

VREMENSKO-PROSTORNI PRIKAZ SLOŽENIH PODATAKA NA WEB-u

Izv. prof. dr. sc. Jadranka Pečar Ilić

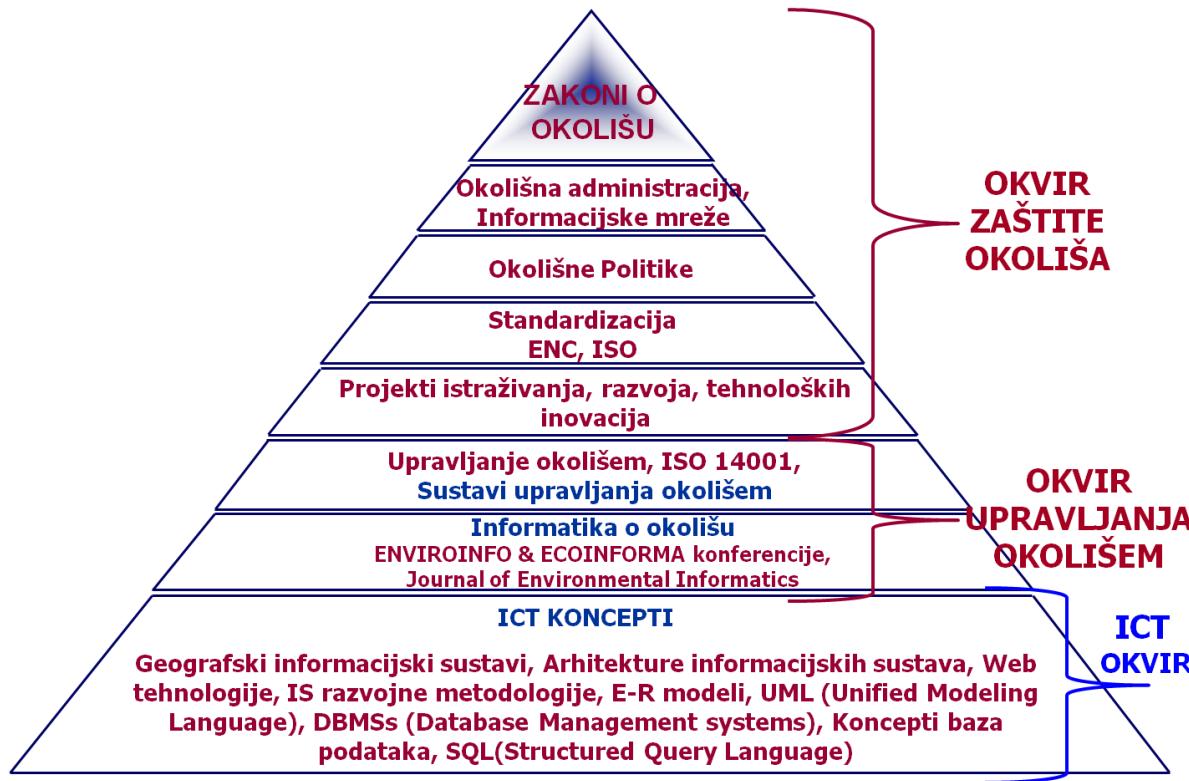
*viša znanstvena suradnica u
Laboratoriju za informatiku i modeliranje okoliša*



*Institut Ruđer Bošković
Zavod za istraživanje mora i okoliša*

Bijenička 54, HR-10000 Zagreb, Hrvatska
e-mail: pecar@irb.hr

ULOGA ICT U UPRAVLJANJU OKOLIŠEM



Pristup „od vrha prema dnu“ (top-down) [Pečar-Ilić, Ružić, 2005, 2011]

Razine hijerarhijske piramide:

- Vrh: zakoni i propisi o okolišu (politike), važeće norme (standardi)
- Sredina: *Informatika o okolišu* - interdisciplinarno područje omogućuje provođenje znanstveno-istraživačkih projekata u okviru znanosti o okolišu te upravljanja okolišem
- Baza: *ICT koncepti/znanja* (suvremeni pristupi, postupci, metode, tehnike)

POLITIKE EUROPJSKE UNIJE

Economy and Society

- Agriculture
- Audiovisual
- Biotechnology
- Civil Society
- Competition
- Consumers
- Culture
- Customs Union
- Economic and monetary union
- Education and Training
- Employment and Social affairs

Energy

Enterprise

Environment

- Fisheries
- Food Safety
- Freedom, security and justice

Information Society

Internal Market

Public Health

Regional policy

Research, Development, Technology and Innovation

- Space
- Sport
- Taxation
- Trans-European networks
- Transport
- Youth

International Affairs

Institutional Affairs

Finance

PRIMJER

The “Joint Statement on Guiding Principles on the Development of Inland Navigation and Environmental Protection in the Danube River Basin (2007/2008)” dogovorena je između

- Međunarodne komisije za zaštitu rijeke Dunav (ICPDR),
- Dunavske komisije za plovidbu (DC),
- Međunarodne komisije za sliv rijeke Save (ISRBC).

Integracija politike zaštite okoliša (Environmental Policy Integration)

Sve politike EU moraju uključivati i komponentu kojom se vodi računa o zaštiti okoliša!

AMERIČKA I EUROPSKA AGENCIJA ZAŠTITE OKOLIŠA (EPA i EEA)



Environmental Protection Agency (EPA) predvodi tzv. upravitelj, koji je imenovan od Američkog Predsjednika.

Zbog toga se može reći da EPA ima sličnu ulogu kao i Opća uprava za okoliš unutar Europske komisije (EC's: DG Environment) !



European Environment Agency (EEA) i European Environment Information and Observation Network (EIONET)
Ustavljene 22. srpnja 2003. uredbom Europskog parlamenta (EC) No 1641/2003 i amandmanom Vijeća EU 1210/90/EEC.

ECOinformatics initiative pokrenuta je između EPA i EEA 2003. godine u svrhu razmjene informacija o okolišu na međunarodnoj razini.

PROCES EU INTEGRACIJE RH ZAŠTITA OKOLIŠA

Poglavlje Okoliš

- jedno od najopsežnijih poglavlja ~ 300 direktiva i uredbi
- jedno od komplikiranijih, zbog težine i velikih finansijskih ulaganja
- hrvatski pregovarači pripremili su screening listu (98 str.) te odgovore na upitnik Europske komisije (272 str.)
- pregovori su ovisili o poglavljima vezanim uz okoliš - gospodarski sektori (npr. [Nikola Ružinski](#), pregovorač za poglavlja Energetika i Okoliš te [Nenad Mikulić](#), voditelj radne skupine za Okoliš)

ZAKON O ZAŠTITI OKOLIŠA U RH (NN br. 80/2013; NN br. 78/2015)

... zamišljen kao krovni zakon, koji povezuje sve zakone koji se bave zaštitom okoliša i obuhvaća sve direktive u europskoj pravnoj stečevini vezane uz to poglavlje (direktiva o vodama, zraku ...)

EU programi za pomoć RH

[\[http://euinfo.pravo.hr/\]](http://euinfo.pravo.hr/), a ranije www.entereurope.hr

- **CARDS** (Community Assistance For Reconstruction, Development and Stabilisation)
- **IPA** (Instrument for Pre-Accession Assistance)

**30.06.2011. OKONČANI PREGOVORI
RH O PRISTUPANJU EU
punopravna članica 1.07.2013.**

INFORMACIJSKI SUSTAV ZAŠTITE OKOLIŠA U RH

Ministarstva/Vlada/Sabor RH



Studije, projekti, investicije

**Twinning projekt za unapređenje
izvješćivanja o okolišu RH
(sredstva iz EU IPA programa za 2007.)**

[izvor: AGENCIJA ZA ZAŠTITU OKOLIŠA]

INFORMACIJE O OKOLIŠU

- u pisnom, vizualnom, slušnom, elektroničkom ili drugom materijalnom obliku
- o stanju okoliša i čimbenici koji utječu na sastavnice okoliša (poplava, nestanak staništa, istrebljenje vrsta, staklenički plinovi, radioaktivni otpad, buka, itd.)
- o zdravlju ljudi, uvjetima života i njihovoj sigurnosti, o kulturnim stećevinama, ...
- o provođenju okolišnih politika, legislativa, sporazuma i implementacija
- izvješća o okolišu i opterećenjima, analize troškova i koristi kod donošenja odluka

[2003/4/EC; HR Zakon o zaštiti okoliša NN 80/13]



Elektrana u Ohio



Peking, dan nakon kiše i sunčan dan ali s puno onečišćenja



Budimpešta, poplava Dunava 2006



Bengalski tigar

ULOGA ICT-a U ŠIRENJU INFORMACIJA O OKOLIŠU

“Informacijske i komunikacijske tehnologije (ICT) i odgovarajuće aplikacije u području znanosti o okolišu i upravljanja okolišem predstavljaju osnovu za učinkovitu razmjenu (širenje) informacija o okolišu.”

[izvor: Werner Pillmann from International Society for Environmental Protection-ISEP, 2003]

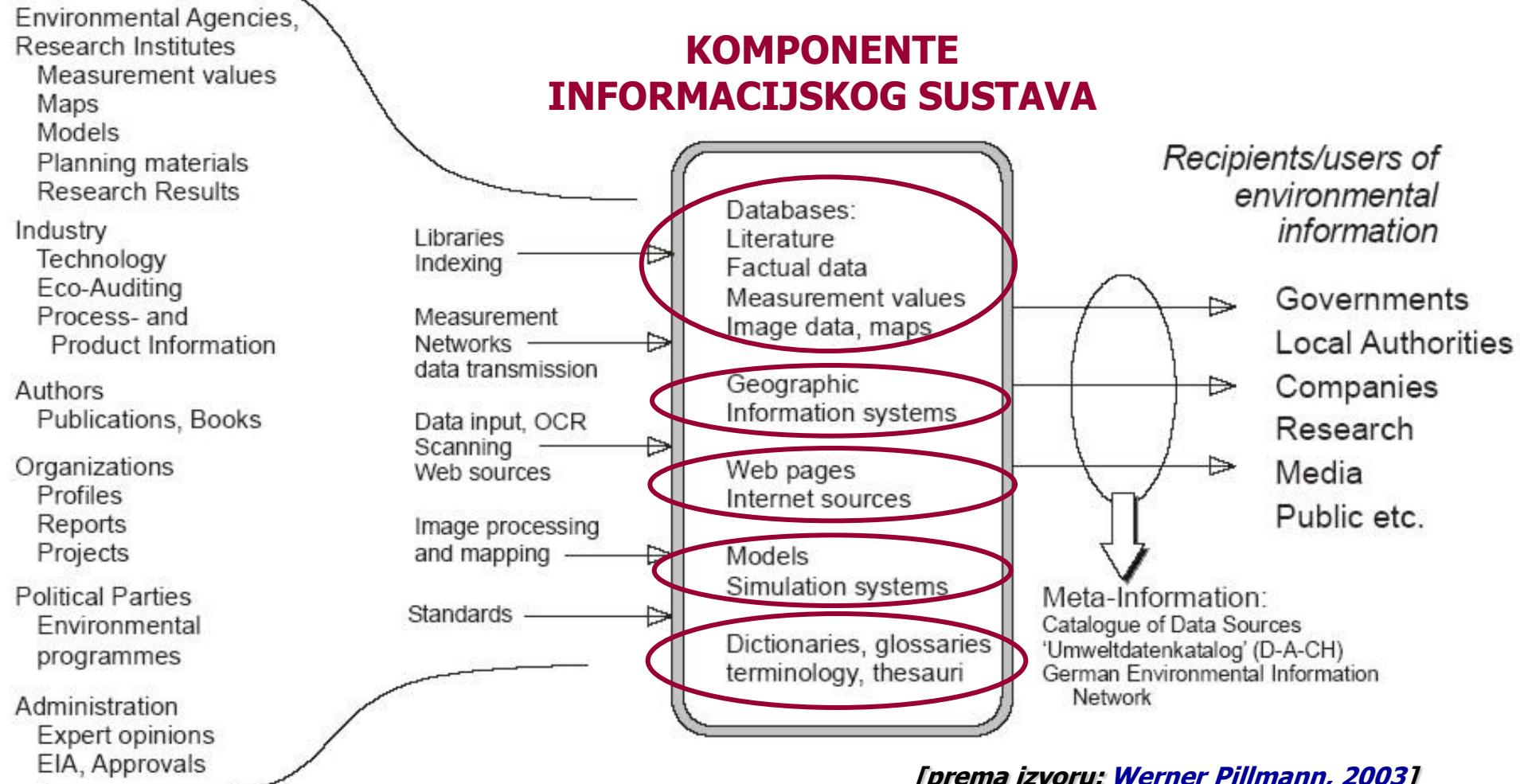
KARAKTERISTIKE INFORMACIJA O OKOLIŠU

| | |
|-------------------------|--|
| VRSTA područja primjene | energija, promet, poljoprivreda, klimatske promjene, otpad i materijalni tokovi, legislativa, oporezivanje i dr. |
| PROSTORNA dimenzija | koordinate točkastih izvora onečišćenja, mjerne postaje, gradovi, države, regije, riječni slivovi i dr. |
| VREMENSKA dimenzija | vrijeme/datum, godina, vremenski niz, interval promatranja |
| OBLIK prikazivanja | osobna komunikacija na konferencijama, pisani materijali, digitalne informacije: podaci, baze, slike, multimedijijski podatci, web portali i dr. |
| POŠILJATELJ/ PRIMATELJ | znanstvenici, agencije za zaštitu okoliša i njezini korisnici, javne uprave, nevladine organizacije, tvrtke, građani i dr. |

TOKOVI INFORMACIJA O OKOLIŠU

Tokovi informacija o okolišu između primatelja i pošiljatelja informacija uspostavljaju se kroz složene informacijske sustave koji se razvijaju u interdisciplinarnom području Informatike o okolišu

KOMPONENTE INFORMACIJSKOG SUSTAVA



[prema izvoru: Werner Pillmann, 2003]

INTERDISCIPLINARAN KARATKER INFORMATIKE O OKOLIŠU

brzo rastuće *interdisciplinarno područje* nastalo rasplinjavanjem granica između **znanosti o okolišu**, **računarstva** i **informacijskih znanosti** kao i brojnih područja u društvenim znanostima



[izvor: [UK Natural Environment Research Council](#)]

Interakcija između polja i tematika

metode praćenja kakvoće okoliša, informacijski sustavi o okolišu, ekološko modeliranje, metapodaci i izvješćivanje o okolišu, normizacija, okolišna statistika, geografski informacijski sustavi (GIS), sustavi temeljeni na znanju, geoprostorne tehnologije, daljinska detekcija i mjerena, web tehnologije, aplikacije i usluge, širenje informacija te podrška politikama i sudjelovanju javnosti u pitanjima okoliša.

ŠTO JE INFORMATIKA O OKOLIŠU?

ENVIRONMENTAL INFORMATICS (kraće ENVIROMATICS)

“**Environmental Informatics** je područje u kojem se telekomunikacijske tehnologije i obrada informacija o okolišu spajaju sa znanstvenim istraživanjima, poslovnim upravljanjem i zahtjevima administracije (uprave).”

“**ENVIROMATICS -Enviro[mental Telematics and Infor]matics** je područje znanstvenih i tehničkih aktivnosti koje objedinjuje sva „*state-of-the-art*“ znanja potrebna za dohvaćanje i upravljanje **prostorno-vremenskim podacima** o prirodi i okolišu podložnom ljudskom djelovanju, kako bi se omogućilo stvaranje, upravljanje i širenje relevantnih **informacija o okolišu**.“

[izvor: *Werner Pillmann, 2003*]

“**Enviromatics** (Telematika i informatika o okolišu) koristi najnovija dostignuća (znanja) u području **telekomunikacija** i **informatike** te **računarstva** za razvoj i/ili primjenu znanstvenih metoda te učinkovitih pristupa i tehnika za problematike istraživanja okoliša i upravljanje okolišem, kojima će se omogućiti integracija i upravljanje podatcima, informacijama i znanjem za donošenje pravovremenih odluka.

MEĐUNARODNE KONFERENCIJE I PROJEKTI

Područje INFORMATIKE O OKOLIŠU - razvoj od 80-tih godina!

German Society
for Informatics
(GI) godišnje od
1986.

6 konferencija
Zajedno
njemački i američki
znanstvenici 1989.-2001.



Od 2001. svake godine:
EnviroInfo

30th International Conference on
Informatics for Environmental Protection
Berlin, 2016

HTW Berlin-University of Applied Sciences

- the latest *state-of-the-art* development on ICT and environmental related fields
- **Hamburg, 2013** Special Theme: ICT & Renewable Energies (Wind Energy)

EU FP7 “ICT-ENSURE” Project

EU network of experts in ICT for environmental sustainability

▪ **INFORMATION SYSTEM ON LITERATURE**

[<http://iai-uiserv1.iai.fzk.de/ictensure/site?mod=litdb>]

- ICT in Energy Consumption and Energy Efficiency
- ICT and Climate Change
- ICT and Sustainable Use of Natural Resources
- ICT for Biodiversity



<http://horizon2020projects.com/>

ČASOPISI POSVEĆENI INFORMATICI O OKOLIŠU

1. International Environmental Modelling & Software Society (iEMSs)

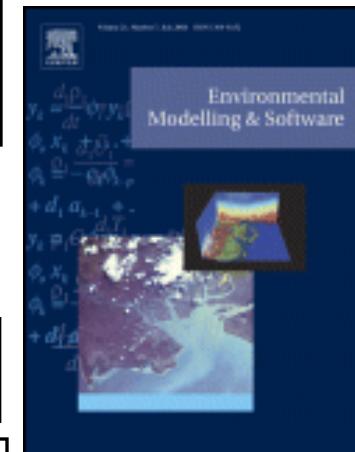
➤ **Environmental Modelling and Software journal (Elsevier)** – research articles, reviews, short communications, software and data news, on recent advances in environmental modelling and software.

| Category Name | Total Journals in Category | Journal Rank in Category | Quartile in Category |
|--|----------------------------|--------------------------|----------------------|
| COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS | 100 | 8 | Q1 |
| ENGINEERING, ENVIRONMENTAL | 42 | 6 | Q1 |
| ENVIRONMENTAL SCIENCES | 210 | 27 | Q1 |

Abstracting/ indexing:

- Current Contents/Agriculture, Biology & Environmental Sciences
- Science Citation Index, SCOPUS, INSPEC

Impact factor: 4.420
Journal Citation Reports, ® 2014



➤ iEMSs Inter. Conferences (since 2002, every two years)

2. International Society for Environmental Information Sciences (ISEIS)

➤ **Journal of Environmental Informatics (JEI)** – on environmental systems science and information technology.

Abstracting/ indexing:

- Web of Science, Scopus, ProQuest, EBSCO Environmental Engineering Abstracts

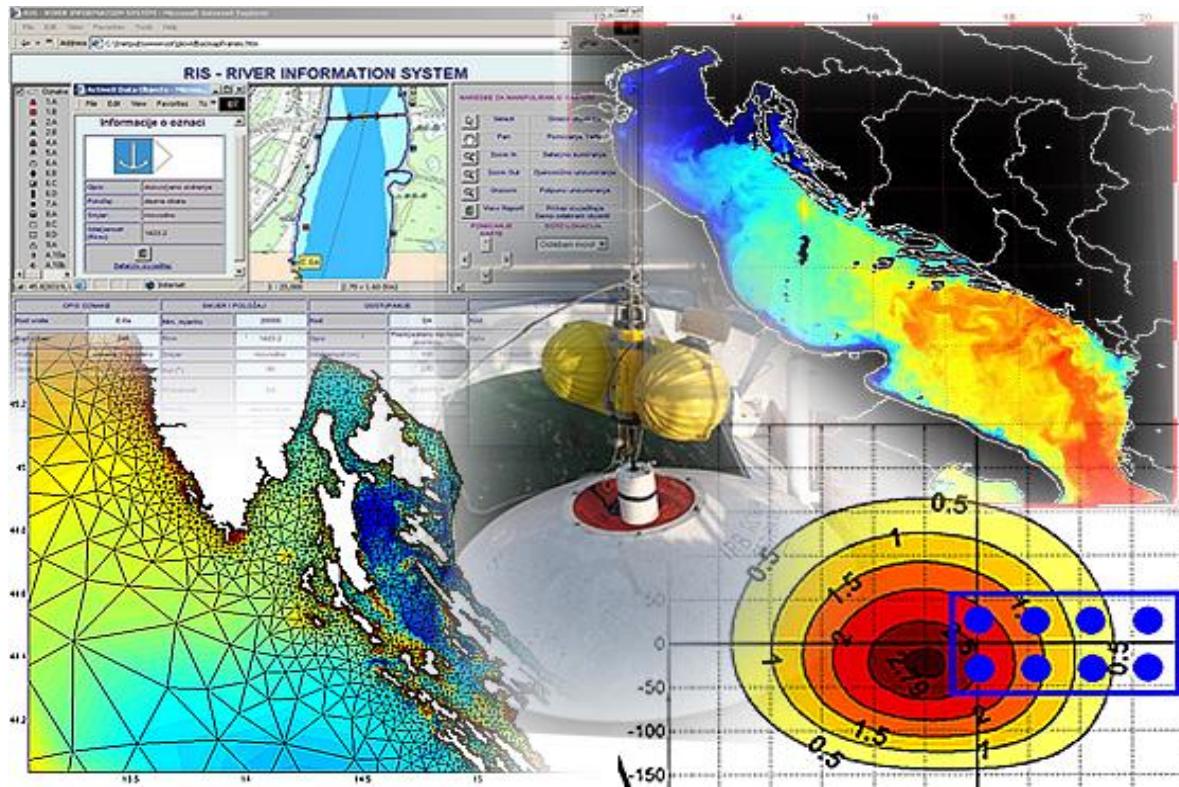
Impact factor: 3.619
Journal Citation Reports, ® 2012



➤ ISEIS Inter. Conferences on Environmental Informatics (since 2003 -)

ISTRAŽIVANJE I ZAŠTITA OKOLIŠA – ZIMO, IRB

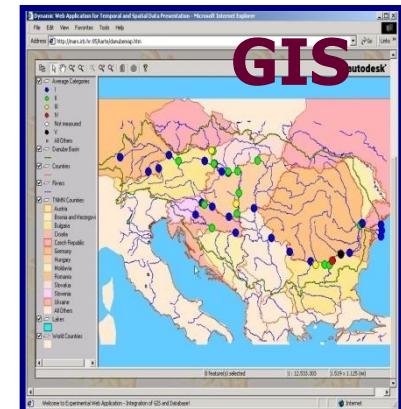
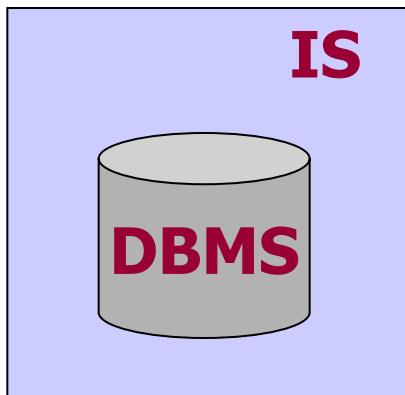
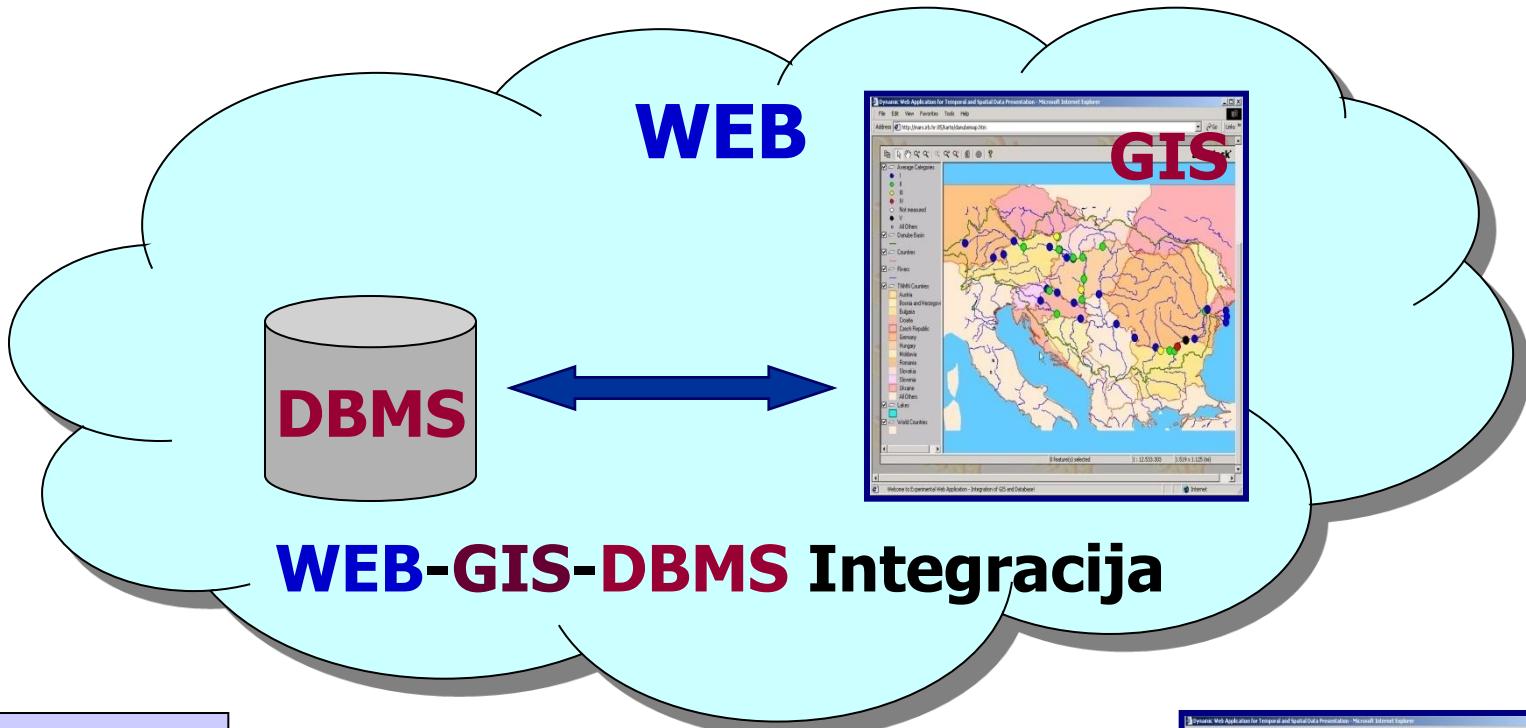
- informatika o okolišu
- satelitska oceanografija



- ekološko modeliranje

[izvor: [ZIMO](#)]

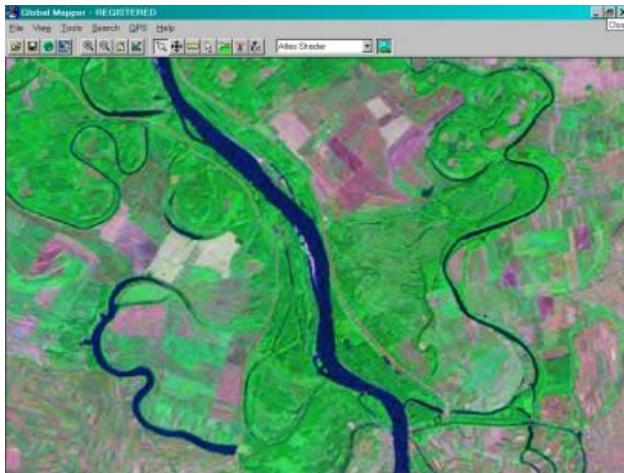
WEB GIS TEHNOLOGIJE ZA PROSTORNO-VREMENSKI PRIKAZ (Informacija o okolišu)



GEOPROSTORNE INFORMACIJE

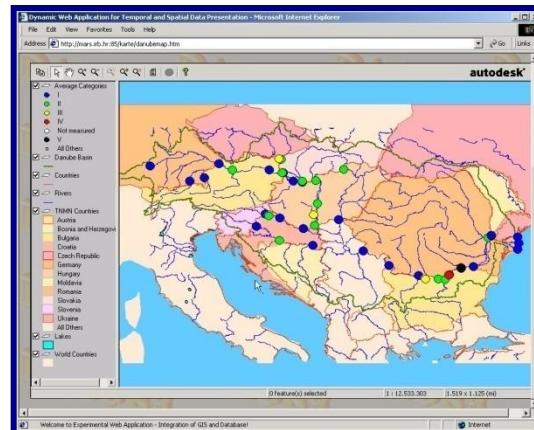
Geoinformacije (georeferencirane ili geoprostorne informacije) su informacije o pojavama koje su implicitno ili eksplisitno vezane uz lokaciju na Zemljinoj površini (te iznad ili ispod nje).

