jezik SQL*
statistična podatkovna baza 36
stolpec 93
strojno učenje 194
struktura podatkov 17, 23, 75-76, 88-109
  večdimenzionalna 20
SUM 141
SUPB *glej sistem za upravljanje podatkovnih baz*
svetovanje 39

Š

šablona 114-115
številčnost *glej kardinalnost*

T

tabela 93
  dejstev *glej tabela dejstev*
  dimenzijska *glej dimenzijska tabela*
tabela dejstev 107
tip povezave 48
transformacija 19
tveganje 157

U

učinkovita uporaba podatkov 2-4
UPDATE 132
uporabnik 9
  analitik 10, 105, 167, 169, 180, 182, 185, 195-196
  odločevalec 105, 147, 157, 178
  operativni 9
  podatkovne baze 85
uravnavanje odnosov s kupci 185
urejanje 64, 143-144

V

varovanje 34
  politika 34
vir 31
vloga 84-85
vodenje projektov 155, 160, 162
vprašalniki 56
vrstica 93
vrtanje v globino 183, 198
vrtenje 184
vrtilna tabela 166-168
vzdrževanje 31
vzvratno inženirstvo 124

W

WHERE 133-144

X

XML, *glej jezik XML*

Z

zahteva 104

zakaj 160, 161, 183

zaloge 148, 152, 155, 156

zaščita 34-37
- pred nepooblaščenim dostopom 81-82
- pred posledicami poškodb 82

zaupanje 190

znanje 59
- odkrivanje 184
- omogočanje dostopa 60
- ponovno odkrivanje 196
- pridobivanje 62
- tvorba 60
- uporaba 60, 62
- upravljanje 59-66, 60
- viri 63

zrcaljenje 82

zvijanje 183

Ž

življenjski cikel upravljanja podatkov 28-30

MARIBOR UNIVERZITETNA KNJIŽNICA LJUBLJANA

NARODNA IN UNIVERZITETNA
KNJIŽNICA
COBISS ®
00000253994

NARODNA IN UNIVERZITETNA KNJI:NICA
GS 544 734
COBISS I