[izvor: *Međunarodna organizacija za standarde (ISO), norma 19101*]



Satelitske slike

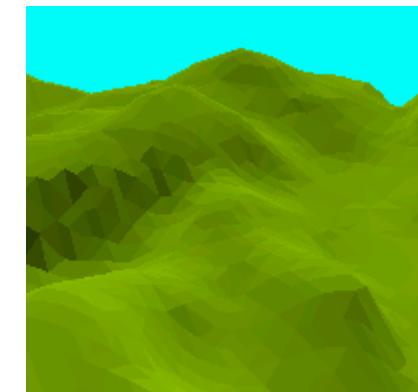
[npr. **IKONOS 2 satelit**
prikuplja i obrađuje
900 - 1000 slika
rezolucije od 1m na **dan**]



Digitalne karte



Aerosnimke



3D digitalni
model reljefa

WEB GIS

- Web GIS se odnosi na Geografske informacijske sustave koji koriste web tehnologije kao osnovu za komunikaciju između pojedinih elemenata (podsistava) GIS-a.

Web GIS omogućuje:

- upravljanje **geoprostornim podatcima** u internet (ili intranet) okruženju
- upravljanje **interaktivnim korisničkim upitima**
- prikaz geoprostornih podataka krajnjim korisnicima koristeći **standardne web preglednike**
- **integraciju GIS-a s drugim tehnologijama** (npr. *Relational Database Management System - RDBMS* sustavima, geoprostornim tehnologijama i servisima) kako bi se osigurala arhitektura za moćni prostorni sustav za podršku odlučivanju (*Spatial Decision Support System - sDSS*).

PRIMJER PROIZVOĐAČA WEB GIS SOFTVERA

- Više od 30 različitih proizvoda za Web GIS kartiranje koje nude pojedini GIS proizvođači.
- Tri najpoznatija **komercijalna** Web GIS softverska paketa od vodećih GIS proizvođača su:
 - ArcIMS – ESRI [<http://www.esri.com>],
 - MapGuide – AutoDesk [<http://www.autodesk.com>],
 - GeoMedia WebMap – Intergraph [<http://www.intergraph.com>].

OSGeo.org - Open Source Geospatial Foundation Website
[<http://www.osgeo.org/>]

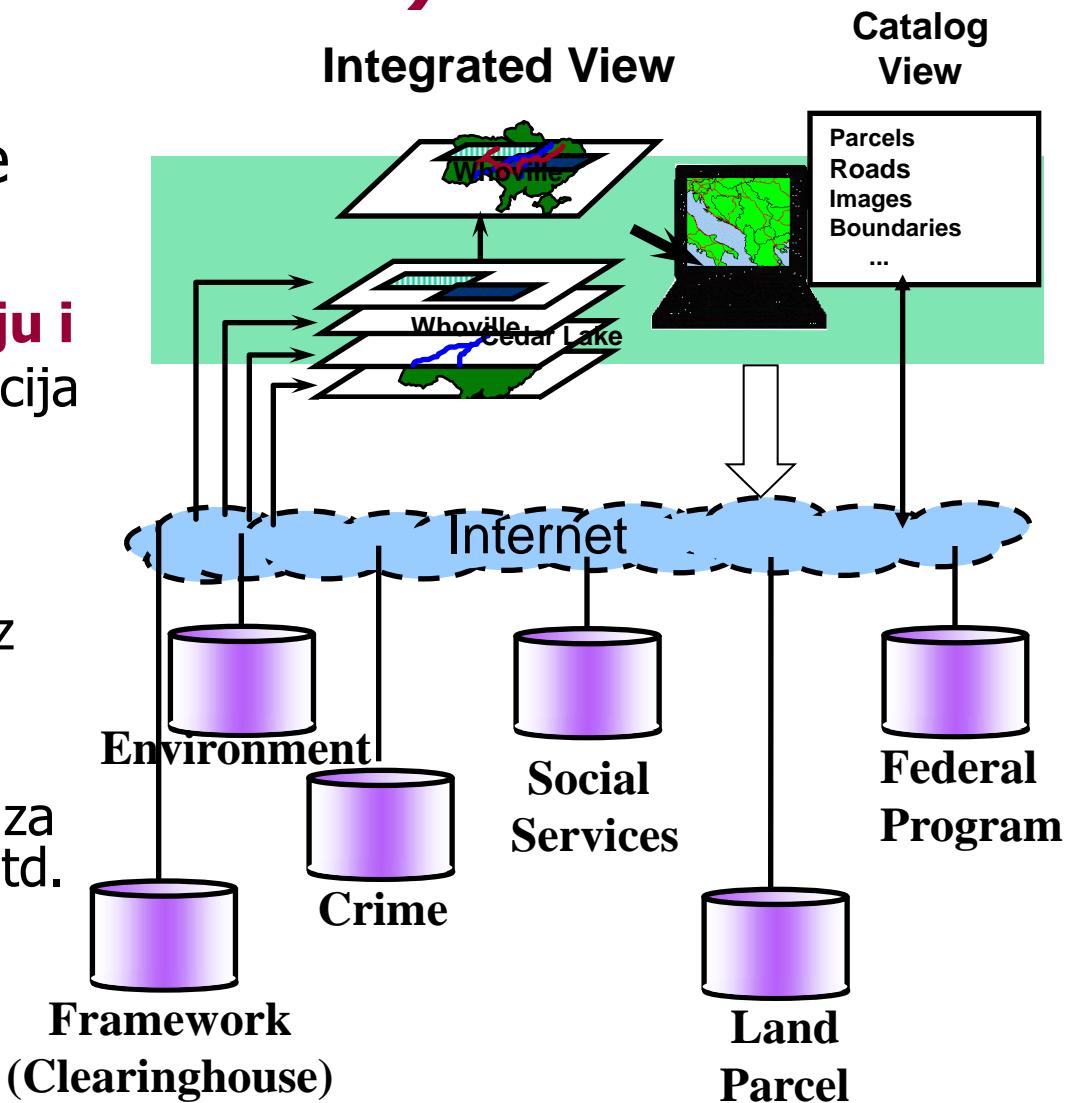
Primjer funkcionalnosti za: MapGuide Open Source
([MapGuide 2.6 Release Notes for Windows 7, 8](#)
<http://mapguide.osgeo.org/>)

- **is a web-based platform that enables users to quickly develop and deploy web mapping applications and geospatial web services.**
- includes an interactive viewer and XML database for storing and managing content, and supports most popular geospatial file formats, databases, and standards. The MapGuide platform can be deployed on Linux or Microsoft Windows, supports Apache and IIS web servers, and offers extensive PHP, .NET, Java, and JavaScript APIs for application development.

DEFINICIJA KARTOGRAFIJE ZA WEB (WEB MAPPING)

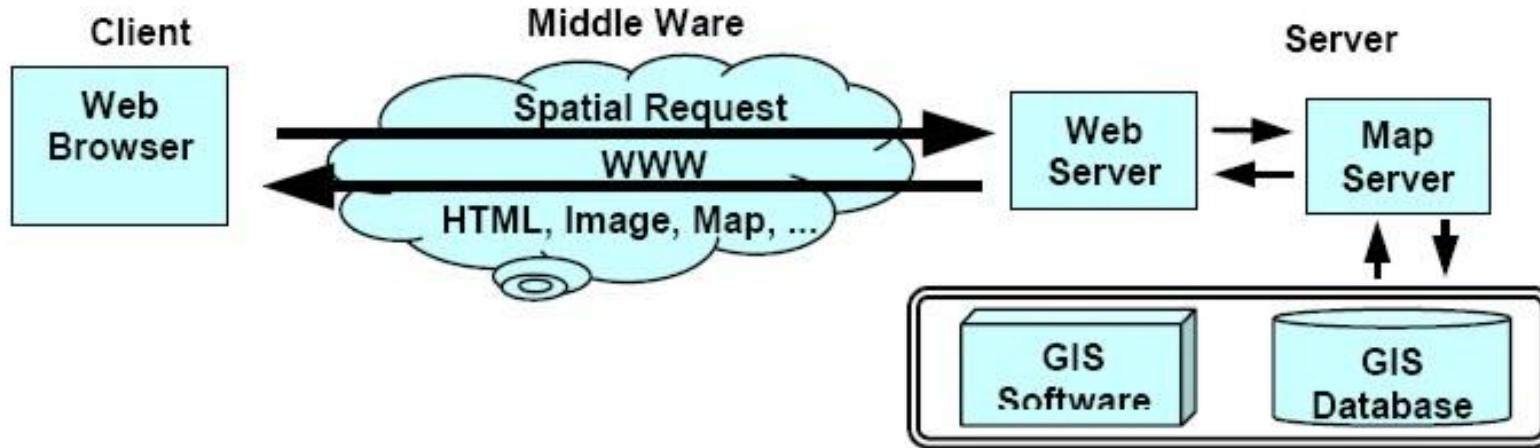
Web poslužitelji i klijenti za prikaz karata komuniciraju te omogućuju:

- **Otkrivanje, pristup, integraciju i korištenje** geoprostornih informacija sa različitih poslužitelja baza podataka
- **Integrirani prikaz** višestrukih slojeva geoprostornih podataka kroz zajedničku **web aplikaciju**, te postavljanje upita i ažuriranje
- **Prostorne analize i aplikacije** za vizualizaciju, podršku odlučivanju itd.
- **Podršku za geokodirane rasterske, vektorske** i druge podatke



[izvor: OGC-OPEN GEOSPATIAL CONSORTIUM- <http://www.opengeospatial.org/>
standardi za interoperabilnost geoprostornih tehnologija]

WEB GIS ARHITEKTURE



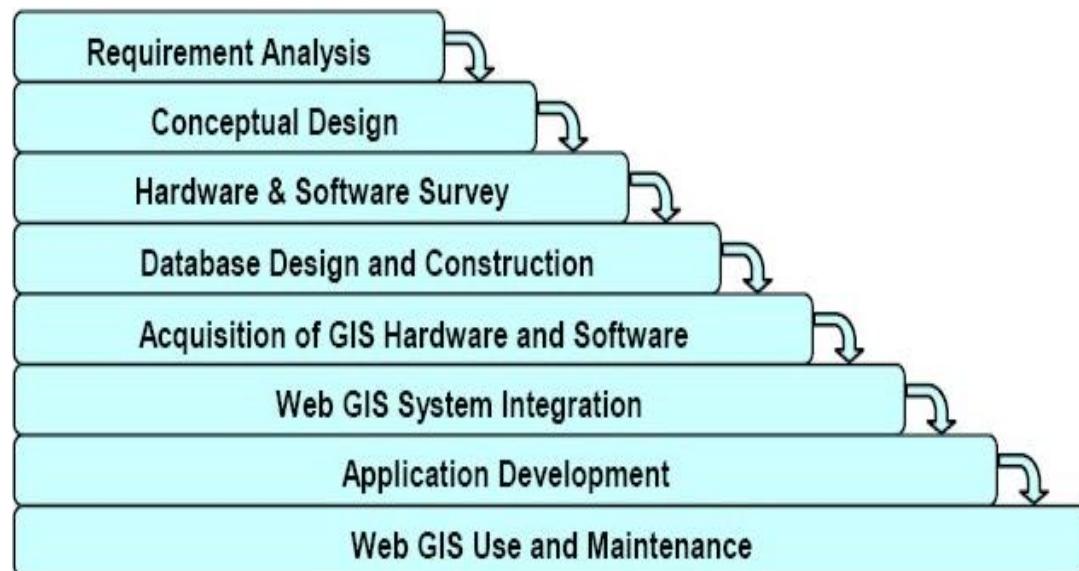
[izvor: AA. Alesheikh et.al., 2002]

Pristupi za obradu podataka u troslojnoj Web GIS arhitekturi

- **STRANA POSLUŽITELJA** - pristup omogućuje postavljanje zahtjeva na klijentima, te proslijđivanje zahtjeva za obradom prema web poslužitelju, odnosno poslužitelju karata. Zatim poslužitelji obrade zahtjeve i vraćaju rezultate udaljenom klijentu.
- **STRANA KLIJENTA** - pristup omogućuje postavljanje upita i izvođenje nekih obrada podataka lokalno na strani klijentskog računala.
- **STRANA POSLUŽITELJA/KLIJENTA ("Hibridni pristup")** – optimiziraju se performanse izvođenja (komunikacija klijent/poslužitelj) i osiguravaju posebni korisnički zahtjevi.

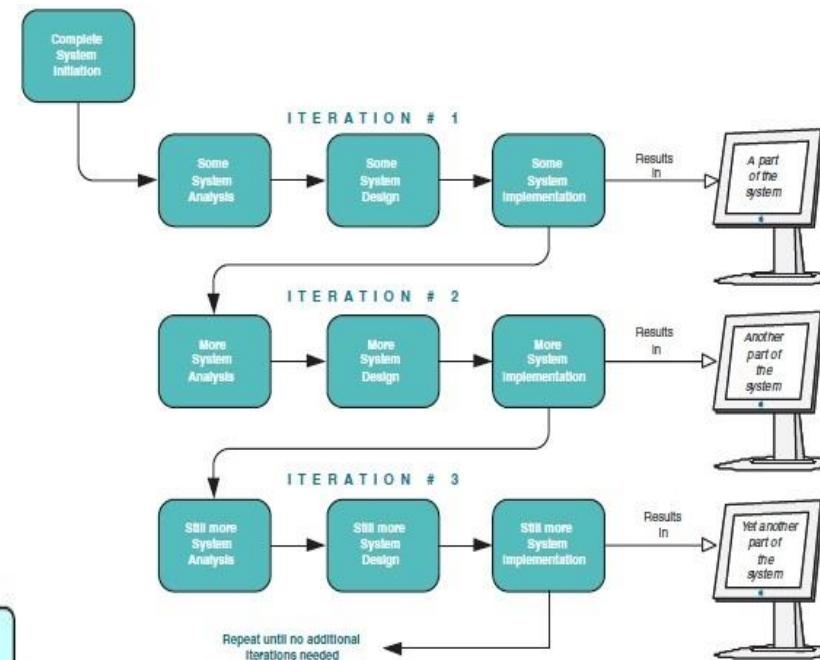
Faze razvoja Web GIS sustava

- Razvojni proces Web GIS sustava je iterativni i inkrementalni
- Započinje analizom zahtjeva i konceptualnim modeliranjem a završava korištenjem i održavanjem razvijenog WEB GIS sustava.



[izvor: AA. Alesheikh et. al., 2002]

**Primjer faza
WEB GIS razvojnog procesa**



[izvor: J. L. Whitten et. al.: *Systems Analysis and Design Methods, Sixth Edition*, 2004]

METODA RAZVOJA Informacijskog sustava za vremensko-prostorni prikaz složenih podataka

SVRHA ISTRAŽIVANJA



Koncept dinamičkih vremensko-prostornih prikaza na Web-u

Arhitektura integriranog informacijskog sustava za
vremensko-prostorni prikaz složenih podataka

Web-GIS-DBMS integracija

- Zaštita okoliša
- Promet (riječni promet)
- Statistika za okoliš
- Istraživanje i obrazovanje ...

SUVREMEN PRISTUP RAZVOJU

Informacijskog sustava za vremensko-prostorni prikaz složenih podataka

- *Objektno Orijentirani (OO) pristup razvoju*
- *UML (Unified Modeling Language)* jezik te njegove metode i dijagramske tehnike za *OO* modeliranje
- Iterativni i inkrementalni razvojni proces temeljen na *use-case* pristupu i modelu arhitekture sustava (višedimenzionalni pogled na sustav)
- Primjena suvremene objektne tehnologije (programski sustavi, razvojna pomagala i jezici)
- Interaktivne dinamičke aplikacije temeljene na troslojnoj arhitekturi za *Web (three-tier client/server)*
- MOGUĆNOST PROŠIRENJA: Primjena porodice *XML* tehnologija za objedinjavanje podataka iz različitih izvora

UML JEZIK I METODE ZA OO MODELIRANJE

- Dokazani *standard OMG* grupe za razvoj objektno orijentiranih sustava podržan od strane vodećih *CASE* proizvođača
- *Standardizirana notacija* temeljena na konceptima i najboljim praktičnim iskustvima triju najpoznatijih *OOD* metoda *OMT* (1991. god.), *Booch* (1994. god.) i *Objectory* (1992. god.)
- Predstavlja *vizualni jezik* za modeliranje, ali ne i vizualni programski jezik
- Temelji se na općem metamodelu (što objedinjuje semantiku) te na općoj notaciji (što osigurava prikazivanje te semantike)
- Sadrži "*standardnu*" *metodu za opis arhitekture sustava*, kao višedimenzionalnog pogleda na sustav (*UML dijagrami*)
- Promovira razvojni proces koji je vođen *use-case* pristupom, temeljen na *arhitekturi, iterativni* i *inkrementalni*

“Use case” pristup

- Predstavlja inkrementalni i iterativni pristup u kojem se prvo proizvodi izvršna arhitektura programskog sustava
 - Osigurava vanjski pogled na sustav

Ovaj pristup iskoristili smo za:

- ✓ **Modeliranje zahtjeva** koje sustav treba omogućiti
- ✓ **Pronalaženje osnovnih elemenata arhitekture** tokom faze objektnog modeliranja
- ✓ **Provjeru i prikaz arhitekture** sustava tokom analize

Iterativni i inkrementalni razvoj modela arhitekture sustava uključuje postupke za:

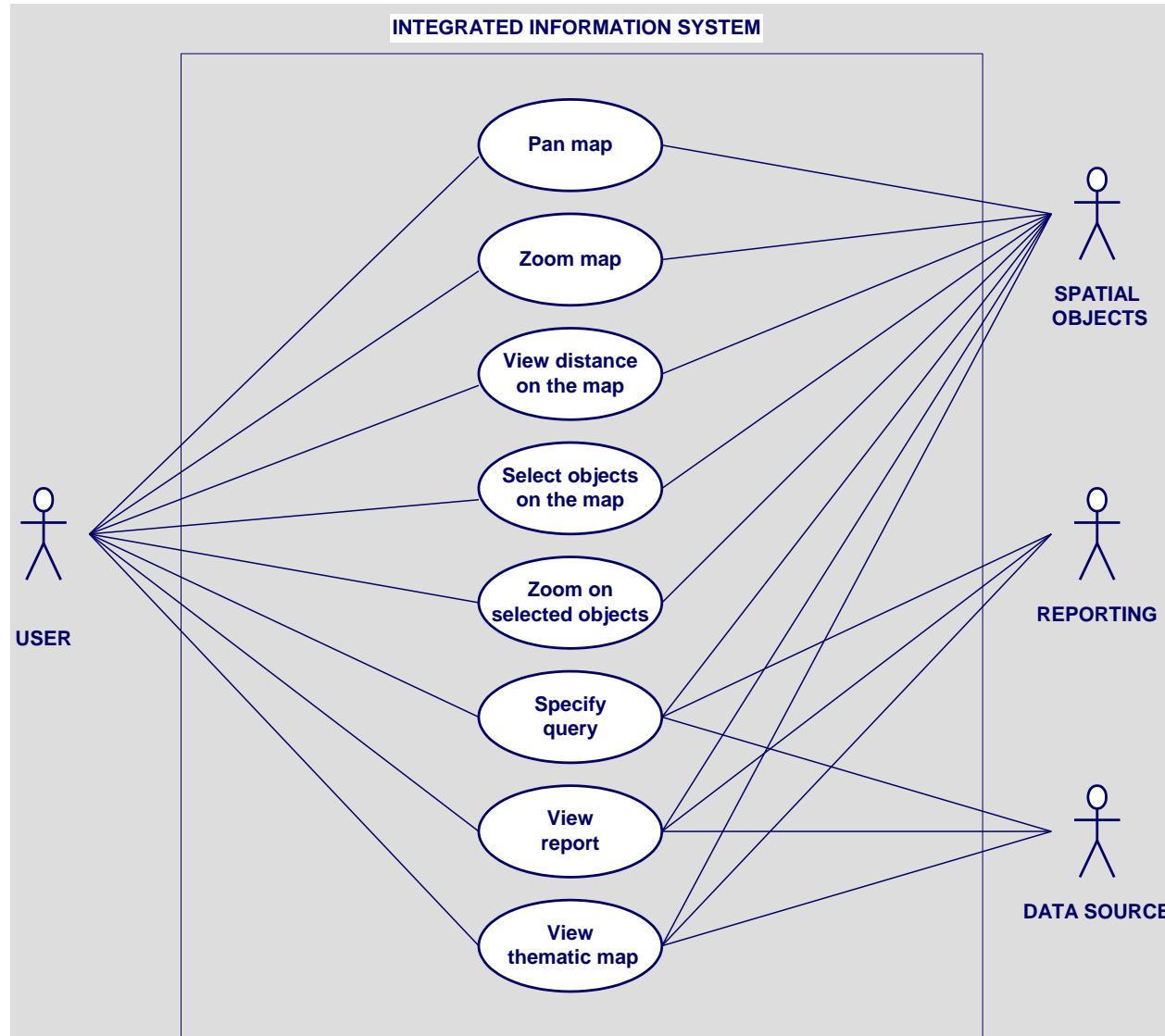
- ✓ Definiranje **struktura objekata** (paketi, klase i atributi)
- ✓ Definiranje **struktura ponašanja objekata** (operacije i metode)
- ✓ Definiranje **veza** između objekata kako bi se ostvarile osnovne funkcije sustava i zahtjevi

DEFINIRANJE OSNOVNIH FUNKCIJA SUSTAVA

IS treba osigurati sljedeće osnovne funkcije:

- Objedinjene karakteristike **GIS i DBMS** sustava (podrška osnovnih prostornih funkcija, učinkovita pohrana, pretraga, te obrada)
- Pravovremenu dostupnost informacija u najpogodnijem obliku (digitalne karte, dijagrami, tablice i izvještaji)
- Mogućnost korištenja distribuiranih izvora podataka (prostornih podataka i baza podataka)
- Postavljanje korisničkih zahtjeva
- Automatsko i dinamičko izvještavanje te generiranje tematskih karata za odabrane objekte s karte
- Sigurnost od neovlaštenog pristupa te zaštitu i integritet podataka

KORISNIČKI POGLED NA INTEGRIRANI IS

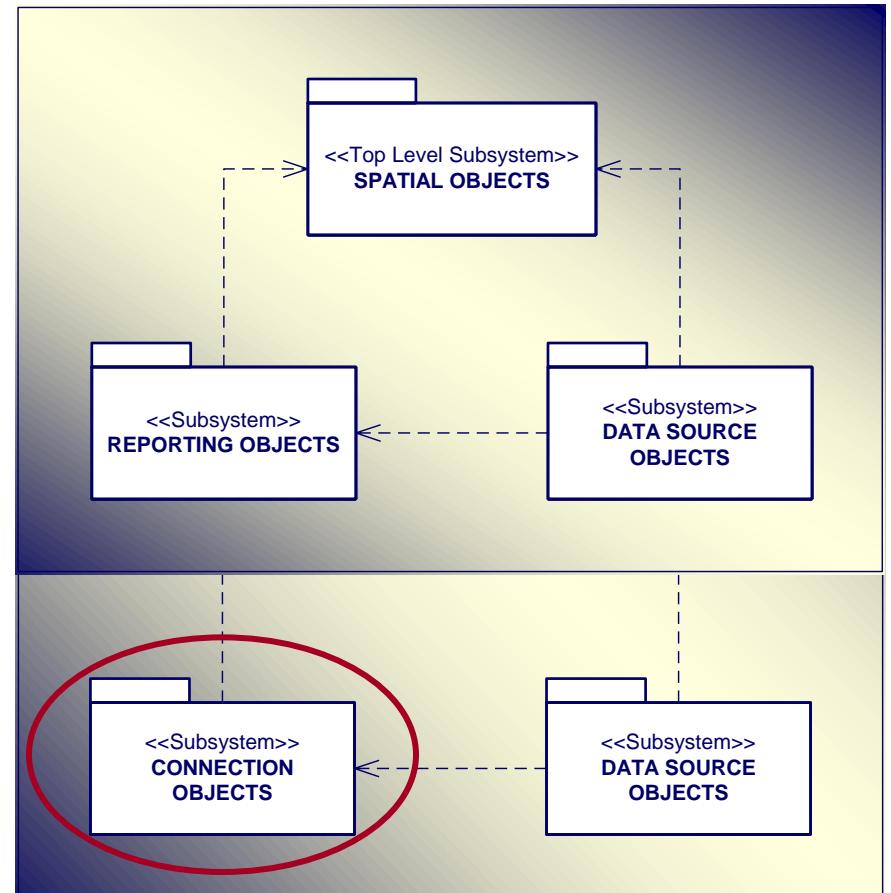


UML Use case dijagram

[J. Pečar-Ilić, 2001]

Osnovni paketi (podsustavi) i njihova međusobna povezanost, odnosno način komuniciranja

- UML dijagram paketa (package diagram) → UML dijagram klasa (class diagram) na najvišoj razini apstrakcije (isprekidana usmjerena linija opisuje ovisnost u međusobnoj komunikaciji)
- Pod sustav najviše razine – obuhvaća *prostorne objekte* (*Spatial Objects*)
- Pod sustav koji predstavlja izvor opisnih podataka, odnosno *bazu podataka* (*Data Source Objects*)
- Pod sustav za izvještavanje - *dinamičko izvještavanje* iz baze podataka preko prostornih objekata (*Reporting Objects*)



[J. Pečar-Ilić, 2001]

Konačni UML dijagram paketa

Osnovni objektni tipovi (klase) za opise prostornih objekata (paket *Spatial Objects*)

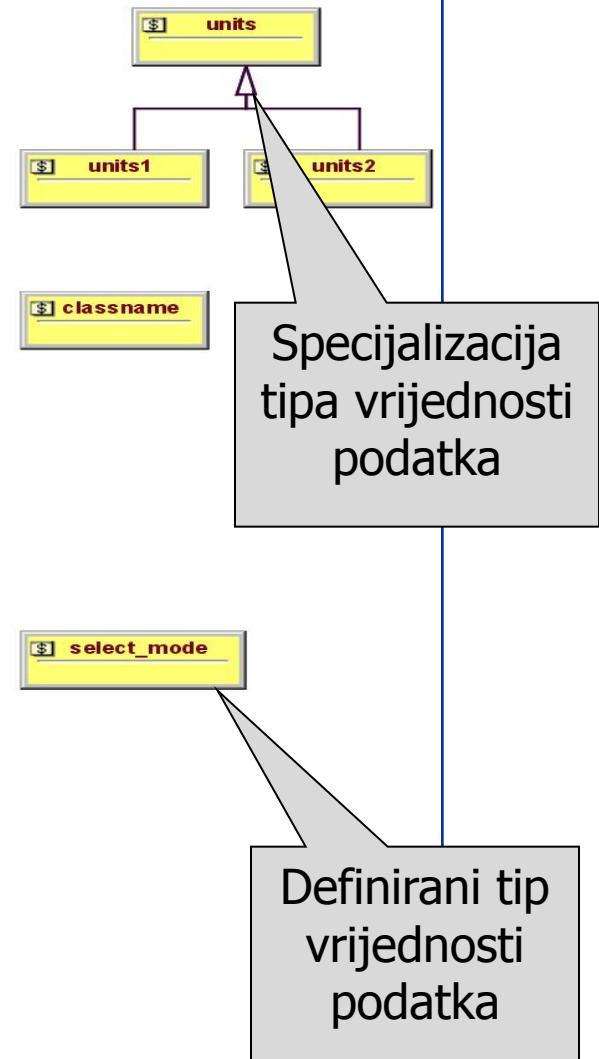
- *Map* – objektni tip koji predstavlja pojavu prozora *Viewer-a* i digitalne karte preko njega
- *MapLayer* – objektni tip koji predstavlja pojavu sloja karte u grafičkom prikazu, a koristi se kako bi se dobila svojstva slojeva karte (ime, prioritet, vidljivost, selektivnost i dr.) kao i lista *MapObject* objekata prikazanih u sloju karte
- *MapObject* – predstavlja pojavu objekta iz karte na grafičkom prikazu (gradovi, rijeke, granice sliva, država i dr.), a omogućuje dobivanje i promjenu svojstava objekata karte (ime, ključ, *URL* i dr.)
- *MapPoint* – predstavlja pojavu točke na digitalnoj karti definirane koordinatama položaja točke na površini zemlje
- *Collection* – predstavlja pojavu skupa različitih vrsta objekata
- *Selection* – predstavlja pojavu skupa odabranih objekata na karti, a koristi se za dodavanje ili oduzimanje *MapObject* objekata iz tog skupa

Objektni tip Map

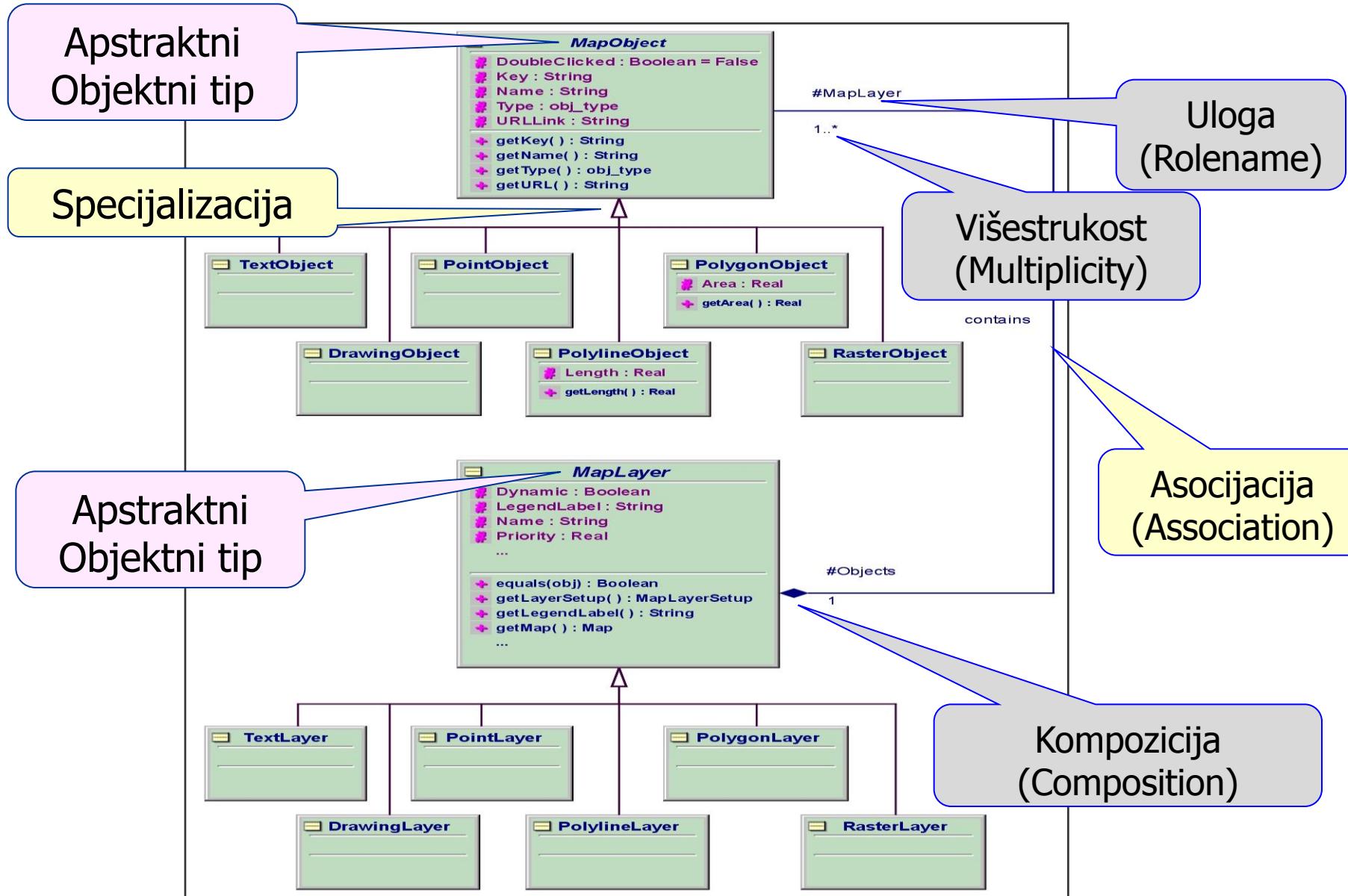
Atributi

zaštićena
vidljivost

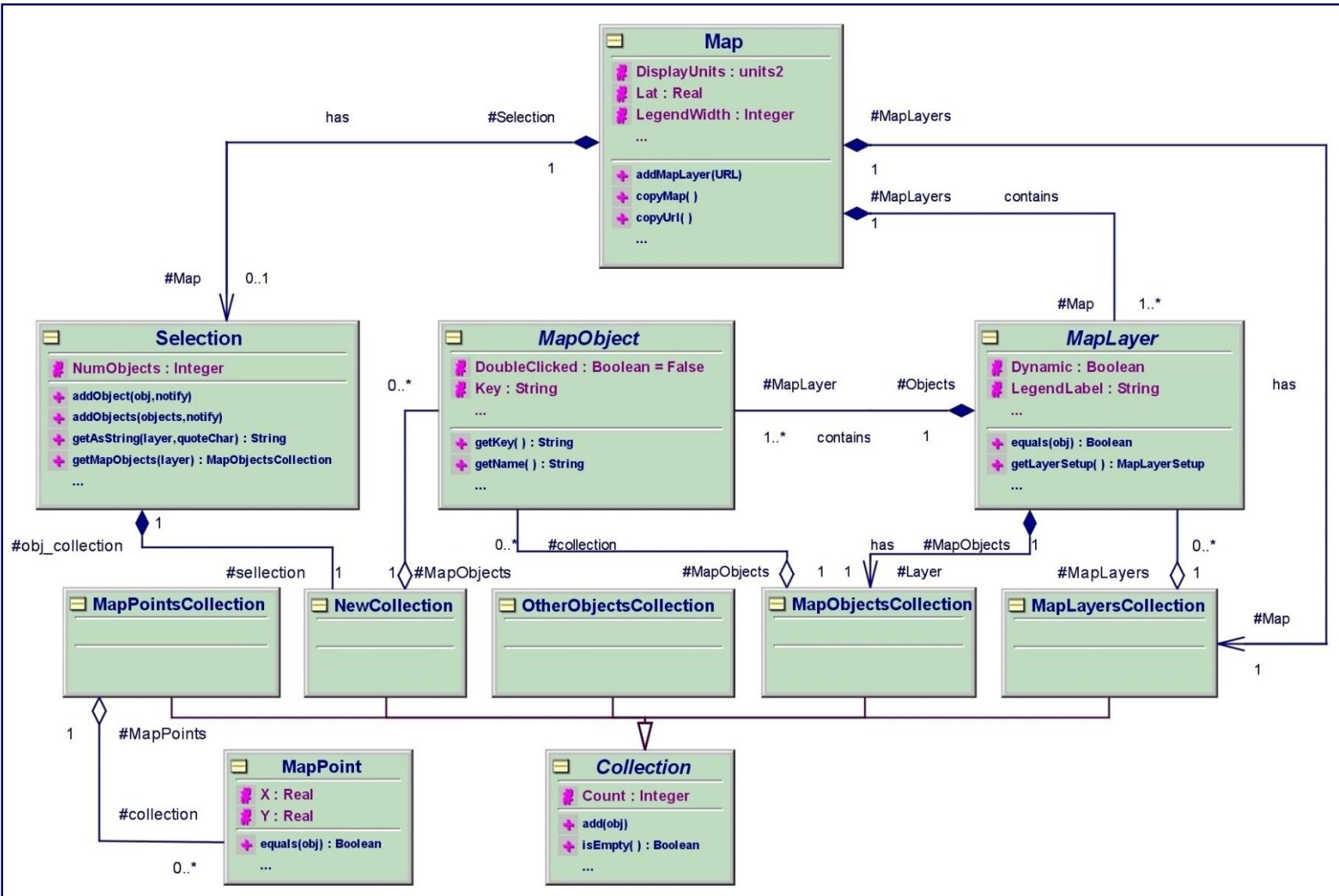
Operacije
+
javna
vidljivost



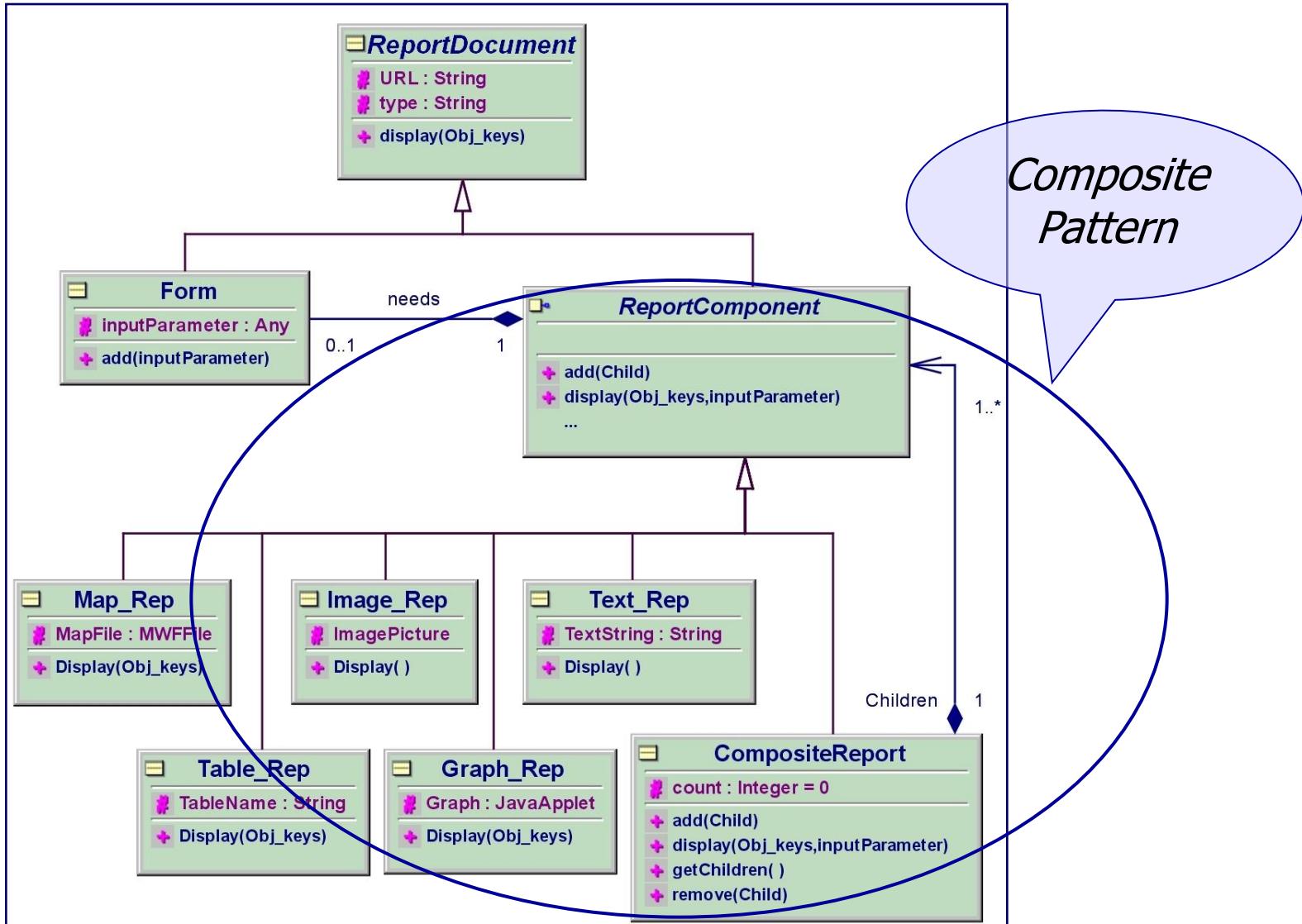
Specijalizacija slojeva karte i njihovih objekata



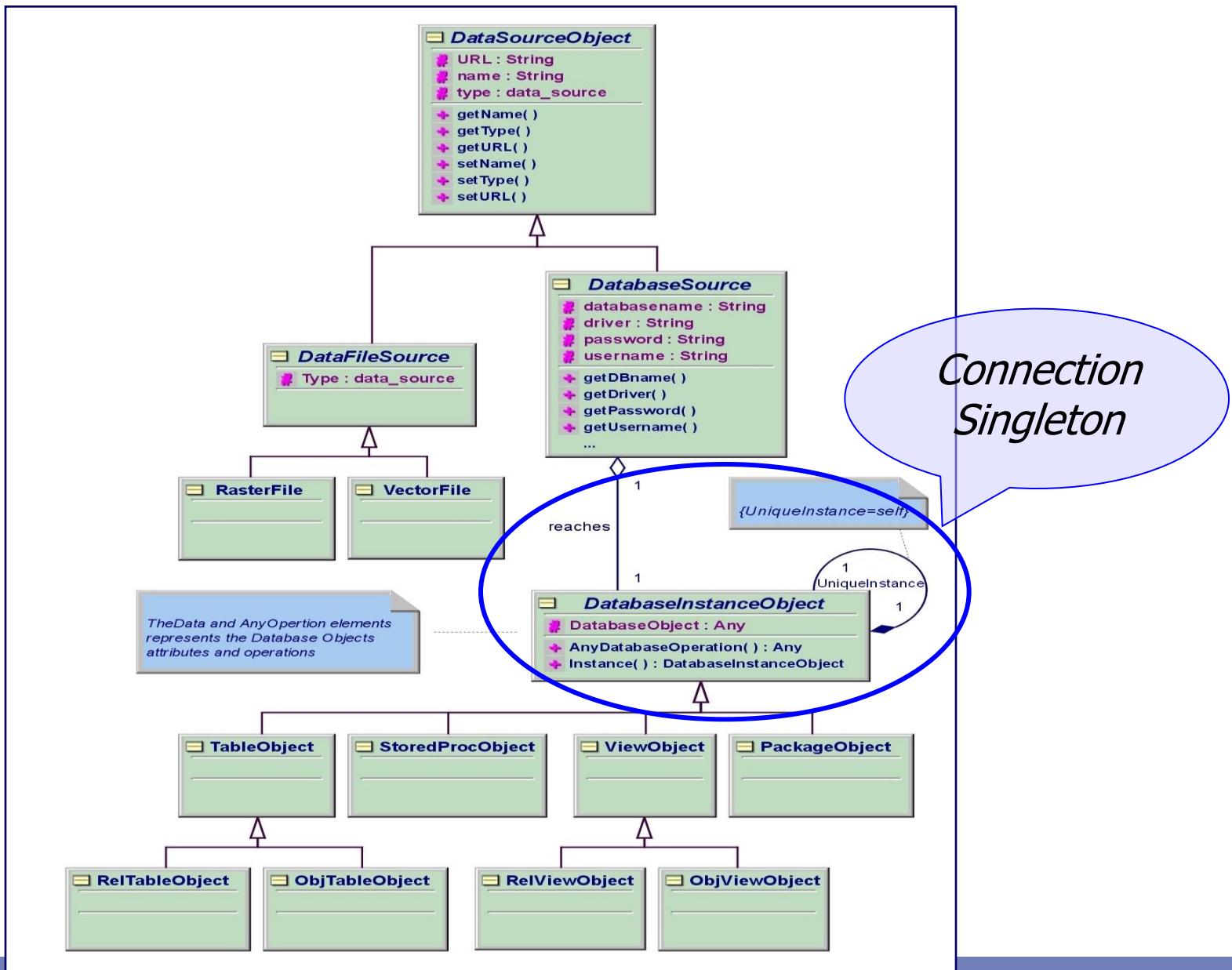
Detaljan prikaz paketa *Spatial Objects*



Paket *Reporting Objects* koji se temelji na uzorku dizajna *Composite Pattern*



Paket *Data Source Objects* temeljen na uzorku dizajna *Connection Singleton Pattern*



OSTVARENI REZULTATI

Informacijskog sustava za vremensko-prostorni prikaz složenih podataka

- Formalno je opisana arhitektura integriranog IS-a koja se temelji na izvršenoj Web-GIS-DBMS integraciji
- Formalno su opisani složeni odnosi između objekata kao i akcije u cilju dobivanja dinamičkih vremensko-prostornih prikaza, koji uključuju različite vrste izvještaja i tematske karte

DOPRINOS

Višestruka iskoristivost razvijenog sustava na najvišoj konceptualnoj razini apstrakcije:

ARHITEKTURA SUSTAVA

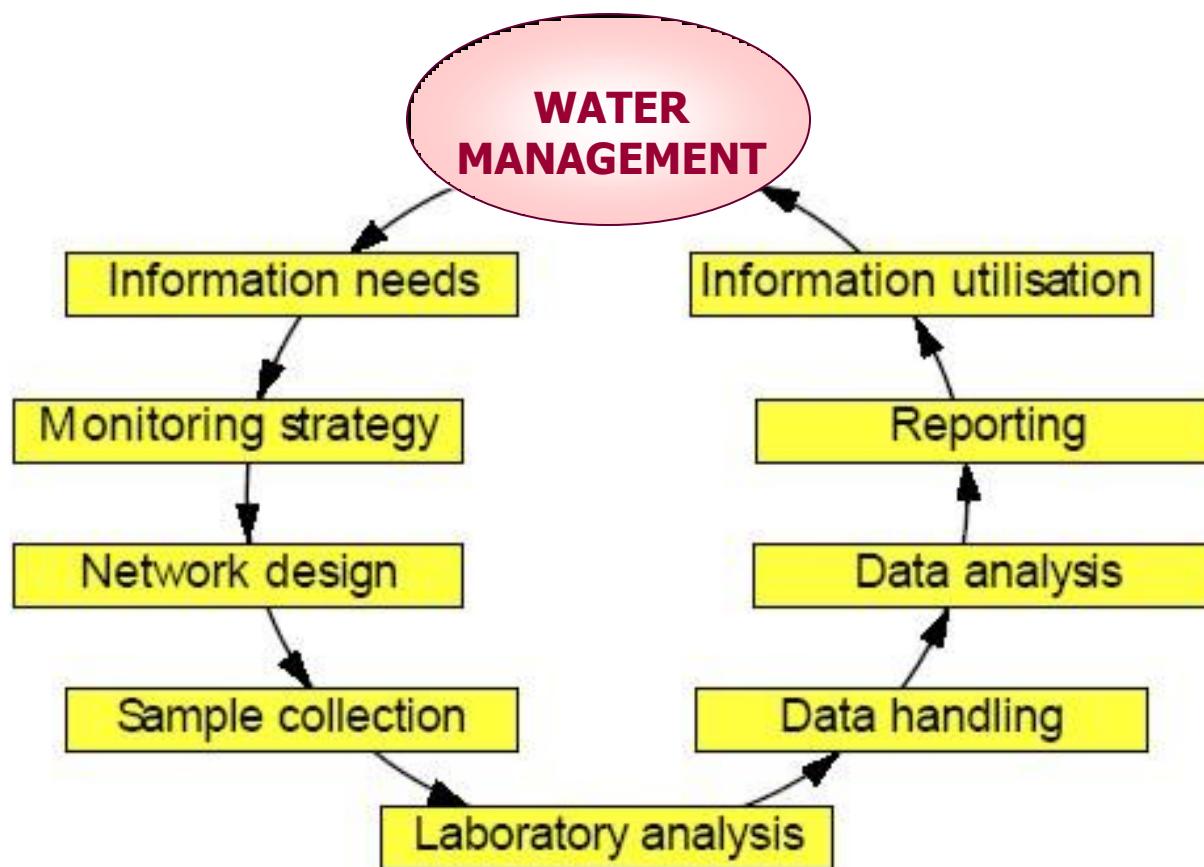
kao okvir za razvoj specijaliziranih IS-a
sličnih karakteristika za različita problemska područja

PRIKAZ REZULTATA SPECIJALIZIRANOG Web GIS-a

1. Specijalizirani informacijski sustav za *praćenje kakvoća površinskih voda u slivu rijeke Dunav*

Primjer 1: PRAĆENJE KAKVOĆE VODA U SLIVU RIJEKE DUNAV

- Međunarodni program (*Environmental Programme for the Danube River Basin*)
- Sinteza osnovnih ciljeva IV. i V. okvirnog programa *TAP* i *ISTP*
- *Trans-National Monitoring Network (TNMN)* – 12 podunavskih zemalja



Primjer 1: OSTVARENI REZULTATI IS-a za praćenje kakvoće voda u slivu rijeke Dunav

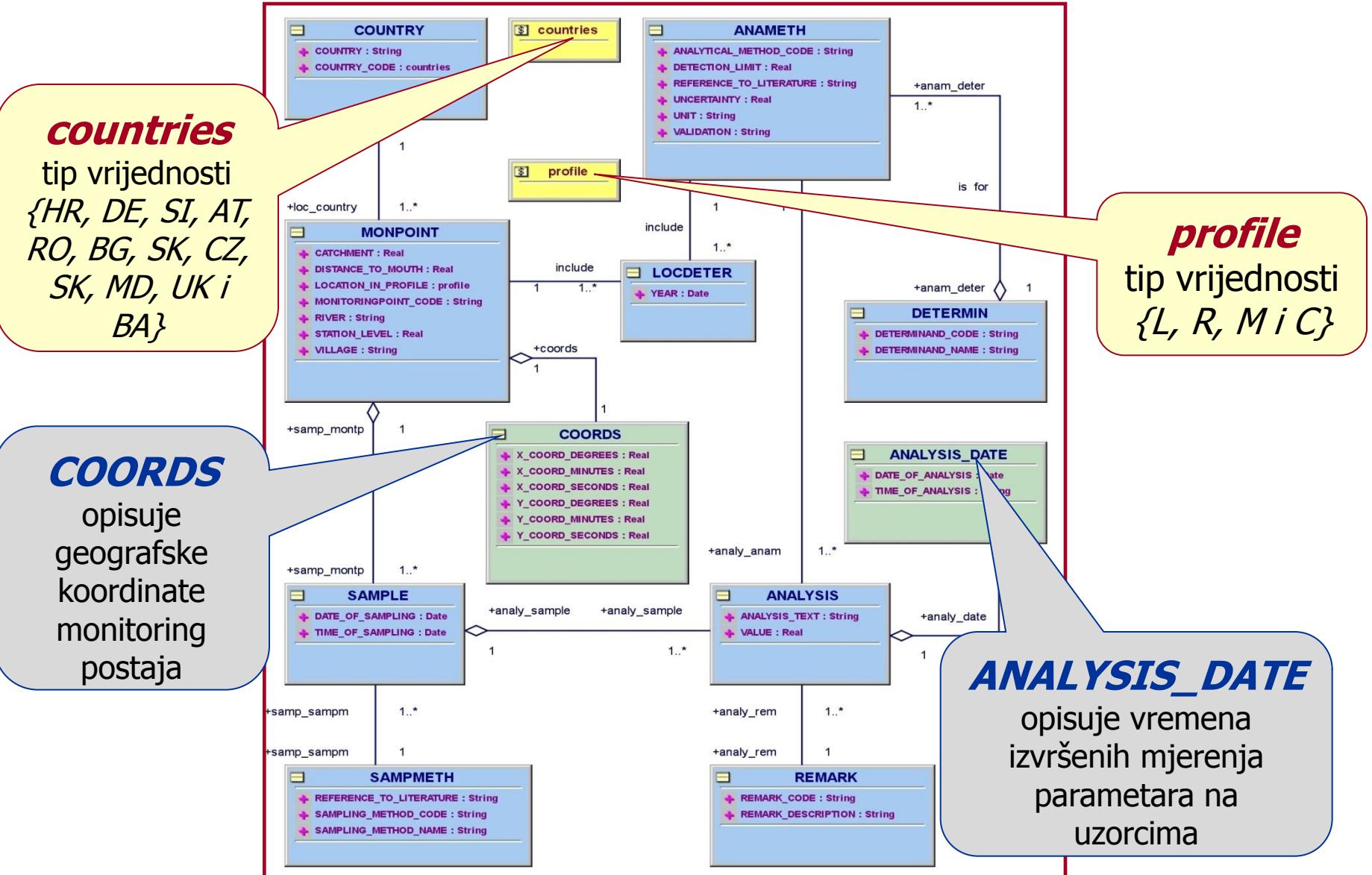
- Razvoj prema WEB-GIS-ORDBMS arhitekturi Integriranog sustava
- Web GIS aplikacija za ovlaštene korisnike EPDRB programa za prikaze kao tematske karte, dijagrami, izvještaji, ...
- Verifikacija (TNMN podaci, 12 zemalja, kao administrator 1996.-2000.)
- Nova originalna rješenja - učinkovitije upravljanje na osnovu pravovremene dostupnosti informacija i različitih vremensko-prostornih prikaza (nacionalna razina i čitav EPDRB)

Primjer 1: Primijenjeni programski sustavi alati i jezici

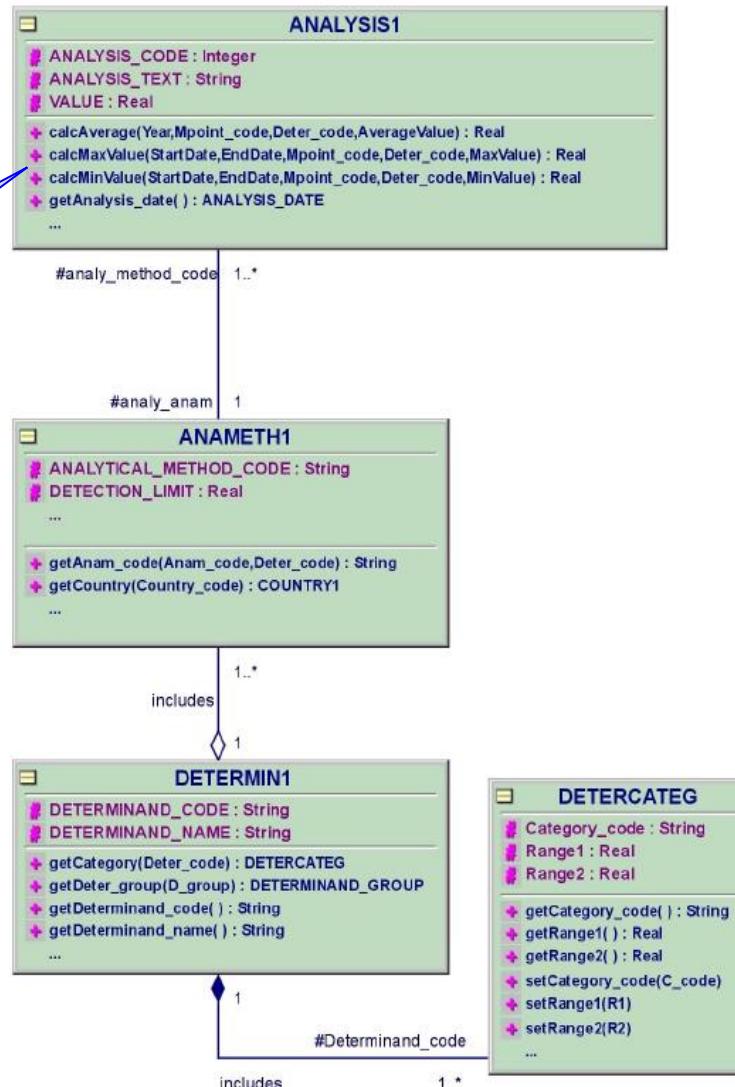
- *CASE pomagalo* za OO modeliranje i UML dijagrame – *Object Database Designer (ODD) Oracle™*
- *ORDBMS* za upravljanje objektno-relacijskom bazom podataka o kakvoći voda rijeka – *Oracle™ 9i Enterprise for Windows 2000*
- *WEB GIS* – Autodesk MapGuide™ alati:
Autodesk MapGuide™ Server (pristup vektorskim i rasterskim digitalnim kartama, te *ODBC* izvorima podataka preko intraneta, extraneta i Interneta), *Autodesk MapGuide™ Author* (izrada digitalnih karata čiji objekti su vezani uz izvještaje) te *Autodesk MapGuide™ Viewer* (za prikaz takvih karata)
- *Web-GIS-DBMS integracija* – *Active Server Pages (ASP)* sa *Microsoft® ActiveX Data Objects (ADO)*, *Java applets*
- *Web server* – *Microsoft® IIS 5.0*
- *Web preglednik* – *Microsoft® IE 6.0*

Primjer 1: Sustavno praćenje kakvoće voda

Dijagram klasa (ili dijagram objektnih tipova u Oracle ODD CASE alatu)



Primjer 1: Prikaz Dijagrama klasa za automatsku kategorizaciju srednjih vrijednosti parametara (tj. dijagram objektnih tipova u Oracle ODD CASE alatu)



DODATNO
UGRAĐENO
ZNANJE

IMPLEMENTACIJA OBJEKTNOG MODELA U ORDBMS ORACLE

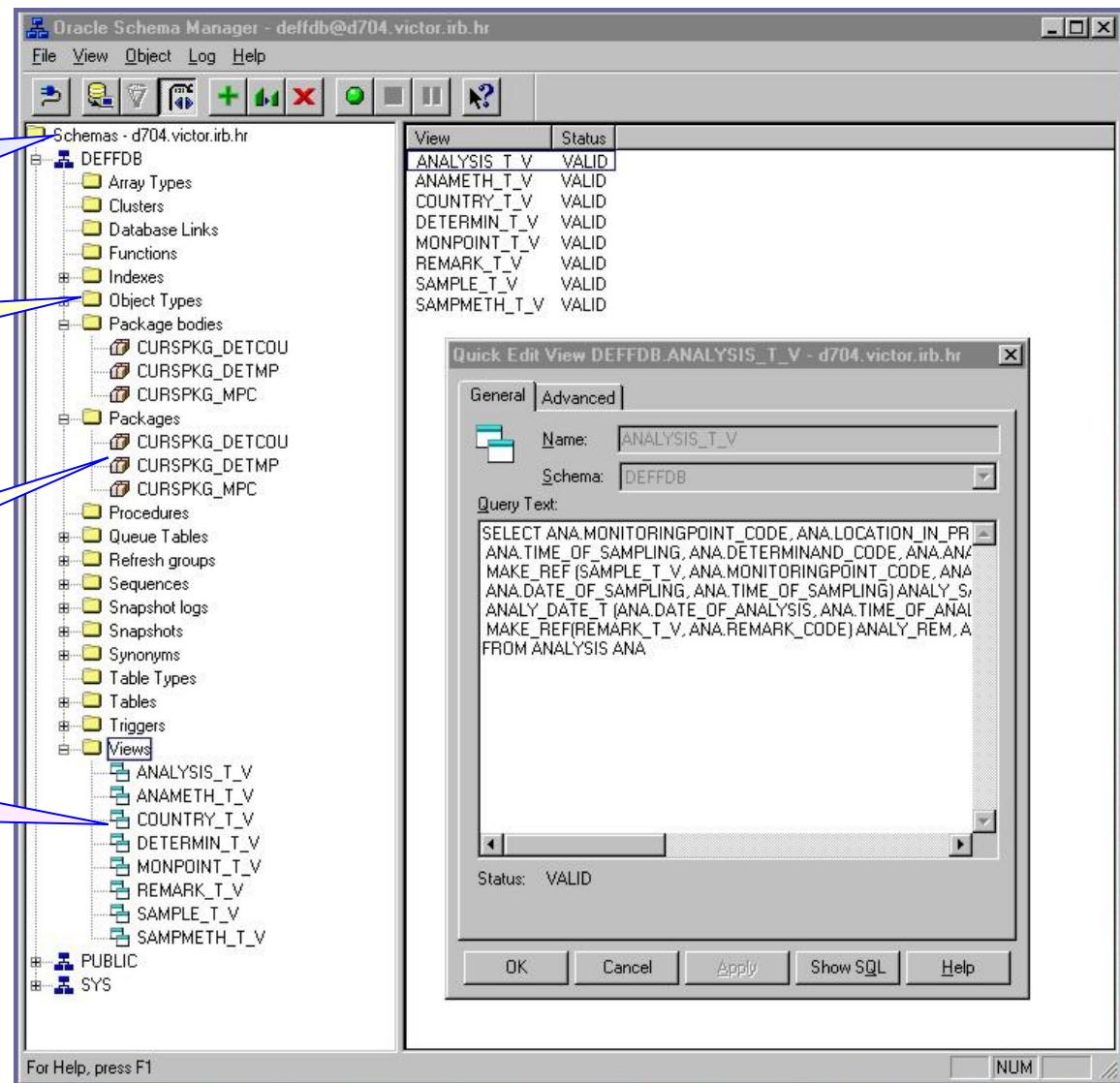
Objekti koji su stvorenici u korisničkoj shemi *deffdb*

KORISNIČKA
SHEMA
deffdb

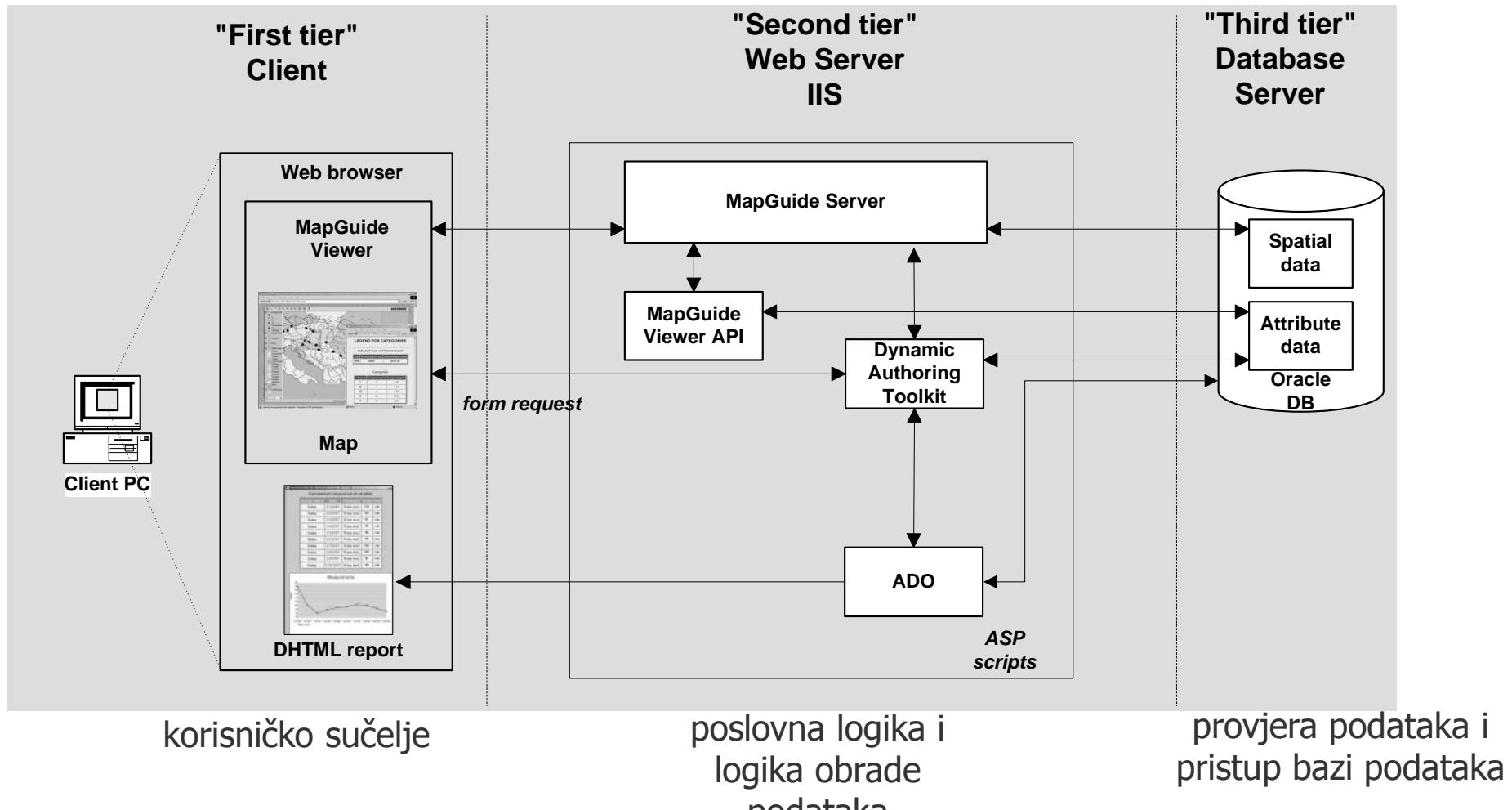
Oracle 8
objektni
tipovi

Oracle 8
paketi

Oracle 8
objektni
pogledi



Primjer 1: Arhitektura Web aplikacije



[J. Pečar-Ilić, 2001]

*Three-tier client-server arhitektura Web aplikacije za
automatsko generiranje karata, dijagrama i izvještaja*

IZRADA DINAMIČKIH IZVJEŠTAJA I TEMATSkiH KARATA PRIMJENJUJUĆI *ASP SKRIPTe*

Dinamička Web aplikacija – kombinacija:

- dinamičkih *HTML* stranica (*DHTML, Dynamic HTML*) i
- *ASP* skripata (realizacija paketa *Reporting Objects*).

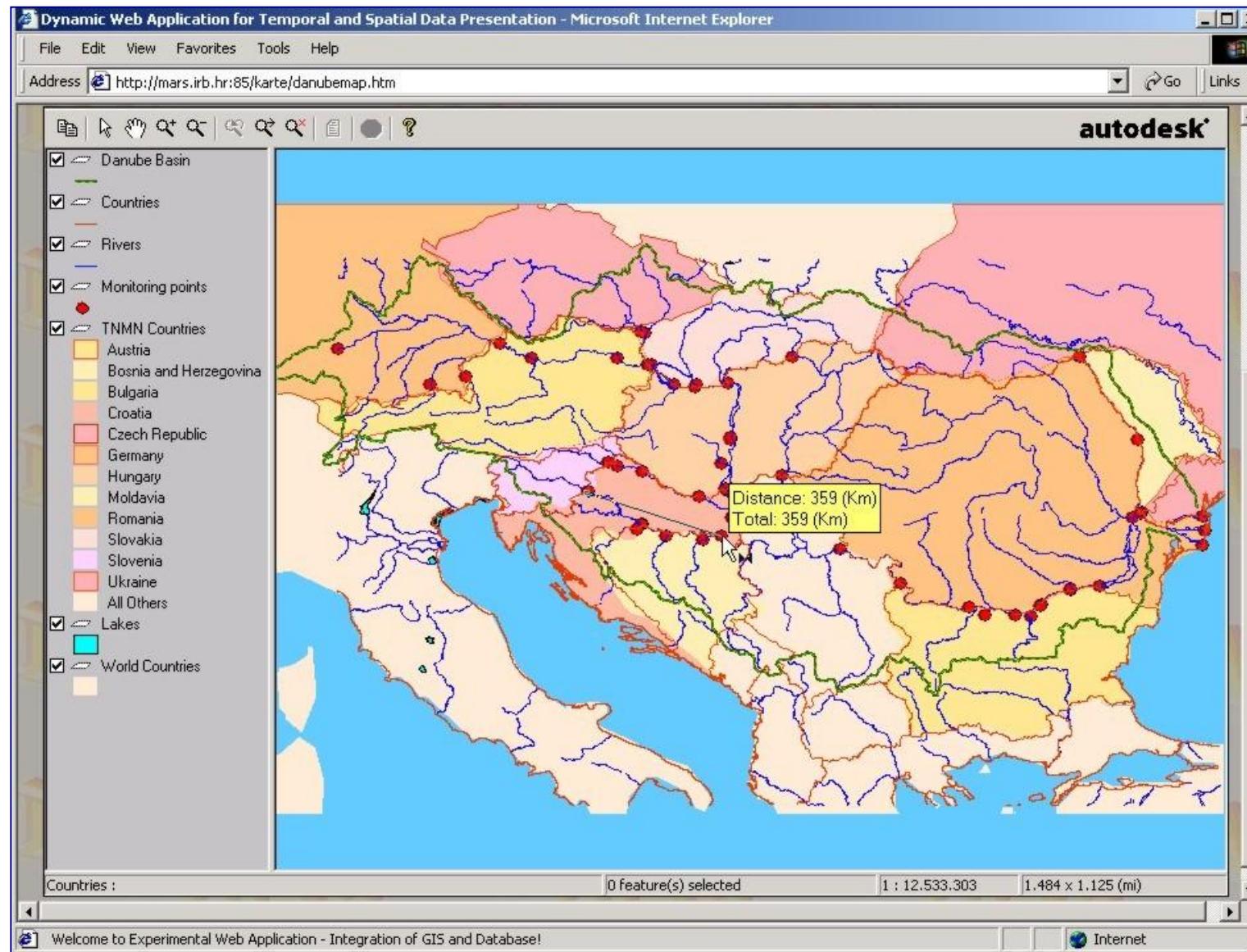
Dinamičnost:

- **Dinamičko *HTML (DHTML)* ponašanje** - pridodavanje dinamičkog ponašanja inače statičkom sadržaju *HTML* stranica
- **Dinamičke *HTML* stranice** (generirane na osnovu *ASP* skripata) – promjenjivog sadržaja (kao odgovor na unos podataka preko forme ili rezultati postavljenog upita nad bazom podataka)

Pomoću *ASP skripata*:

- Izvođenje *SQL upita* na osnovu proslijedjenih parametara
- Dobivanje rezultata toga upita u *HTML* formatu uključujući i dijagrame vremenskog niza (temeljeni na *Java Applets*)
- Korištenje *ADO objekata* i *Oracle 8 paketa* (povećanje performansi)

Primjer 1: INTERAKTIVNA DIGITALNA KARTA Dunavski sliv



[J. Pečar-Ilić, 2001]

Primjer 1: GENERIRANJE DINAMIČKOG IZVJEŠTAJA

The screenshot illustrates a dynamic web application for temporal and spatial data presentation, specifically for the Danube Basin. The interface includes a map of the basin with various monitoring points marked. A 'View Report' dialog box is open, providing general information about monitoring points and their determinants. Another window shows a form for selecting a time period and choosing specific determinants.

Main Window (Top Left):

- Address: http://mars.irb.hr:85/karte/danubemap.htm
- Map of the Danube Basin showing monitoring points across several countries.
- Left sidebar with a tree view of data layers:
 - Danube Basin
 - Countries
 - Rivers
 - Monitoring points
 - TNMN Countries
 - Austria
 - Bosnia and Herzegovina
 - Bulgaria
 - Croatia
 - Czech Republic
 - Germany
 - Hungary
 - Moldavia
 - Romania
 - Slovakia
 - Slovenia
 - Ukraine
 - All Others
 - Lakes
 - World Countries

View Report Dialog Box:

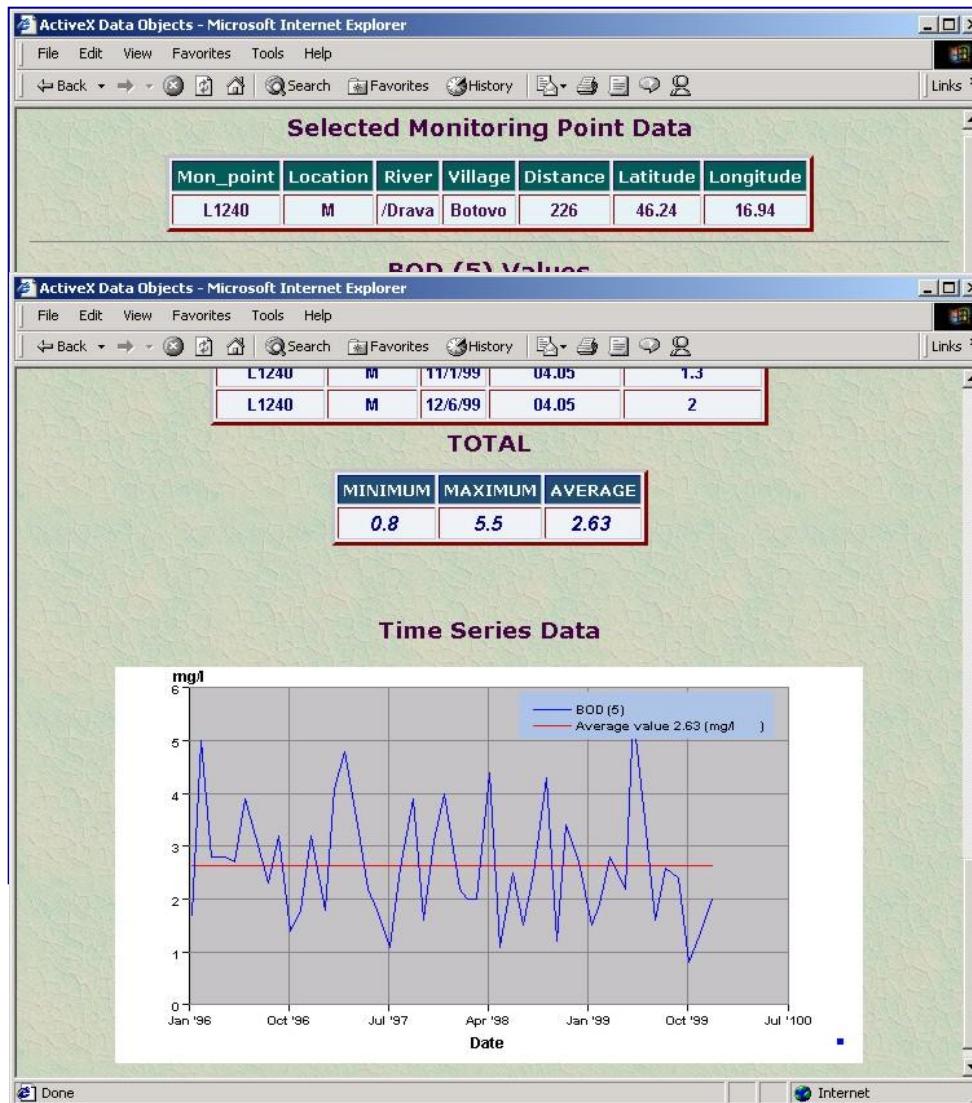
- General Information for Monitoring Points
- Determinands... Time Series for Monitoring Point
- Determinands for Selected Monitoring Points
- Integration with ISEP
- Tematic Maps

Second Window (Bottom Right):

- Title: Determinands from Selected Monitoring Point in Defined Time Period
- Form fields:
 - Start Date (dd-MON-yyyy): 01-JAN-1996
 - End Date (dd-MON-yyyy): 31-DEC-1999
- Dropdown menu of determinants:
 - 04.05BOD (5)
 - 03.65Cadmium (Cd)
 - 03.70Mercury (Hg)
 - 03.75Nickel (Ni)
 - 04.05BOD (5)
 - 04.10COD (Cr)
 - 04.15COD (Mn)
 - 04.30Phenol index
 - 04.35Anionic active surfactants
 - 04.40Petroleum hydrocarbons
 - 06.40Saprobic / biotic index

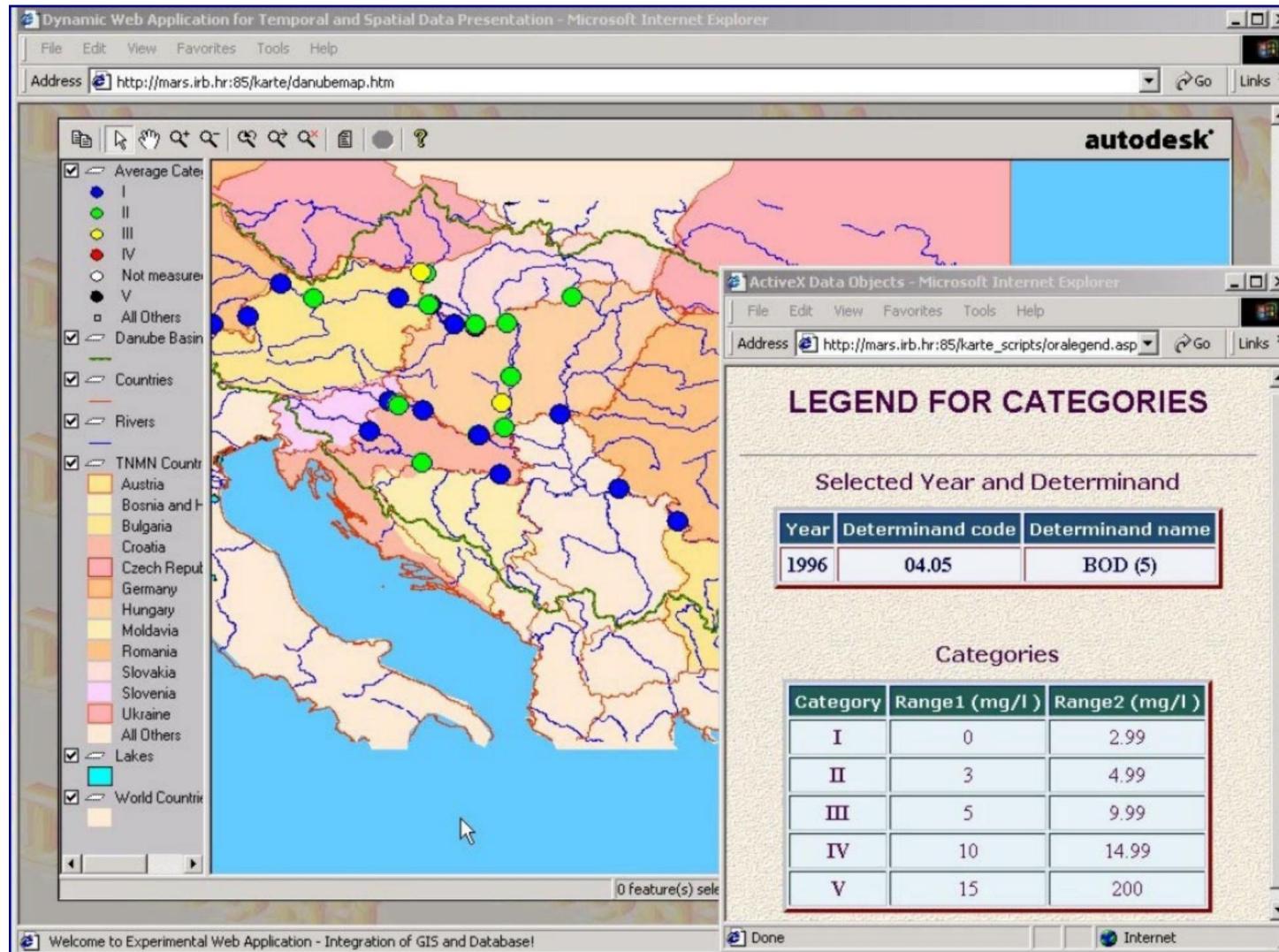
[J. Pečar-Ilić, 2001]

Primjer 1: VRIJEDNOSTI ZA BOD5 PARAMETAR



[J. Pečar-Ilić, 2001]

Primjer 1: Dinamička Web Aplikacija za prikaz kakvoće površinskih voda u slivu rijeke Dunav



[J. Pečar-Ilić, 2001]

Tematska karta s BOD5 prostornom raspodjelom u slivu rijeke Dunav

ZAKLJUČCI

- Metoda projektiranja i izgradnje specijaliziranih IS-ova za područja zaštite okoliša te npr. utjecaja riječnog prometa - strateški dugoročni pravci istraživanja u RH.
 - OO pristup, UML jezik, razvoj temeljen na modelu arhitekture sustava (integracija GIS-a i objektno-relacijskih baza podataka)
 - Dinamičke aplikacije za Web uz primjenu XML tehnologija
- Istraživanja u području **Informatike o okolišu** (*Environmental Informatics* ili kraće *Enviromatics*) - znanstvene i tehničke aktivnosti u upravljanju informacijama o okolišu.

Korištenje UML-a za oblikovanje baze podataka

Upravljanje podacima
ak.god. 2015./2016.

UML (za potrebe oblikovanje baze podataka)

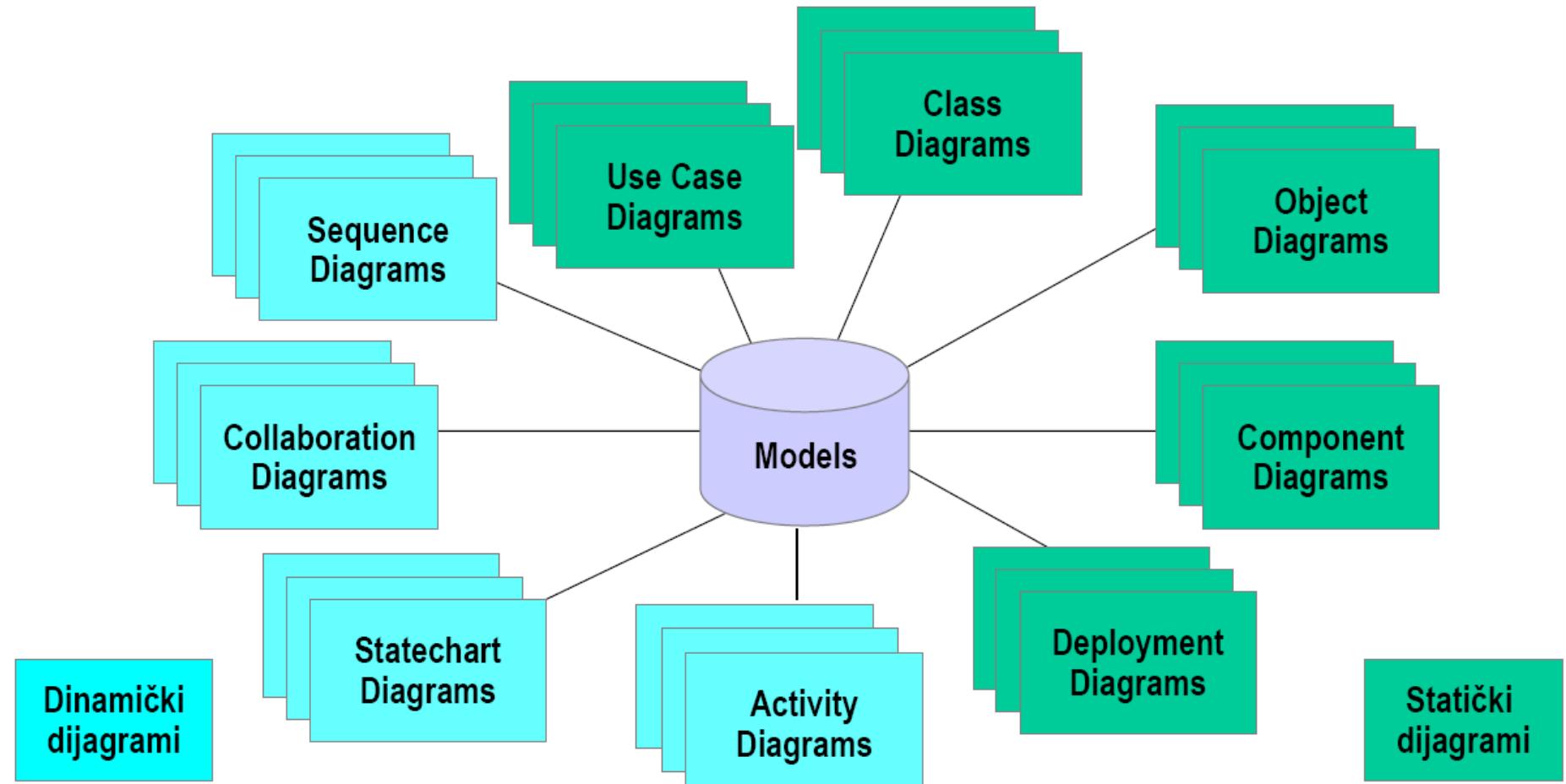
Zavod za telekomunikacije

- ◆ UML – skraćenica od Unified Modeling Language
- ◆ UML je grafički jezik pomoći kojeg prikazujemo objektno orijentirane paradigme
- ◆ Koristi se za:
 - Vizualizaciju
 - Specifikaciju
 - Izradu modela
 - Dokumentiranje

UML (za potrebe oblikovanje baze podataka)

Zavod za telekomunikacije

- ◆ UML – primarno se koristi se za razvoj programske podrške
- ◆ Oblikovanje podataka (engl. data modeling) – sve više se koristi UML za prikaz E-R dijagrama (dijagrami koji opisuju relacijsku bazu podataka)
- ◆ UML – omogućava opis cijelokupnog procesa razvoja relacijske ili objektno relacijske baze od zahtjeva (engl. business requirements) do fizičkog modela



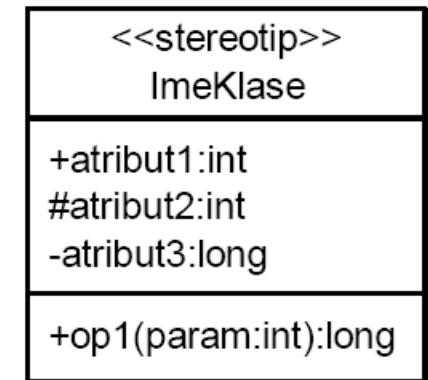
Elementi za oblikovanje ER dijagrama

Zavod za telekomunikacije

- ◆ Postoje 4 glavna elementa koja se koriste za oblikovanje ER dijagrama:
 - Entiteti
 - Atributi
 - Tipovi veza
 - Atributi veza

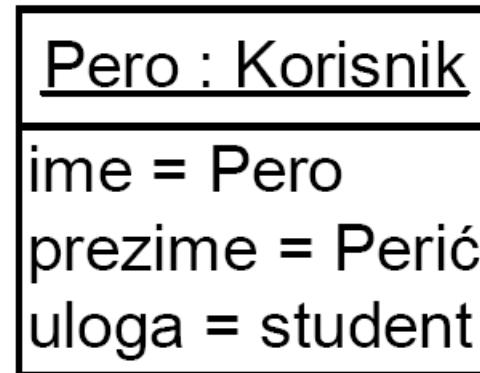
Dijagram klasa

- ◆ Dijagram klasa – opisuje tipove klasa i veze koje postoje između njih
- ◆ U dijagramu klasa su navedeni atributi klase
- ◆ Klasa je slična konceptu entiteta
- ◆ Ime klase treba biti smisленo i izvučeno iz domene koju opisujemo



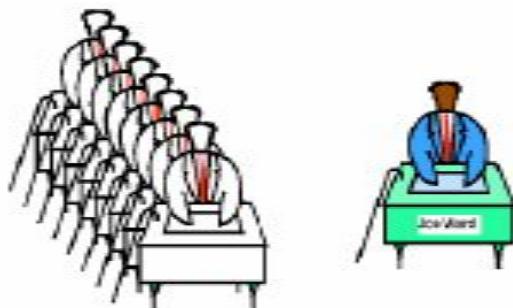
Dijagram objekata

- ◆ Dijagram objekata – opisuje sustav u pojedinom trenutku
- ◆ Prikazuje stvarne podatke
- ◆ Konceptualno sličan ER dijagramu (entiteti-veze)



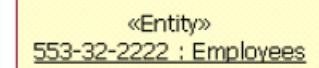
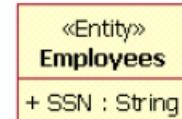
Prikaz entiteta

- ◆ Entitet: bilo kakav objekt iz pojavnog svijeta koji može biti stvaran ili apstraktan
- ◆ Tip entiteta: objekti iz pojavnog svijeta s istim karakteristikama



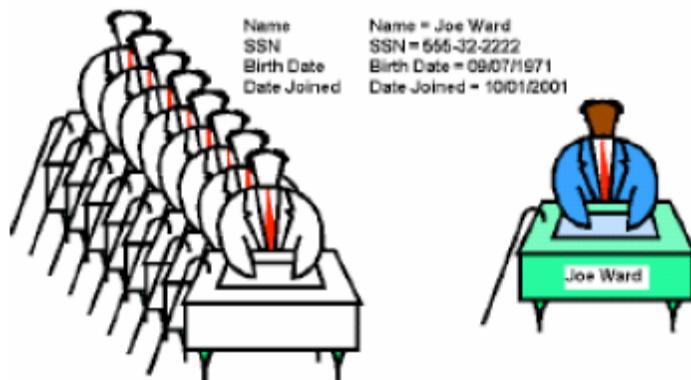
Tip entiteta: **Zaposlenik**
Entitet: **Matko Horvat**

Tip entiteta **Zaposlenik** i entitet 522 390 234 predstavljeni kao dijagram klase i dijagram objekta u UML-u



Prikaz atributa

- ◆ Atribut: jednoznačno opisuje određenu vrstu svojstva



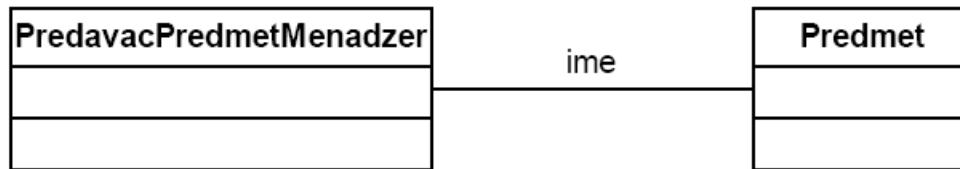
Atributi tipa entiteta **Zaposlenik** i vrijednosti za entitet **Matko Horvat**

Tip entiteta **Zaposlenik** s javnim atributima:
SSN, ime, adresa, god_rod, dat_zap, pozicija

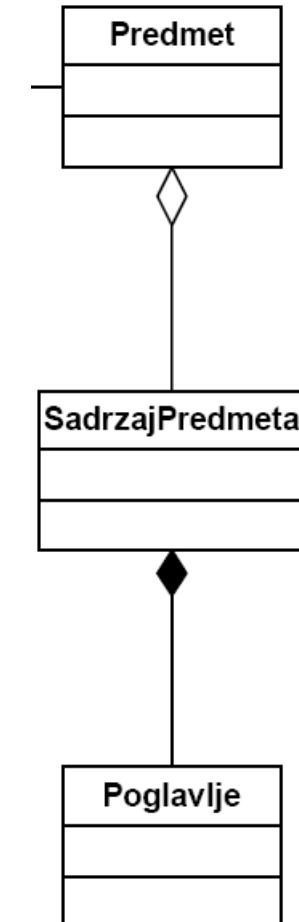
| «Entity» Employees | |
|-----------------------|-------------------|
| + | SSN : String |
| + | name : String |
| + | address : String |
| + | birthdate : Date |
| + | datejoined : Date |
| + | position : String |

Relacije u dijagramu klasa

- ◆ Relacije (tj. veze) u dijagramu klasa:
 - Asocijacija – dvostruka veza između klasa



- Agregacija – klasa je dio neke druge “veće” klase
- Kompozicija – isto kao agregacija, osim što vrijedi egzistencijalna i identifikacijska i ovisnost

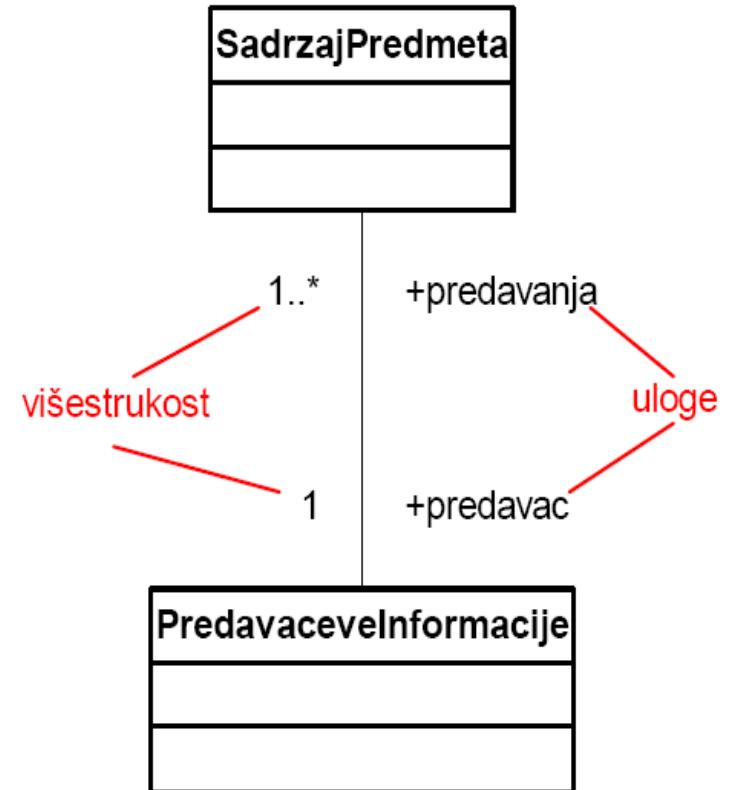


Relacije u dijagramu klasa

- ◆ Veze u dijagramu klasa:
 - Uloge i kardinalnost

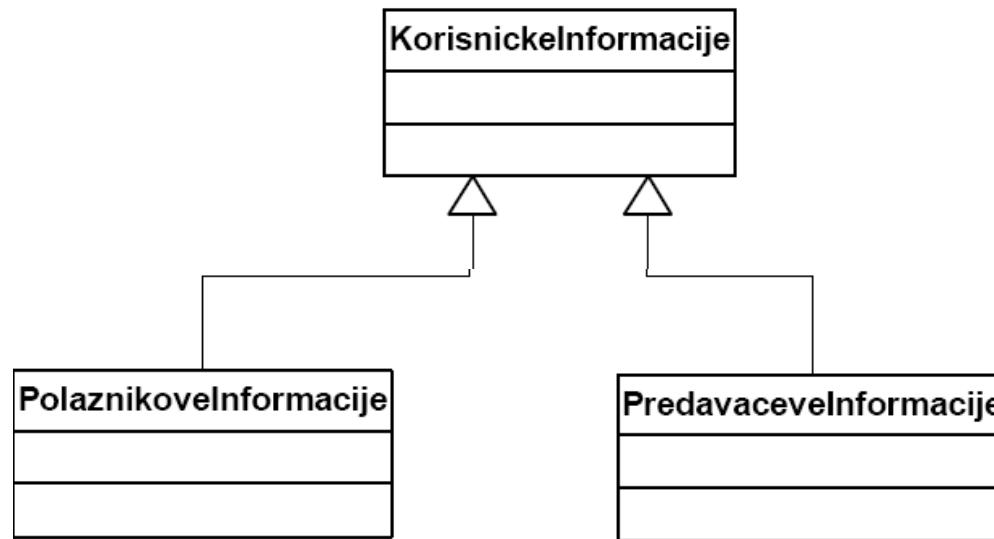
- ◆ Uloga:
 - Imenica koja označava ulogu u vezi
 - Stavlja se pokraj simbola klase

- ◆ Kardinalnost:
 - Brojčano ograničenje
 - * (0 ili više)
 - 1 (točno jedan)
 - 1..3 (jedan i tri)
 - 0..3 (od nula do tri)



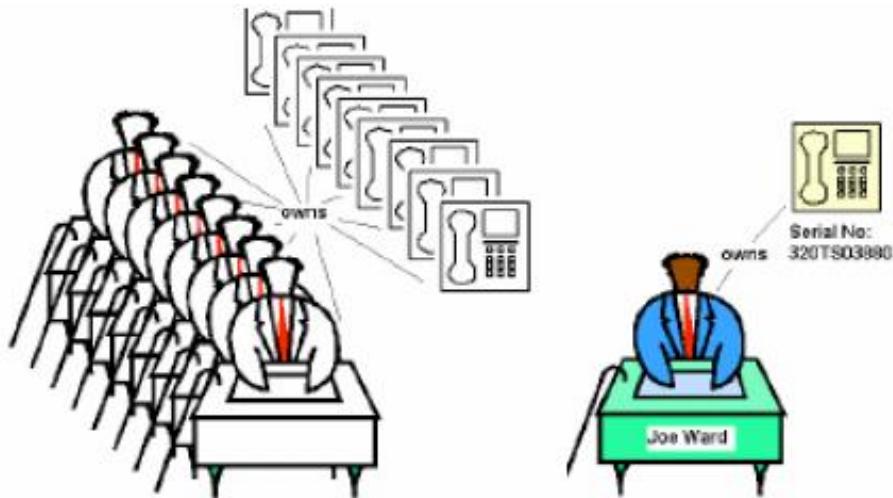
Nasljeđivanje

- ♦ Nasljeđivanje – koncepti generalizacije i specijalizacije



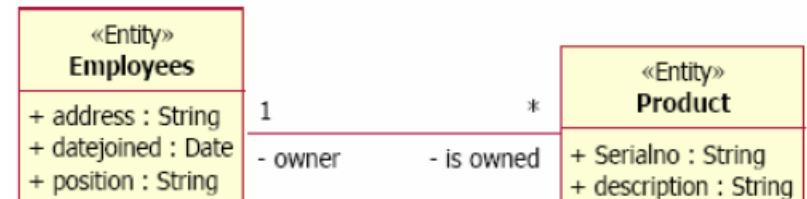
Veze

- Tip Veze: opisuje smisleni odnos koji postoji između entiteta



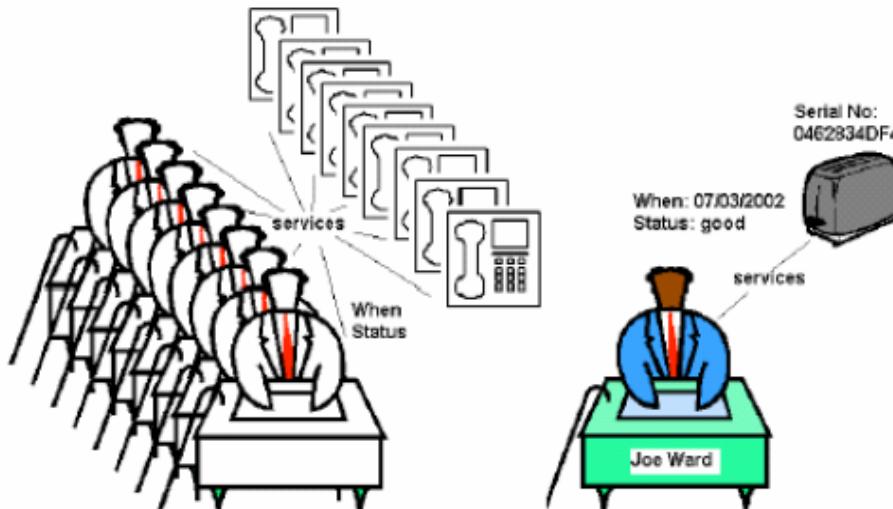
Veza između tipa entiteta **Zaposlenik** i **proizvoda** kojeg zaposlenik posjeduje

Atributi tipa entiteta **Zaposlenik** i vrijednosti za entitet **Matko Horvat**

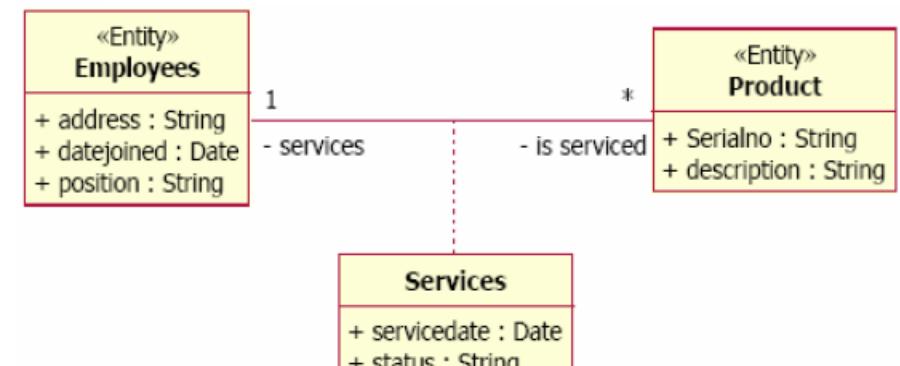


Veze

- Atributi veze: veze se mogu opisati atributima kao i entiteti



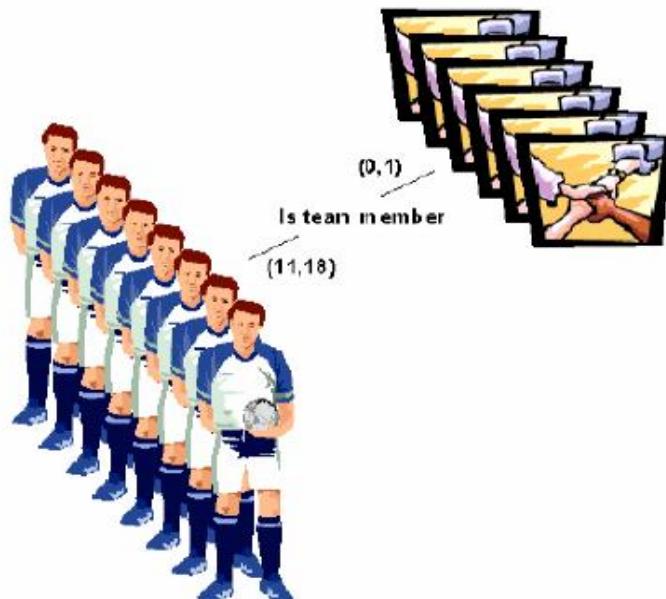
Atributi za tip veze **Usluge** između zaposlenika i uređaja (**Vrijeme i Status**)



Atributi veze **Usluge** specificirani su u dijagramu klase korištenjem asocijacija

Kardinalnost

- ◆ Kardinalnost: svaki tip entiteta specificira kardinalnost veze prema drugom entitetu



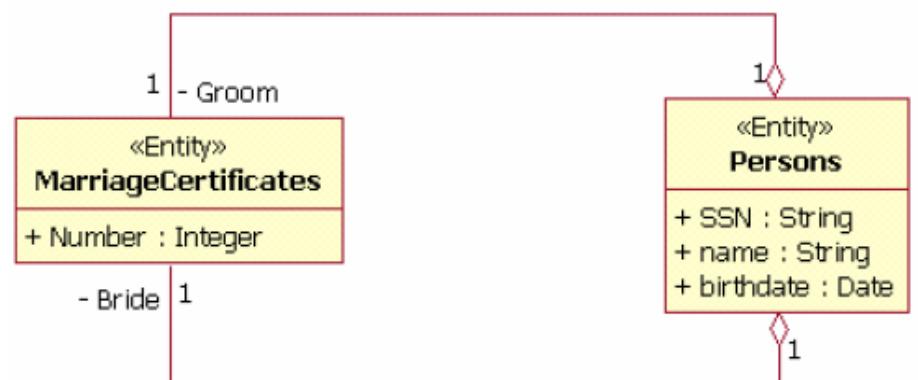
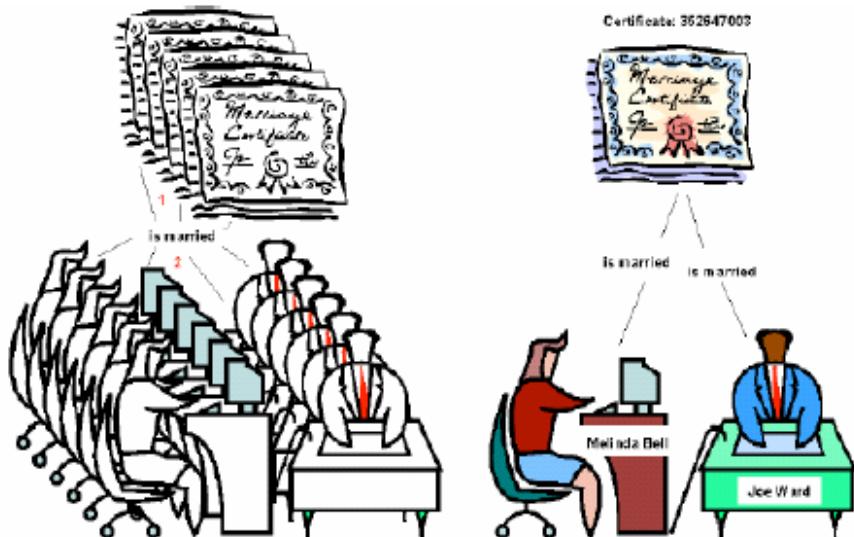
Veza **Igraju** definira broj igrača koji mogu sudjelovati (između 11 i 18).

Između entiteta **Nogometni tim** i **Igrači** Postoji veza **Igraju** čija kardinalnost je postavljena između 11 i 18.



Ovisnost

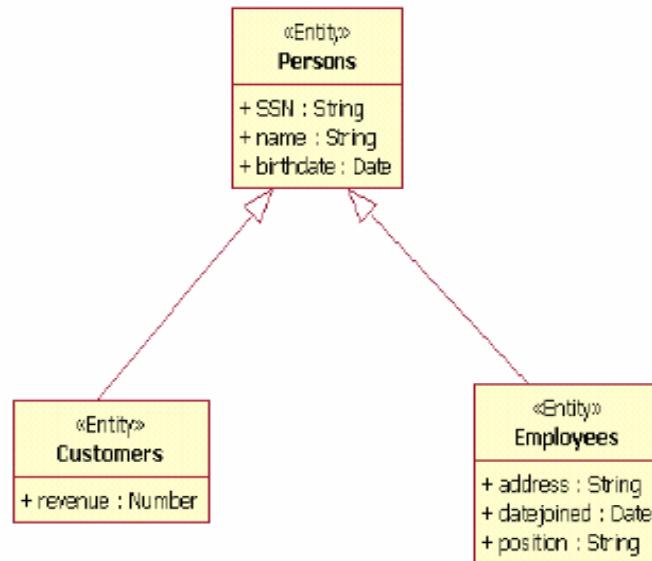
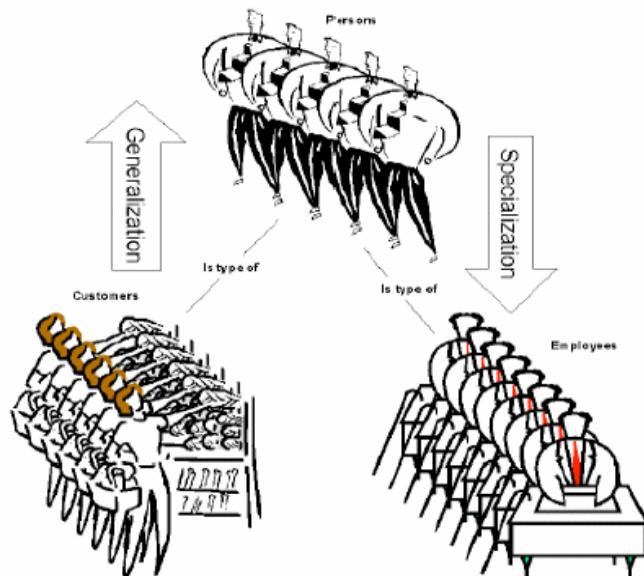
- ♦ Ovisnost: predstavlja situaciju u kojoj pojedini entitet ne može postojati bez veze s nekim drugim entitetom. UML razlikuje dvije vrste ovisnosti između entiteta: Agregacija i Kompozicija



Entitet Vjenčani list ovisi o dvije osobe!

Jednostavna ograničenja u ER oblikovanju

- Specijalizacija: predstavlja specifični segment nekog šireg pojma (entiteta)
- Generalizacija: suprotno od specijalizacije

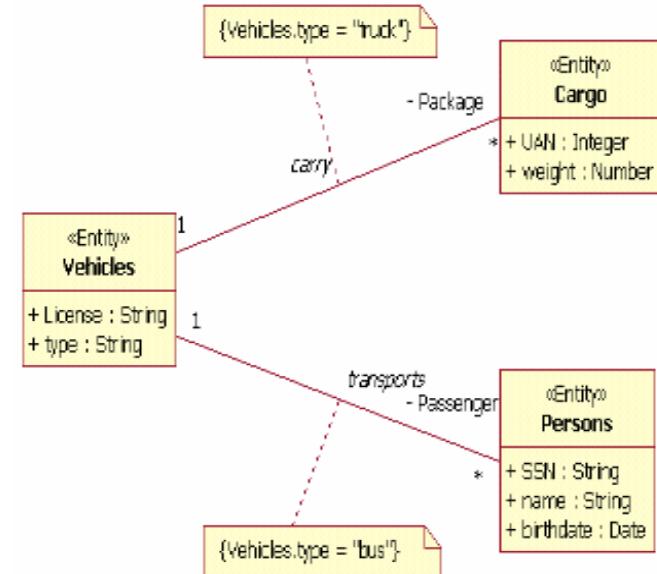
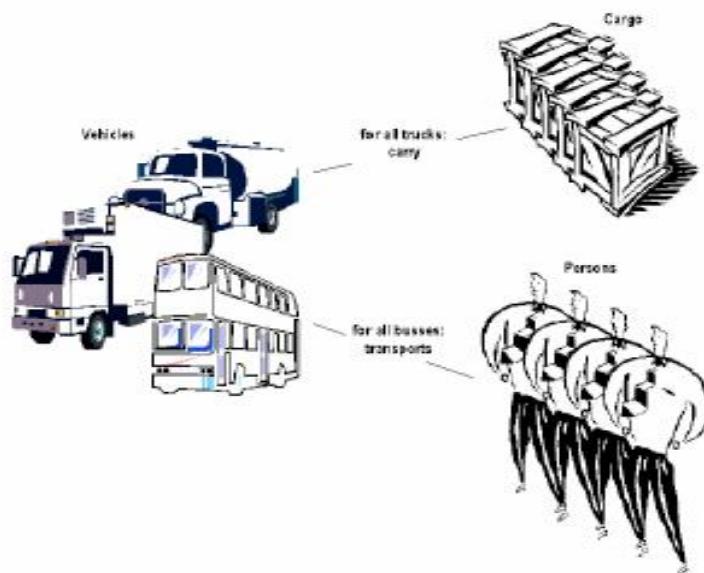


Za **klijente** i **zaposlenike** generalno možemo reći da su **osobe** (obrnuto vrijedi u nekim specifičnim slučajevima).

Entiteti **Klijenti** i **Zaposlenici** nasljeđuju attribute entiteta **Osobe**.

Jednostavna ograničenja u ER oblikovanju

- ♦ Kategorizacija: dok se specijalizacija odnosi na entitete, kategorizacija definira ograničenja na vezama između entiteta



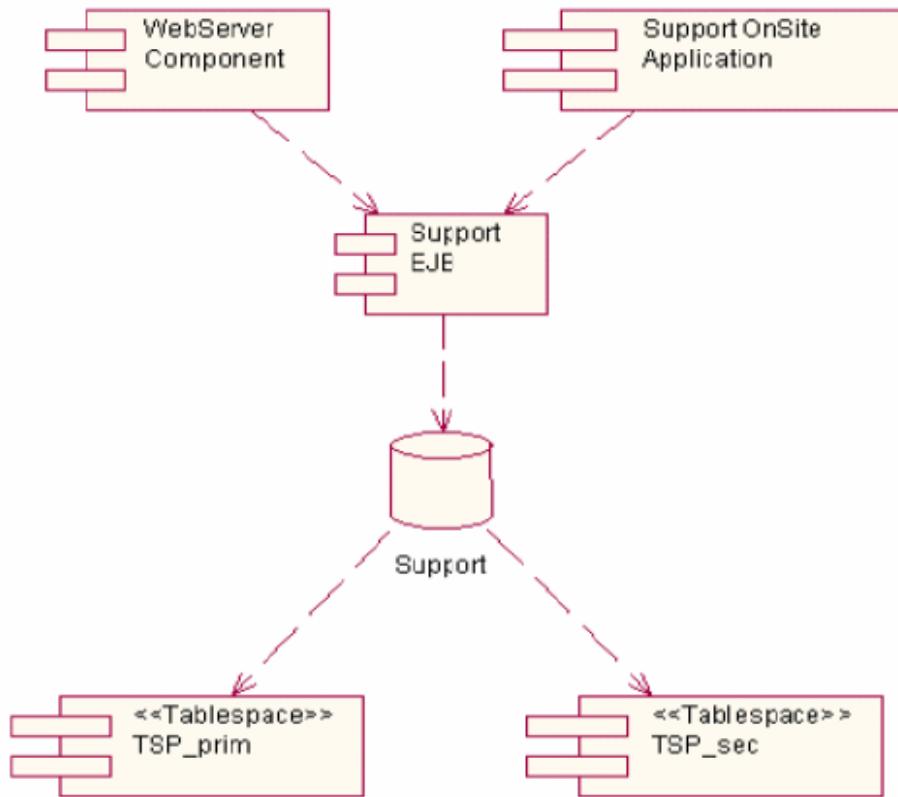
Entitet **Vozila** ima definiranu vezu entitetima **Teret** i **Osobe**.

Definiraju se kriteriji za vrstu transporta.

Fizička implementacija baze podataka prikazana UML dijagramima

- ◆ Fizički dijagrami:
 - Dijagrami isporuke(engl. deployment): fizičke veze između softvera i harvera
 - Dijagrami komponenti (engl. component): opisuje komponente sustava i njihove međuvisnosti

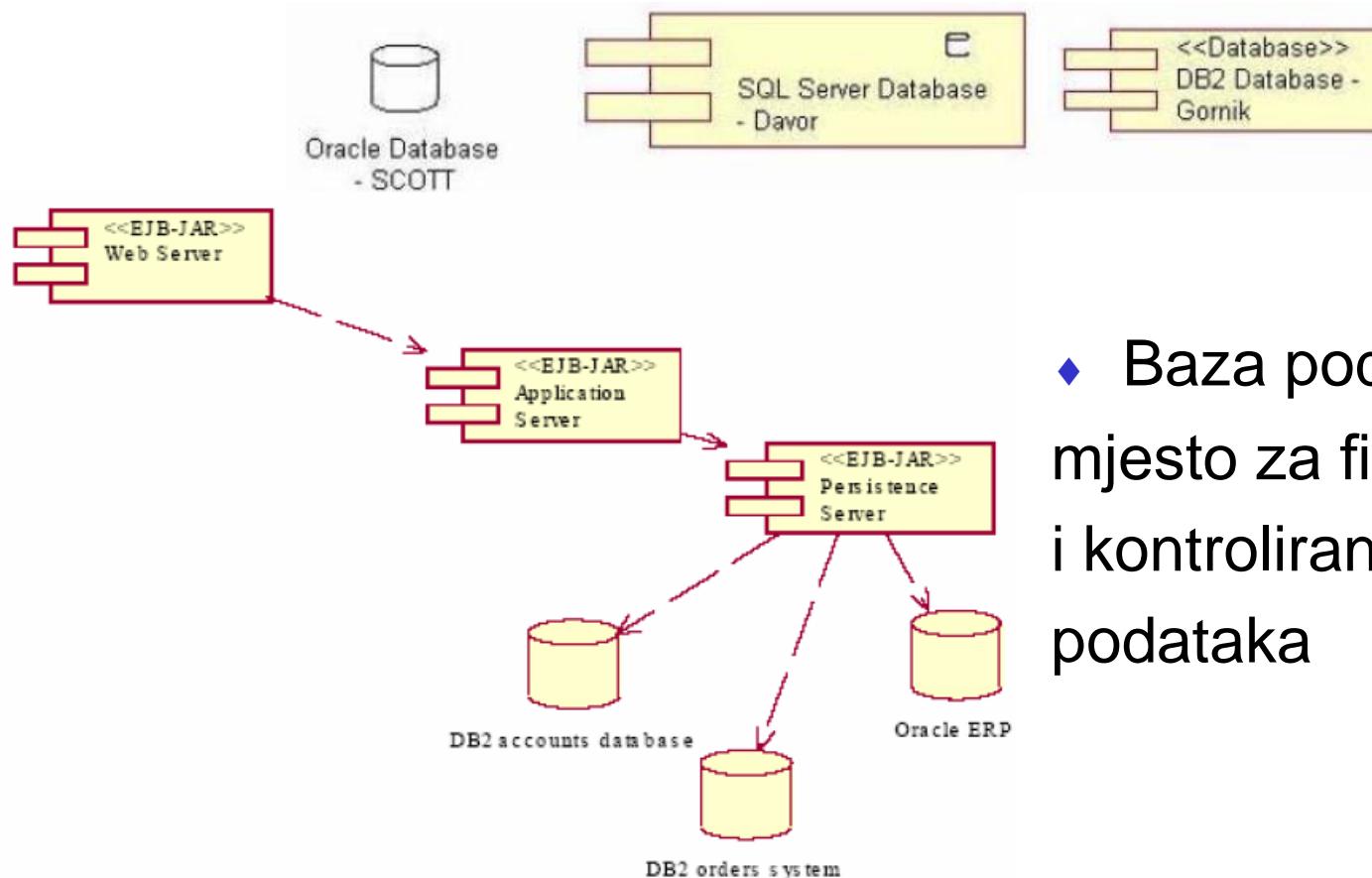
Fizička implementacija baze podataka prikazana UML dijagramima



- ◆ Tablespace - predstavlja spremište za podatke

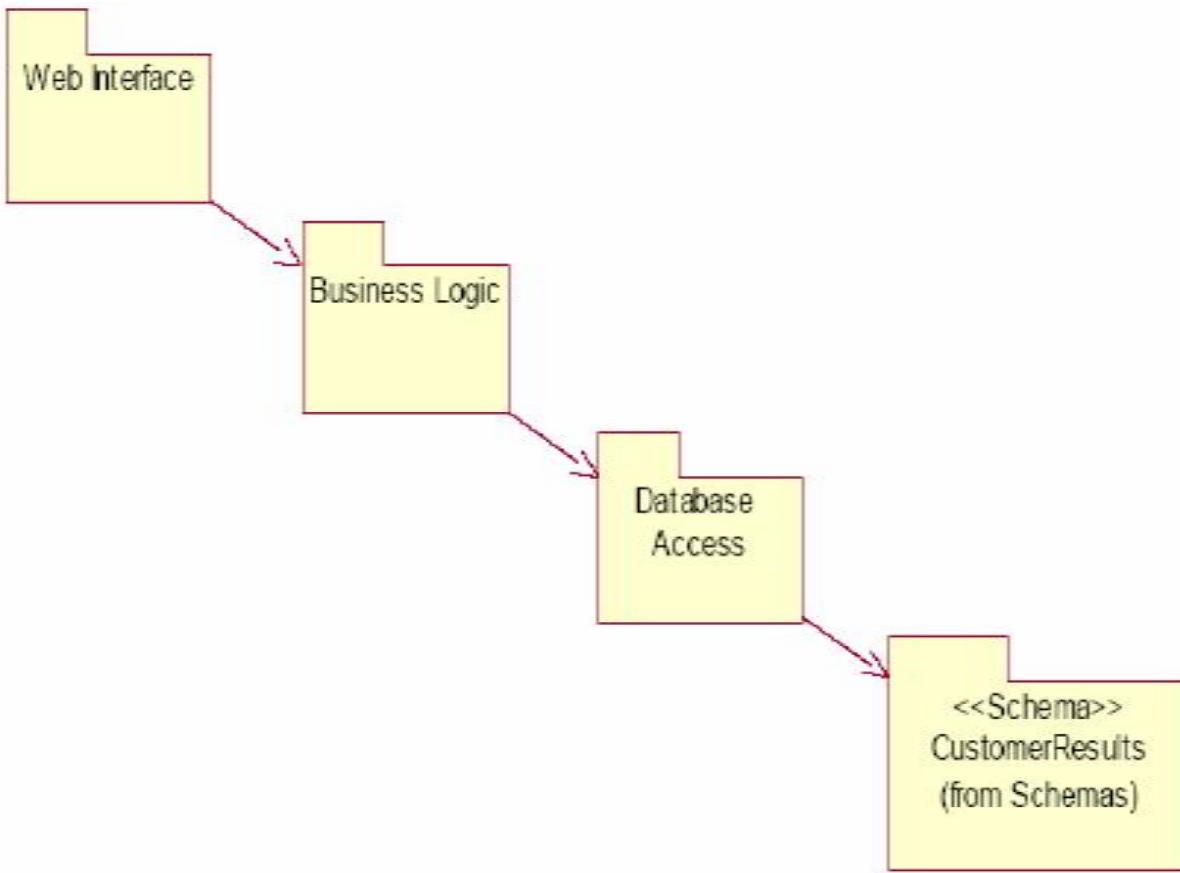
Fizički prikaz baze podataka

Dijagram komponenti



- ◆ Baza podataka - mjesto za fizičku pohranu i kontrolirani dohvati podataka

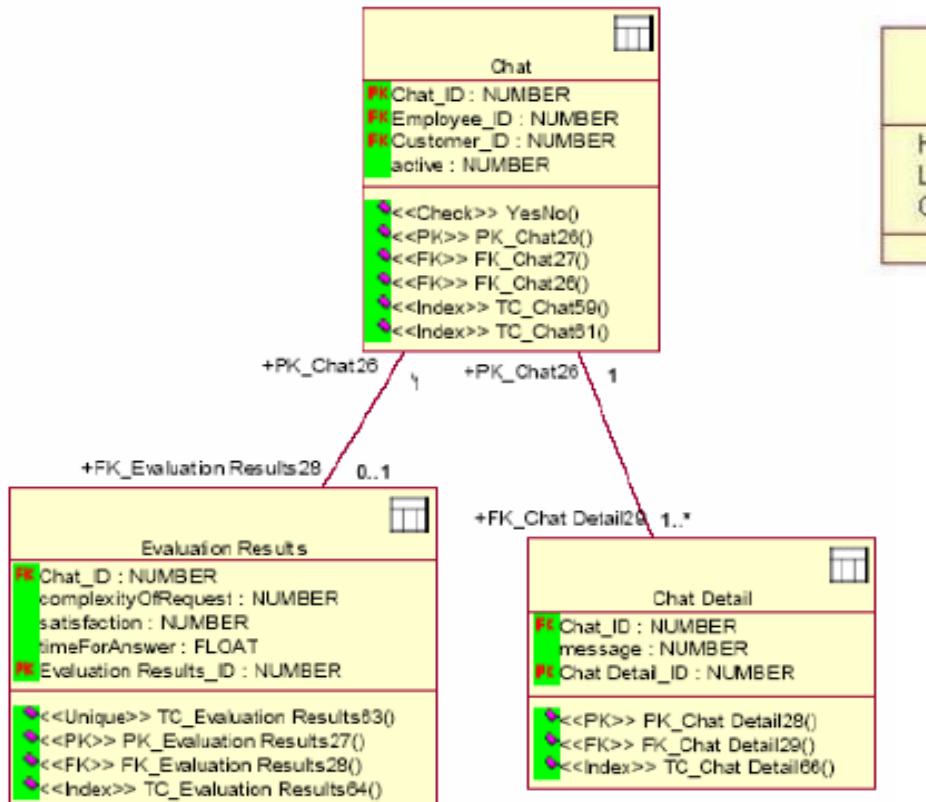
Baza podataka u dijagramu komponenti



◆ Shema -
Osnovna jedinica za organizaciju tablica u bazi

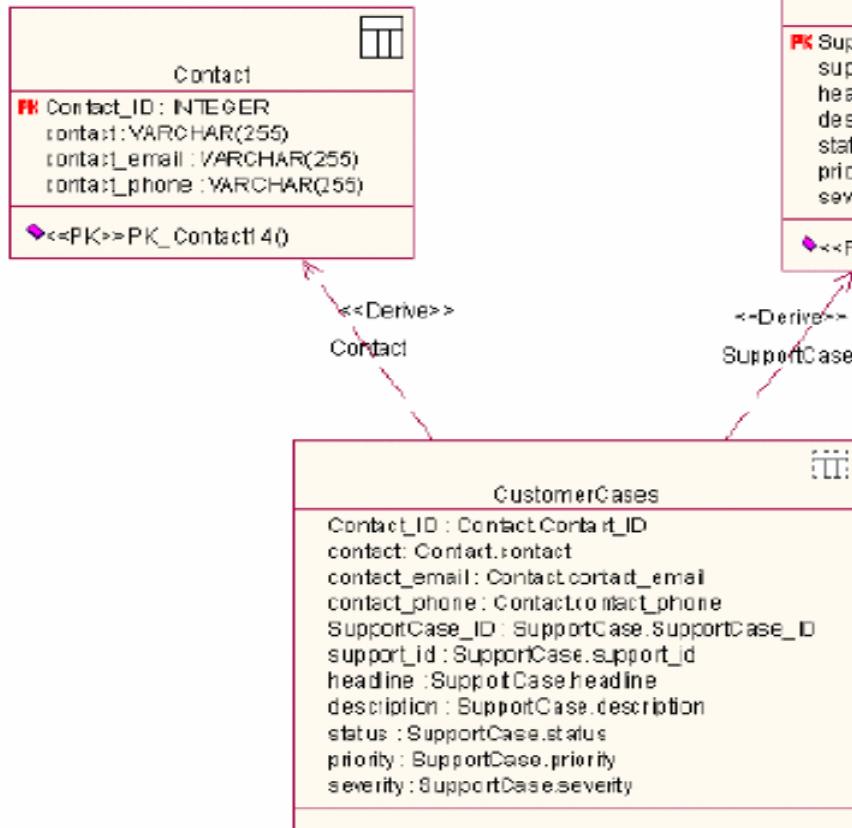
Prikaz shema u dijagramu klasa

Dijagram modela podataka



- ◆ Tablica -
- Osnovna jedinica za oblikovanje podataka u relacijskoj bazi
- ◆ Sastoji se od skupa redaka slične strukture

Pogled na tablice i veze predstavljen dijagramom modela podataka.



- ◆ Pogled (engl. View) - predstavlja virtualnu tablicu
- ◆ Fizički podaci se nalaze u različitim tablicama

Pogled – nastao od dvije tablice.

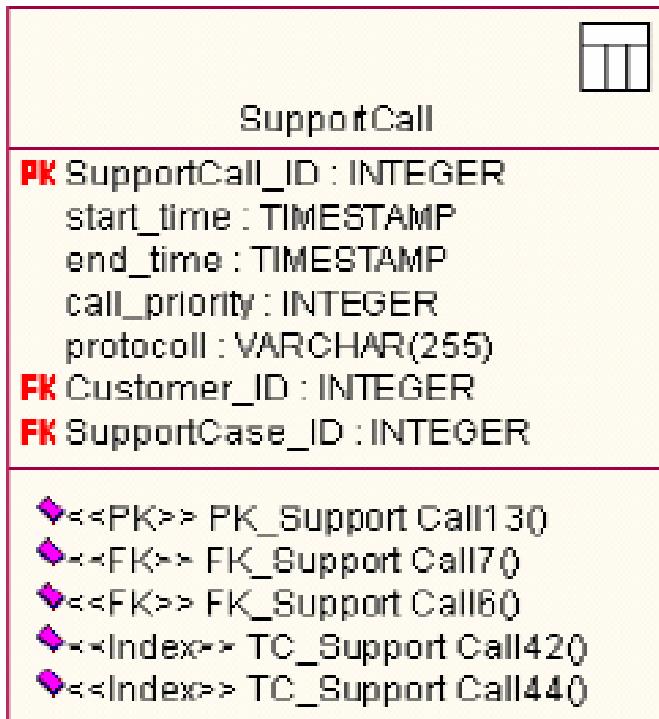
Fizička implementacija baze podataka prikazana UML dijagramima (Stupac)



- Stupac (engl. *Column*) - osnovni organizacijski element unutar relacijske baze podataka
- Svi podaci moraju biti unutar određenog stupca i određenog retka

Primjeri tablica sa stupcima.

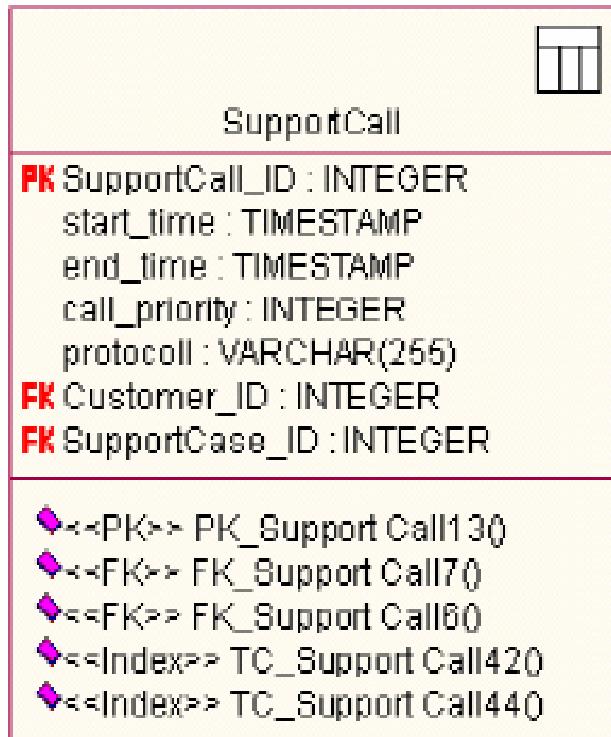
Fizička implementacija baze podataka prikazana UML dijagramima (Ključ)



- ◆ Ključ (engl. Key) - pomoću ključa pristupamo određenoj tablici
- ◆ Primarni ključevi jednoznačno identificiraju redak u tablici
- ◆ Strani ključevi se koriste za pristupanje podacima u drugim tablicama

Tablica s primarnim i stranim ključevima.

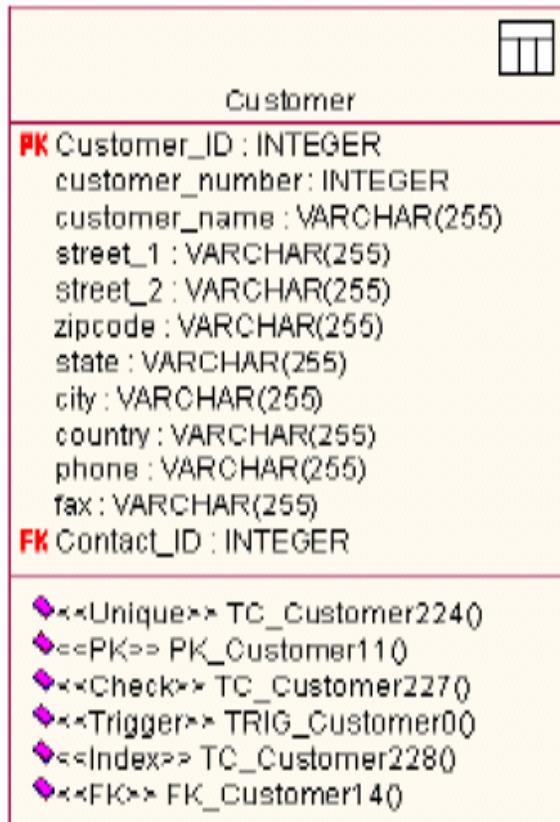
Fizička implementacija baze podataka prikazana UML dijagramima (Indeks)



- ♦ Indeks (engl. Index) - fizički model podataka
- ♦ Koristi se za brži pristup podacima

Tablica s dva indeksa.

Fizička implementacija baze podataka prikazana UML dijagramima (Ograničenje)

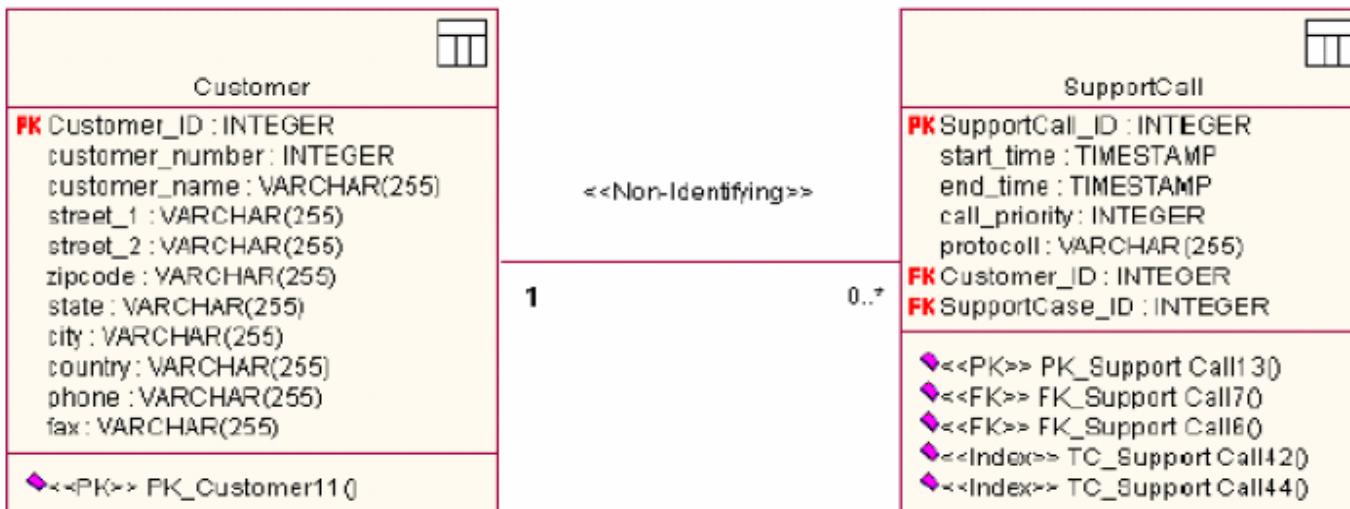


- ◆ Ograničenje (engl. Constraint) - predstavlja pravilo na strukturu baze podataka
- ◆ Ograničenje se primjenjuje na tablicu ili stupac

Tablica s definiranim ograničenjima.

Fizička implementacija baze podataka prikazana UML dijagramima (Veze)

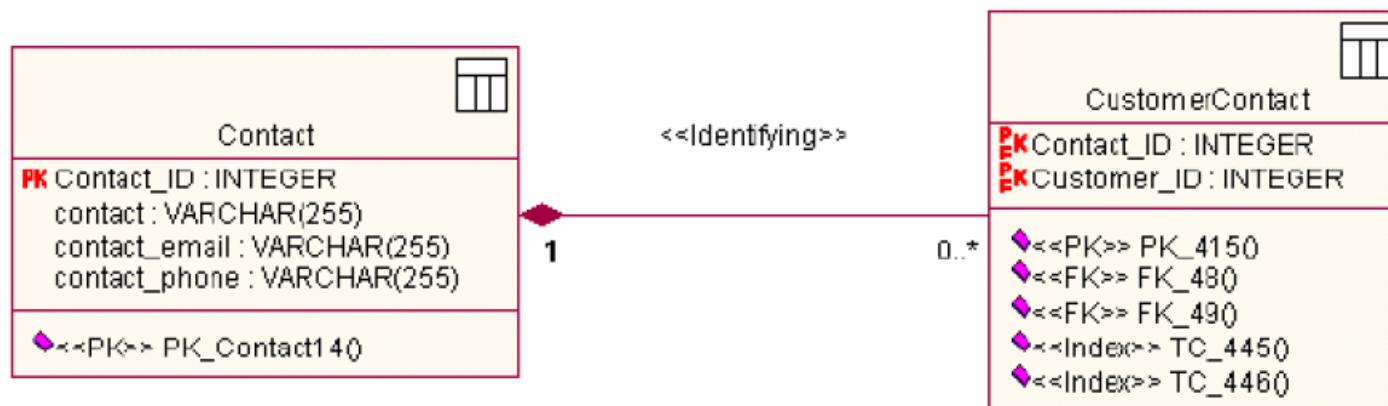
- ◆ Veze (engl. Relationship) - ovisnost bilo kakve vrste između tablica
- ◆ Na slici se prikazuje veza između dva identifikacijski jaka entiteta



Tablica s definiranim ograničenjima.

Fizička implementacija baze podataka prikazana UML dijagramima (Slabi entiteti)

- Na slici se prikazuje veza između identifikacijski jakog i identifikacijski i egzistencijalno slabog entiteta.



Tablica s definiranim ograničenjima.



Diplomski studij

Informacijska i
komunikacijska tehnologija:

Telekomunikacije i informatika

Upravljanje podacima

**Oblikovanje baze podataka
Faze u postupku oblikovanja
Objektni model**

Ak.g. 2015./2016.

OBLIKOVANJE BAZE PODATAKA

KOMENTAR

Danas sve više dolaze do izražaja postupci oblikovanja informacijskog sustava koji su objektno orijentirani. Oni u proces podatkovnog oblikovanja integriraju i procesnu komponentu (oblikovanje procesa). Primjer takvog postupka je korištenje **UML-a**.

Objektno-orientirani modeli podataka objedinjuju **podatkovnu i procesnu strukturu** i osiguravaju brže izvršavanje odgovarajućih operacija.

Objekt, kao osnovni koncept objektno-orientiranih modela podataka **sadrži**, osim opisa **podatkovne strukture** sličnog entitetu, i opis **načina rukovanja** (metode posluživanja).

MODELI PODATAKA

OBJEKTNO-ORIJENTIRAN MODEL PODATAKA



OBJEKT je apstrakcija nečega u problemskoj domeni, o čemu se prikupljaju podaci i što skriva (sadrži) vrijednosti svojih atributa (obilježja) i svojeg ponašanja.

KLASA je opis jednog ili više objekata koji imaju isti skup atributa i jednak opis ponašanja.

OBJEKT kao osnovni **koncept modela podataka** ima sljedeća **svojstva**:

- a) Objekt je **jedinstvena posebnost** u prostoru i vremenu i može se jedinstveno prepoznati
- b) Objekti imaju **svojstva**, tj. **atribute**
- c) Objektima se **rukuje** pomoći **metoda posluživanja**

OBJEKT kao osnovni **koncept modela podataka** ima sljedeća **svojstva (nastavak)**:

- d) Obilježje objekta je njegovo **ponašanje** i **stanje** koje se **mijenja**
- e) Objekti se mogu **klasificirati**
- f) Objekti mogu biti **sastavljeni**
- g) Objekti **razmjenjuju poruke**

MODELI PODATAKA

OBJEKTNO-ORIJENTIRAN MODEL PODATAKA

Grafički prikaz svojstva objekata dan je u odgovarajućim ***OBJEKTNIM DIJAGRAMIMA.***

Najpoznatija je notacija **Yourdona i Coada.**

Opis i definiranje navedenih svojstava objekata prikazati će se na primjerima dijagrama koji koriste navedenu notaciju.

MODELI PODATAKA

OBJEKTNO-ORIJENTIRAN MODEL PODATAKA



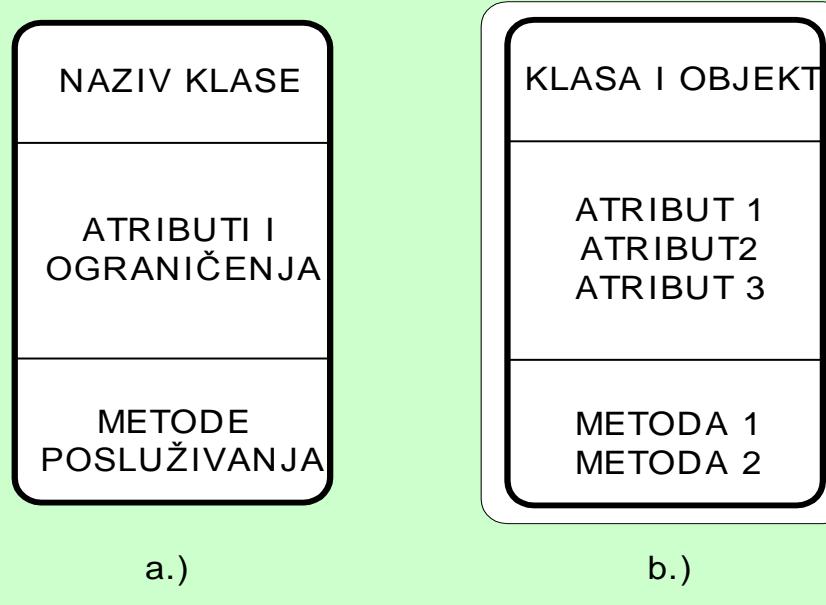
Cilj nam je da damo samo okvirni pregled karakteristika objektnog modela i da ukažemo na bitne razlike između modela koji koriste entitet kao osnovni koncept i objektno-orientiranih modela.

KLASA je opis jednog ili više objekata koji imaju **isti skup atributa** i jednak **opis ponašanja**, a **objekt** je *pojava klase*.

Klasa odgovara **tipu entiteta** u modelu entiteti-veze, osim što dodatno sadrži i **opis ponašanja** objekata klase. Grafički je klasa predstavljena otisnutim (debljim) pravokutnikom sa zaobljenim vrhovima, a njezini objekti tankim pravokutnikom. Svaki **objekt** sastoji se od **identifikatora** (*svojstvo a*), **popisa atributa** (*svojstvo b*) i **metoda posluživanja** (*svojstvo c*).

MODELI PODATAKA

OBJEKTNO-ORIJENTIRAN MODEL PODATAKA



Prikaz klase objekta (a) i klase-i-objekta)

MODELI PODATAKA

OBJEKTNO-ORIJENTIRAN MODEL PODATAKA



Među pojedinim objektima postoje **veze** kao i među pojavama entiteta. Određivanje veza između objekta te **kardinalnosti** tih veza, jednako je kao i u modelu entiteti-veze. Najčešće se u grafičkom prikazu koristi označavanje kardinalnosti slično **Martinovom** prikazu u **MEV**-u.

Neki autori umjesto takvog označavanja, koriste označavanje slično **Chenovom** prikazu u **MEV**-u, dakle s brojkama **0, 1, 2** ili općim brojem **M**. Na ovaj način se može preciznije izraziti kardinalnost pridruživanja.

MODELI PODATAKA

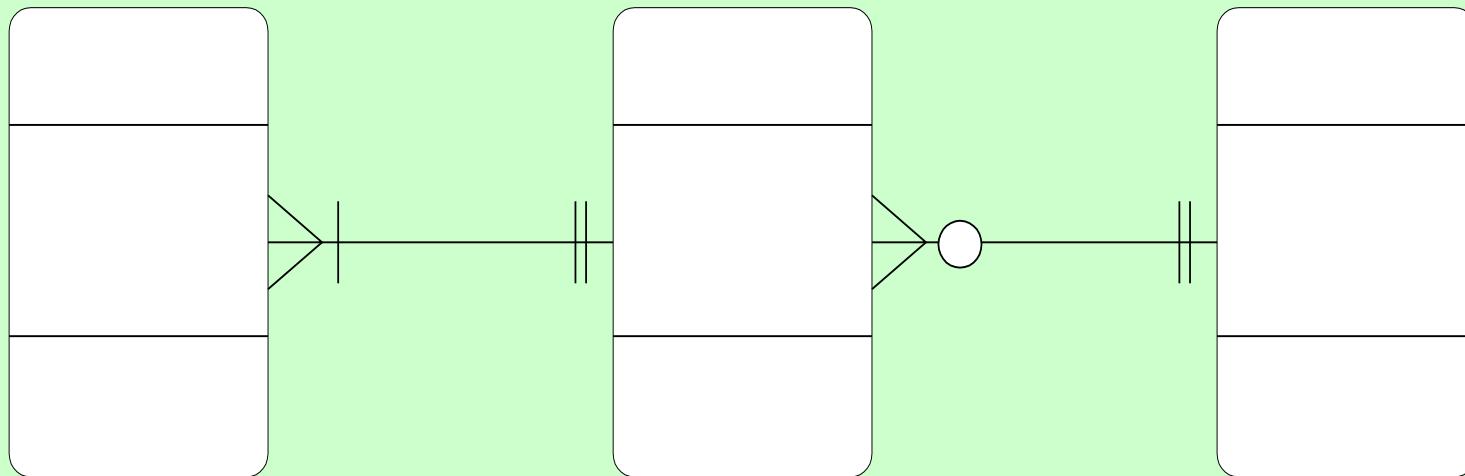
OBJEKTNO-ORIJENTIRAN MODEL PODATAKA



*Mjesto uz koje se oznaka **kardinalnosti** piše je obrnuto u odnosu na MEV* (kardinalnost pridruživanja dotičnog objekta piše se uz taj objekt).

MODELI PODATAKA

OBJEKTNO-ORIJENTIRAN MODEL PODATAKA



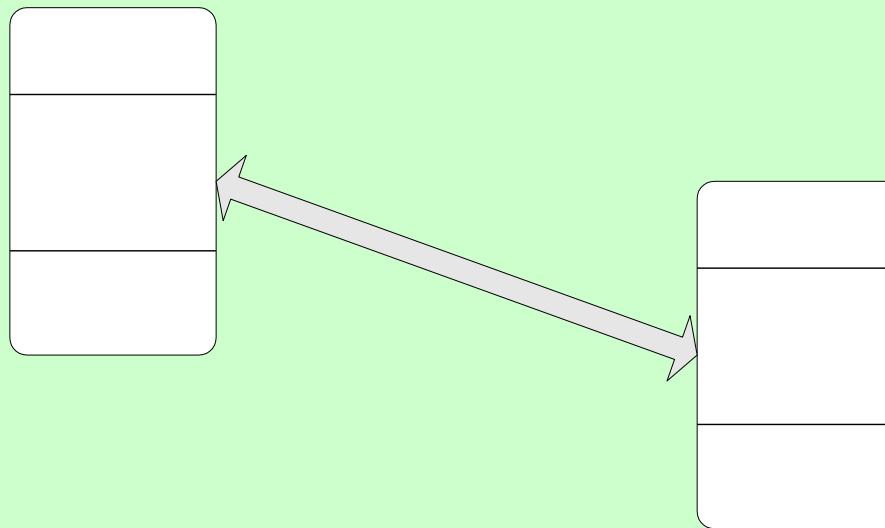
Veze između tipova objekata

Komunikacija između objekata slanjem i primanjem poruka (*svojstvo g*) zapravo se obavlja tako da se **primanjem poruke** aktivira neka od **metoda posluživanja** objekta koji je primio poruku ili se izvršava neka **procedura posluživanja** sastavljena od metoda posluživanja objekta. Nakon toga objekt može odgovoriti objektu koji je poslao poruku.

Poruke su koncepti analogni tokovima podataka ili dijagramima prijelaza stanja.

MODEL PODATAKA

OBJEKTNO-ORIJENTIRAN MODEL PODATAKA



Spojnica poruka (notacija Yourdona i Coada)

Koncept **poruka** (*svojstvo g*) je bitno za ostvarivanje *svojstava c* (**rukovanje** objektima pomoću **metoda posluživanja**) i **d** (**ponašanje** objekta i **stanje** koje se mijenja).

Vrijednosti atributa objekata mogu se promijeniti ili nadzirati isključivo pomoću operacija pridruženih atributima. Ove operacije nazivaju se **metodama posluživanja objekata**.

Atributi objekata i metode posluživanja nedjeljiva su cjelina i **“ugrađeno”** svojstvo objekta.

Stanje objekta je **prepoznatljivo i stabilno ponašanje** objekta u određenom vremenskom razdoblju. Stanje objekta se **mijenja**.

MODEL PODATAKA

OBJEKTNO-ORIJENTIRAN MODEL PODATAKA



Ponašanje i stanje objekta su povezani pojmovi jer ponašanje objekta, kao odziv na poticaj, ovisi o stanju objekta u trenutku kad se poticaj pojavio.

Metode posluživanja su *mehanizmi djelovanja na objekte.*

Njima se pokreće izvršavanje određene procedure ili akcije od strane objekta te promjena stanja objekta. Ponašanje objekata je bitan pokazatelj dinamike sustava i u modelima objekata se posebno naglašava.

U raspoznavanju strukture objekata koriste se postupci apstrakcije:

- **generalizacija/specijalizacija** (kraće gen-spec),
- **agregacija.**

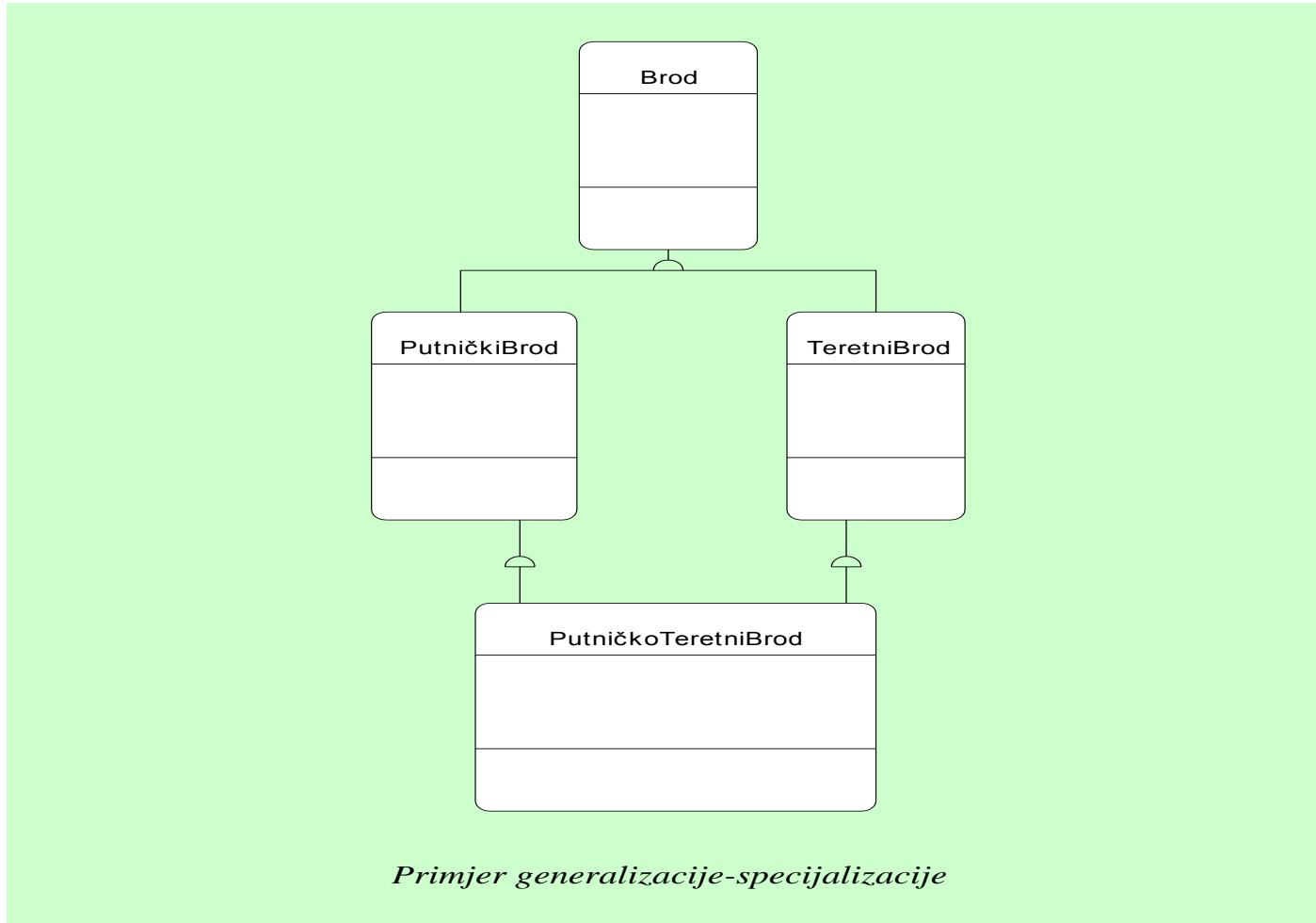
Oni omogućuju **klasifikaciju** (*svojstvo e*) i **povezivanje** objekata (*svojstvo f*).

Kod **gen-spec** se nadklasa crta gore, a podklasa dolje. Povezuju se linijom na kojoj **gen-spec** simbol ukazuje na odnos klasa.

Principom nasljeđivanja, podklasa nasljeđuje atributе svoje nadklase.

MODEL PODATAKA

OBJEKTNO-ORIJENTIRAN MODEL PODATAKA

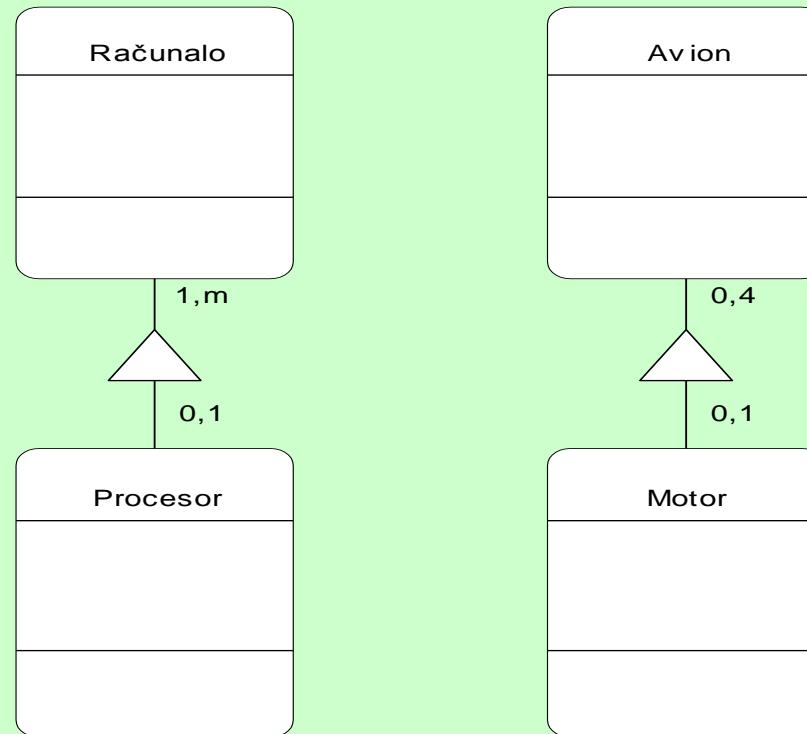


Agregacija je *sastav objekata* između kojih vlada **odnos cjeline i dijelova**. Cjelina-dio opisuje cjelinu objekta i njegove dijelove, koji su također objekti.

Objekt se kod prikaza obično crta gore, a njegovi dijelovi dolje. Povezuju se linijom na kojoj simbol cjelina-dio ukazuje na odnos objekata. Uz linije se označava broj koji označava kardinalnost odnosno broj dijelova u cjelini.

MODELI PODATAKA

OBJEKTNO-ORIJENTIRAN MODEL PODATAKA



Primjer agregacije (cjelina-dio)

Osim metode apstrakcije i agregacije, u postupku modeliranja objekata koriste još dvije opće metode modeliranja podataka:

- **podatkovno apstrahiranje,**
- **proceduralno apstrahiranje.**

Podatkovno apstrahiranje je metoda zanemarivanja metoda posluživanja objekta, koje nisu od interesa za određeni atribut nekog objekta, ili su vezane uz druge attribute tog objekta.

Proceduralno apstrahiranje je osnova funkcionalnog rasčlanjivanja i opisivanja unutar logike procesa. Prema ovoj metodi, složena se funkcija rasčlanjuje na skup podfunkcija.

Svaka metoda posluživanja koja ostvaruje uočljiv učinak može se smatrati jednom posebnošću. Ovo je najvažnija metoda modeliranja procesa, a koristi se i kod modeliranja objekata.

Posebne metode modeliranja objekata su i:

- **metoda učahurivanja,**
- **metoda nasljeđivanja.**

Metoda učahurivanja je ugrađivanje skupa atributa i metoda posluživanja u objekt. Objekt učahuje attribute i metode posluživanja te se prema okolici ponaša kao crna kutija.

Način na koji se metode posluživanja izvode za okolicu nije bitan. **Učahurivanje** predstavlja **potiskivanje informacija** o *metodama* i *atributima*, odnosno načinu njihova **izvođenja**.

Nasljeđivanje je svojstvo podklase da nasljeđuje atribute i metode posluživanja od nadređenog objekta. Sve podklase nasljeđuju sve atribute i metode posluživanja nadklase.

Zahvaljujući nasljeđivanju pojednostavljuje se specifikacija podklasa, čime se ubrzava razvoj informacijskog sustava te povećava kvaliteta specifikacije. Svaka podklasa osim naslijeđenih može definirati i vlastite atribute i metode posluživanja.

KOMENTAR O OBJEKTNO-ORIJENTIRANOM MODELU PODATAKA:

- Bitna je razlika između objektnih modela i modela entiteti-veze u definiciji metoda posluživanja (servisa).
- Model entiteti-veze ne predstavlja konceptualni opis kompletног informacijskog sustava jer opisuje samo podatkovnu komponentu informacijskog sustava.
- Objektni model je semantički bogatiji jer opisuje i procesnu komponentu informacijskog sustava pa predstavlja konceptualni opis kompletног sustava.

KOMENTAR O OBJEKTNO-ORIJENTIRANOM MODELU PODATAKA (nastavak):

- Pojava objektno orijentiranih modela i njihova implementacija mogla bi osigurati efikasan rad u stvarnom vremenu.
- Ovi modeli zahtjevaju i **objektno-orijentirane baze podataka** podržane odgovarajućim **sustavima za upravljanje objektno-orijentiranom bazom podataka (OODBMS)**.

KOMENTAR O OBJEKTNO-ORIJENTIRANOM MODELU PODATAKA (nastavak):

- **OODBMS** bi morao biti u mogućnosti pohranjivati objekte na objektni način i isto tako ih pretraživati i dohvaćati. Gledano sa stanovišta objektno-orijentiranih programske jezike, u bazu podataka bi se pohranjivali *objekti s pridijeljenim podacima i programskim kodom za izvršavanje odgovarajućih metoda posluživanja*. Na taj način bi objekt bio nakon dohvata spreman da se nad njim izvrše odgovarajuće akcije.

KOMENTAR O OBJEKTNO-ORIJENTIRANOM MODELU PODATAKA (nastavak):

- Prednost objektno-orientiranog modela kao kompletног konceptualnog modela doћи ће do izražaja onda kad komercijalno budu dostupni i odgovarajući **OODBMS**-ovi.
- Danas se konceptualni objektni model najčešće implementira u objektno-relacijskom sustavu. To je i razumljivo jer na trжишту dominiraju objektno-relacijski **DBMS**-ovi (**ORDBMS**).

KOMENTAR O OBJEKTNO-ORIJENTIRANOM MODELU PODATAKA (nastavak):

- Podatkovna struktura objektnog modela, osim malih razlika u označavanju kardinalnosti i drugačijim grafičkim prikazima, veoma je slična modelu entiteti-veze. Podatkovna struktura konceptualnog objektnog modela se može transformirati na relacijski model podataka, a procesna se može realizirati odgovarajućim aplikacijama nad objektno-relacijskom bazom podataka.

OBLIKOVANJE BAZE PODATAKA

KOMENTAR

Za razliku od **MEV**-a koji razrađuje samo podatkovnu komponentu informacijskog sustava, objektni model je bogatiji jer opisuje i procesnu komponentu čitavog sustava pa predstavlja **potpun konceptualni opis čitavog sustava**.

OBLIKOVANJE BAZE PODATAKA

KOMENTAR

Nadovezujući se na rezultat analize objektnog sustava (informacije o strukturi podataka i informacije o obradama podataka) izrada konceptualnog i logičkog objektnog modela je vrlo slična izradi modela entiteti-veze.

OBЛИKOVANJE BAZE PODATAKA

KOMENTAR

Osnovni postupci ***KONCEPTUALNOG*** i ***LOGIČKOG MODELIRANJA OBJEKATA*** su:

1. ***ODREĐIVANJE OBJEKTNOG SUSTAVA*** – određuje se skup objekata bitnih za informacijski opis sustava koji se modelira, uz korištenje neformalnih specifikacija (pisani tekst) za opis sustava.

OBЛИKOVANJE BAZE PODATAKA

KOMENTAR

Osnovni postupci **KONCEPTUALNOG** i **LOGIČKOG MODELIRANJA OBJEKATA** su:

2. **ODREĐIVANJE TIPOVA OBJEKATA** (klasa) – prepoznaju se zajednički atributi, metode posluživanja objekata, stanje objekata i potreba za njihovim pamćenjem, te se utvrđuje specifikacija osnovnih zahtjeva korisnika i po potrebi proširuje osnovni skup objekata.

OBLIKOVANJE BAZE PODATAKA

KOMENTAR

Osnovni postupci **KONCEPTUALNOG** i **LOGIČKOG MODELIRANJA OBJEKATA** su:

3. **OBLIKOVANJE KLASIFIKACIJSKIH I SASTAVNIH STRUKTURA OBJEKATA.**
4. **ODREĐIVANJE ATRIBUTA** – atributi možemo podijeliti na opisne, izvedene i uvjetno izvodljive. Uvjetno izvodljivi atributi su oni koji se mogu odrediti samo pod određenim uvjetima (npr. ako su poznate vrijednosti drugih atributa).

OBЛИKOVANJE BAZE PODATAKA

KOMENTAR

Osnovni postupci **KONCEPTUALNOG** i **LOGIČKOG MODELIRANJA OBJEKATA** su:

5. **UTVRĐIVANJE VEZA I KARDINALNOSTI** - kao u modelu entiteti-veze.
6. **ODREĐIVANJE DOGAĐAJA** – osnovne tehnike su tehnika stanje-događaj-odziv, dijagram prelaza stanja i životni ciklus entiteta.

OBЛИKOVANJE BAZE PODATAKA

KOMENTAR

Osnovni postupci ***KONCEPTUALNOG*** i ***LOGIČKOG MODELIRANJA OBJEKATA*** su:

7. ***ODREĐIVANJE METODA POSLUŽIVANJA*** – ciljevi ovog procesa su pridruživanje osnovnih metoda posluživanja svakom objektu, definiranje procedura i akcija (koje čine složene metode posluživanja), učahuravanje metoda i provjera metoda za sve slučajeve (u kojima je objekt poslužitelj ili klijent).

OBЛИKOVANJE BAZE PODATAKA

KOMENTAR

Osnovni postupci **KONCEPTUALNOG** i **LOGIČKOG MODELIRANJA OBJEKATA** su:

8. **ODREĐIVANJE SPOJNICA PORUKA** – provjeravaju se veze objekata i određuju spojnice poruka objekata. Provjerava se da li za svaku spojnicu poruka postoje metode posluživanja.
9. **PROVJERA MODELA.**

OBLIKOVANJE BAZE PODATAKA

KOMENTAR



Implementacijsko i fizičko modeliranje objektnog modela ovisi o konkretnoj razvojnoj i radnoj okolini (jeziku i bazi podataka).

Bitne prednosti objektnog modela moći će se realizirati tek s komercijalno dostupnim **OODBMS**-ovima.



Diplomski studij

Informacijska i
komunikacijska tehnologija:

Telekomunikacije i
informatika

Upravljanje podacima

SEMANTIČKI WEB

sastavio: doc. dr. sc. Marko Banek

Ak.g. 2015./2016.

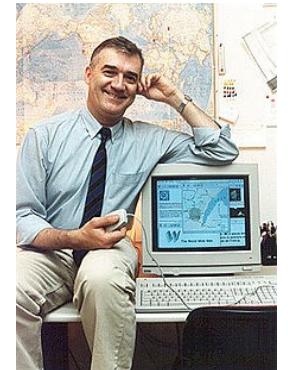
Povijest Semantičkog Weba

◆ World Wide Web

- tvorci Sir Tim Berners-Lee i Robert Cailliau (obojica tada zaposleni u CERN-u):
 - “Proposal for a HyperText Project” (studenog 1990.): *a "web" of "hypertext documents" to be viewed by "browsers" using a client-server architecture*
 - Berners-Lee: prva implementacija poslužitelja i preglednika u prosincu 1990.
- Berners-Lee 1994. odlazi na MIT raditi na projektima koje financira DARPA (Defense Advanced Research Projects Agency)
 - osniva W3C (World Wide Web Consortium)

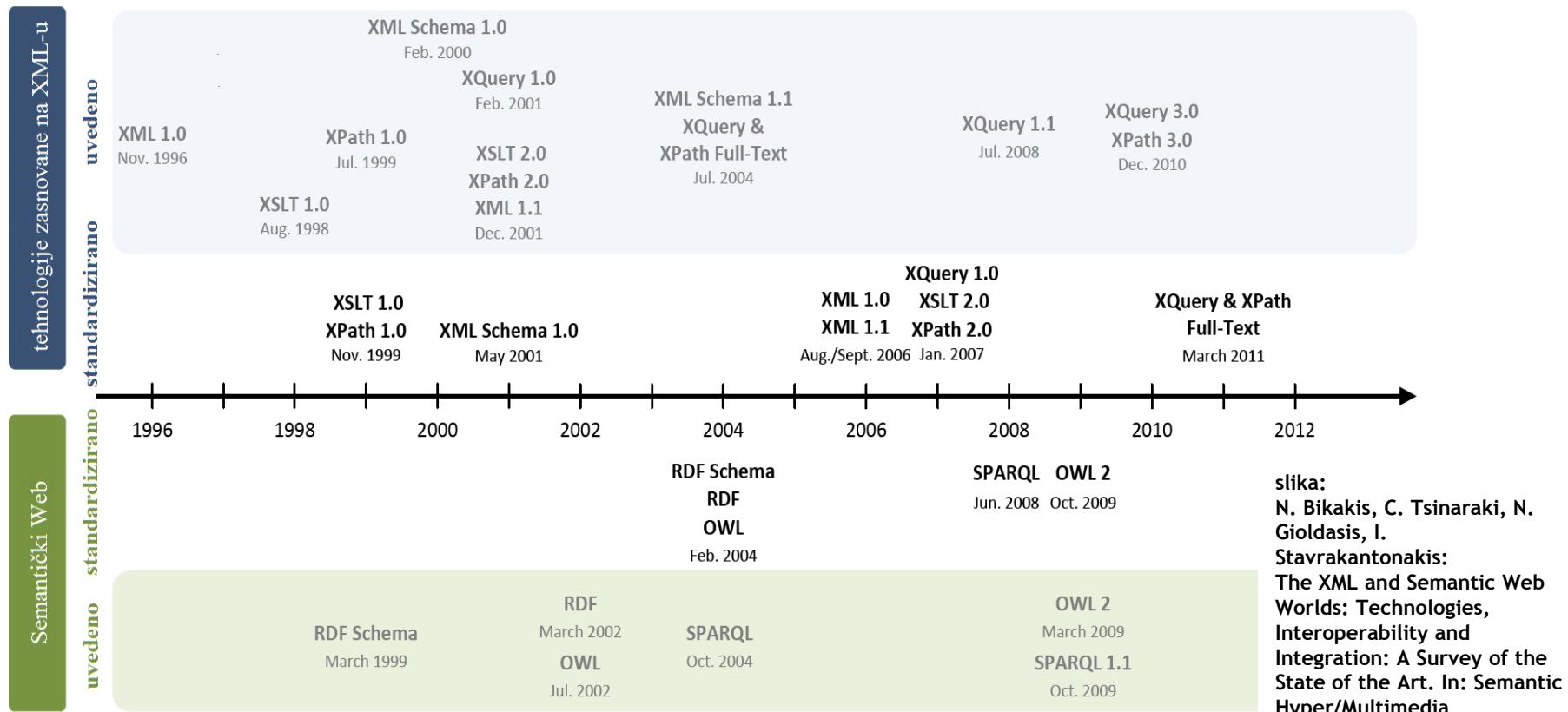


T. Berners-Lee,
2012; Wikipedia



R. Cailliau,
Wikipedia/CERN

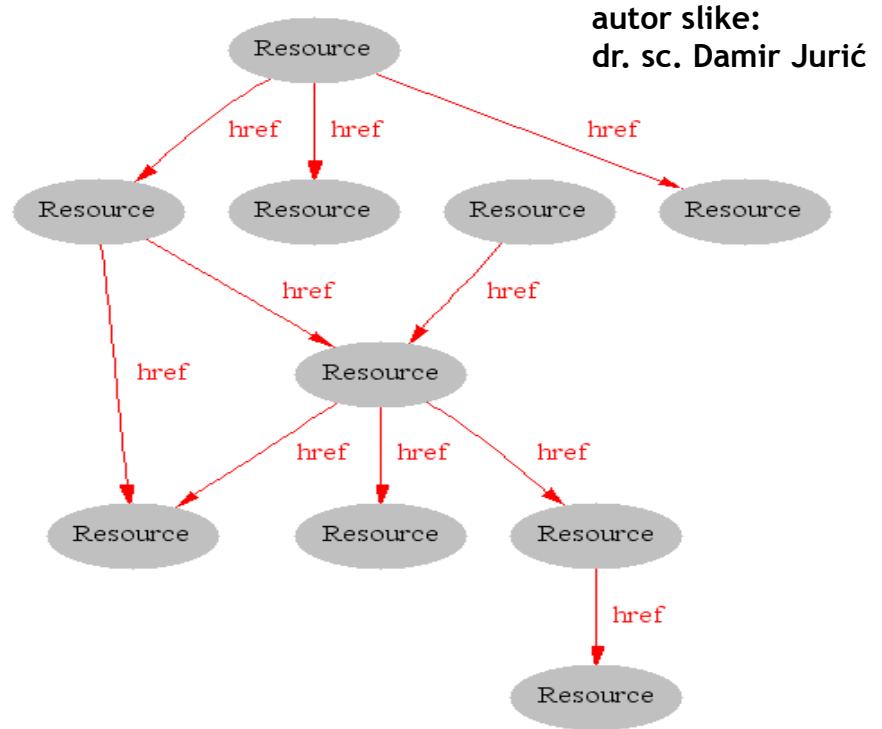
- ◆ konzorcij čiji je cilj razvijati i standardizirati tehnologije vezane uz World Wide Web
 - članovi: sveučilišta, neprofitne organizacije, komercijalne tvrtke



- ◆ najstariji: XML
 - prve verzije standarda 1996.,
 - konačna verzija XML 1.0 1998.
- ◆ Semantički Web
 - sam pojam se pojavio koncem 2000.
 - stvarno započeo razvojem RDF/RDF Schema (prve verzije standarda 1998.)
 - OWL, SPARQL

Osnovni način funkcioniranja Weba

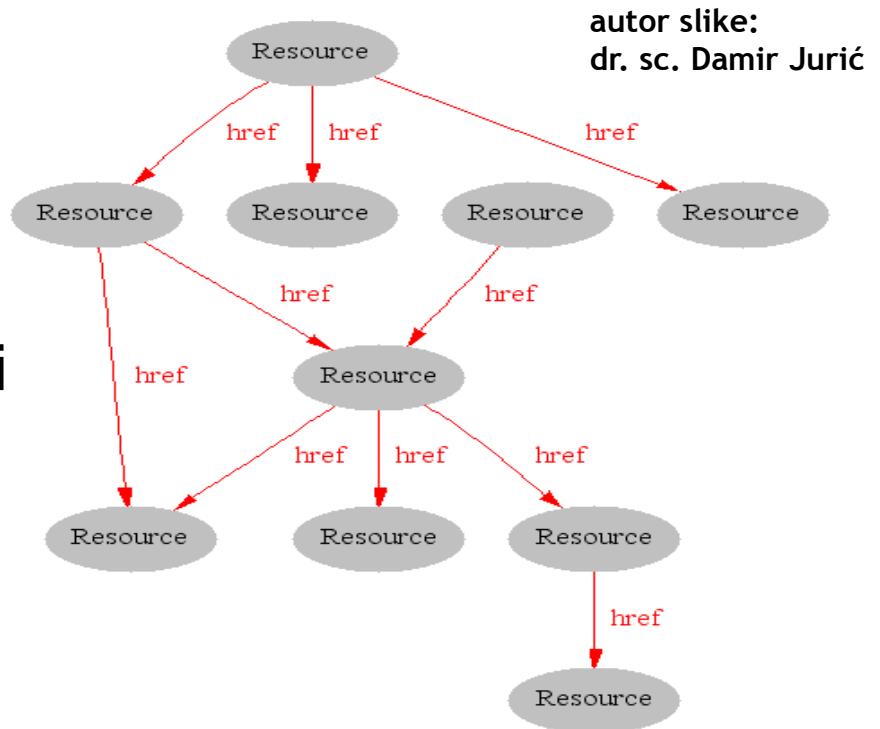
- ◆ nepostojanje konteksta
- ◆ niz sadržaja povezanih linkovima



RDF - uvod

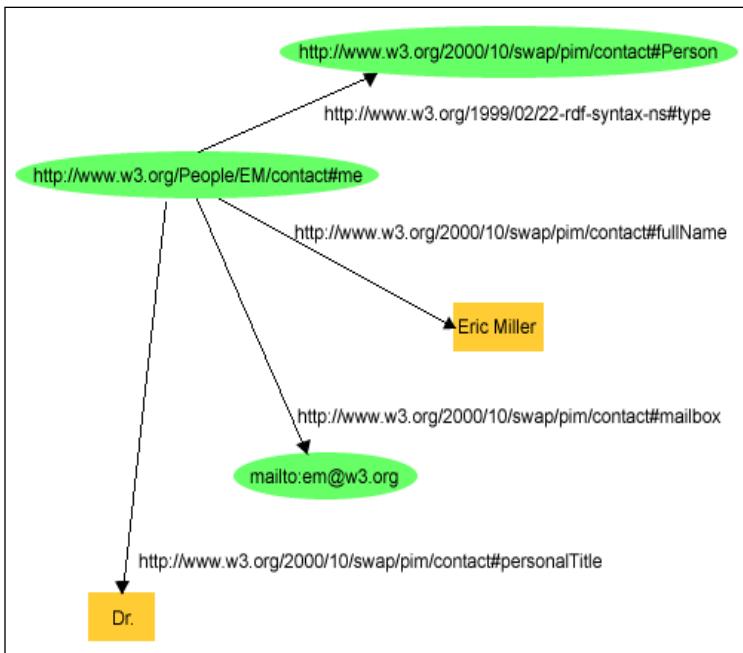
- ◆ nepostojanje konteksta
- ◆ niz sadržaja povezanih linkovima
- ◆ g. 1995. sustav bibliografskih oznaka Dublin Core (nevezano uz W3C)

1. Title
2. Creator
3. Subject
4. Description
5. Publisher
6. Contributor
7. Date
8. Type
9. Format
10. Identifier
11. Source
12. Language
13. Relation
14. Coverage
15. Rights



- ◆ RDF služi za *zapisivanje informacija o resursima na Webu*.
- ◆ u primarnom smislu zapisi u RDF-u su *metapodaci o resursima na Webu* kao što su njihov *naslov, autor, datum posljednje izmjene, informacije o autorskim pravima i licenciranju upotrebe sadržaja na stranici*
- ◆ u širem smislu, resurs na Webu može se proširiti na sve se može identificirati na Webu (tj. ima svoj URI, a ne nužno i dohvatiti.
 - time je otvorena mogućnost da se RDF-om zapravo zapisuju iskazi o realnom svijetu, pri čemu se pojedini element realnog svijeta prikazuje kao resurs Weba koji posjeduje identifikator (URI).

RDF – resursi na webu



slika preuzeta iz: RDF Primer. W3C Recommendation 10 February 2004.

- ♦ resurs čiji je URI
`http://www.w3.org/People/EM/contact#me`
(tj. resurs *me* u imeniku čiji je URI
`http://www.w3.org/People/EM/contact`)
- ♦ resurs *me* pripada klasi *Person* definiranoj u imeniku
`http://www.w3.org/2000/10/swap/pim/contact#Person`
(svojstvo *type*, kojim se izražava pripadnost nekoj klasi, predefinirano je RDF Schemom).
- ♦ vrijednost svojstva *fullName* za resurs *me* je konstanta *Eric Miller*
- ♦ vrijednost svojstva *personalTitle* za resurs *me* je konstanta *Dr.*

RDF: zapis u XML-u

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">

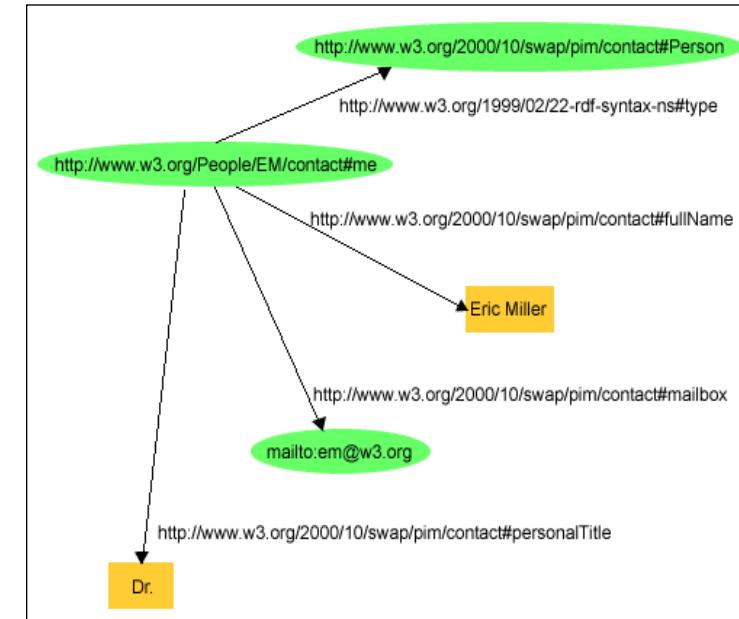
  <rdf:Description rdf:about="http://www.w3.org/People/EM/contact#me">
    <rdf:type rdf:resource="http://www.w3.org/2000/10/swap/pim/contact#Person"/>
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </rdf:Description>

</rdf:RDF>
```

Vizualno preglednija sintaksa Turtle

```
@prefix eric:
<http://www.w3.org/People/EM/contact#> .
@prefix contact:
<http://www.w3.org/2000/10/swap/pim/contact#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-
rdf-syntax-ns#> .
```

```
eric:me contact:fullName "Eric Miller" .
eric:me contact:mailbox <mailto:em@w3.org> .
eric:me contact:personalTitle "Dr." .
eric:me rdf:type contact:Person .
```



- ◆ u širem smislu, resurs na Webu može se proširiti na sve što se može identificirati na Webu (URI), a ne nužno i dohvatiti (pomoću URL-a).
- ◆ RDF-om se mogu zapisati izjavi o realnom svijetu, pri čemu se pojedini element realnog svijeta prikazuje kao resurs Weba koji posjeduje identifikator (URI).
- ◆ načelo subjekt-predikat-objekt.
 - subjekt i objekt su resursi
 - predikat je svojstvo kojim se povezuju subjekt i objekt (svojstvo također ima URI)
 - ovisno o svojstvu, objekt ne mora nužno biti resurs u užem smislu tj. resurs koji posjeduje URI, nego konstantna vrijednost u smislu broja ili niza znakova

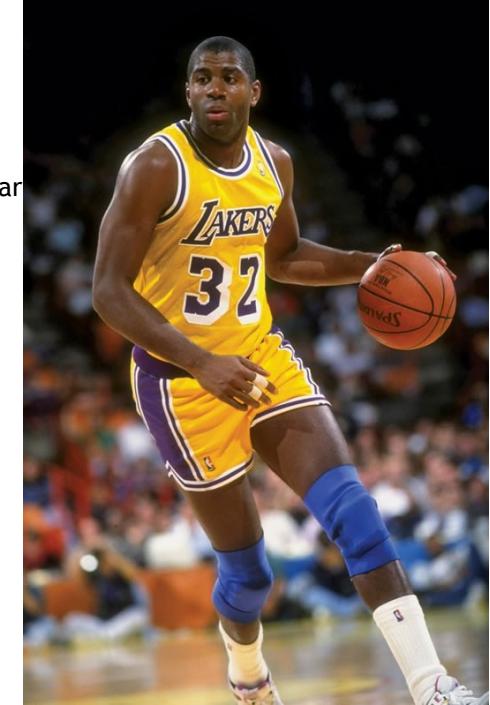
RDF: trojke

Depar

- ◆ Jesu li trojke dovoljne:

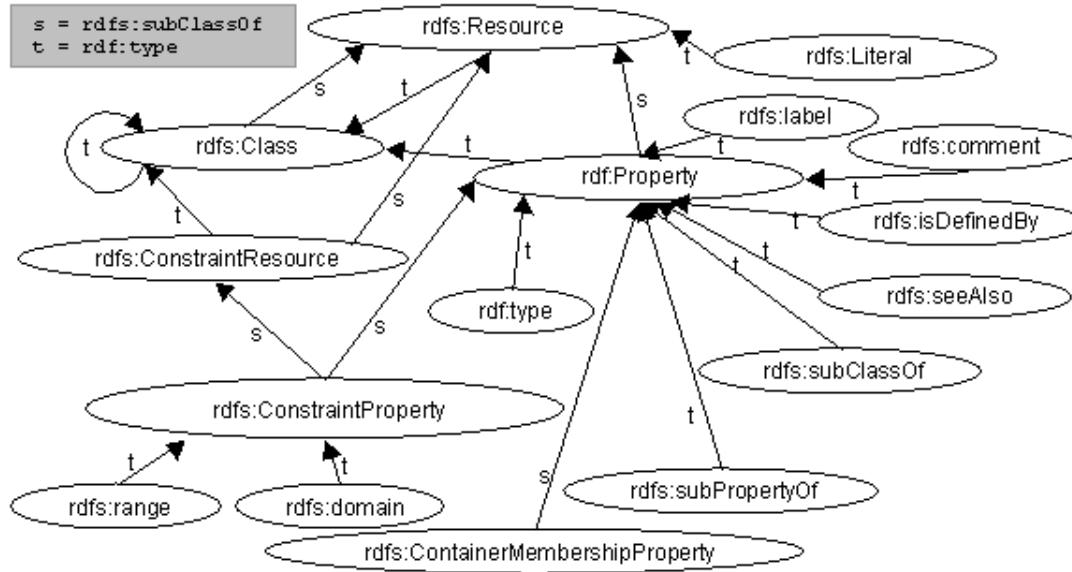
x:WayneGretzky y:igraZa z:LosAngeles
x:MagicJohnson y:igraZa z:LosAngeles

- ◆ očito, riječ je o klubovima iz Los Angelesa koji se ne natječu u istom sportu
- ◆ dodatno:
 - može li Magic imati 4 minute isključenja u utakmici; može li Gretzky imati 4 osobne pogreške



RDF Schema: metamodel za RDF

- ◆ class, property
 - objekti pripadaju klasama
- ◆ type
 - pridružuje instancu klasi
- ◆ subClassOf
 - pridružuje klasu natklasi
- ◆ subPropertyOf
 - pridružuje svojstvo nadsvojstvu
- ◆ domain, range
 - klasa subjekta svojstva (domena): *domain*;
 - klasa objekta svojstva (kodomena): *range*



- ◆ RDF i RDF Schema
 - moguće definirati taksonomije
- ◆ sada možemo zaključivati:

ako vrijedi:

```
<WayneGretzky,type,HockeyPlayer>
<HockeyPlayer,subClassOf,Sportsman>
<Sportsman,subClassOf,Person>
```

onda nužno vrijedi i:

```
<HockeyPlayer,subClassOf,Person>
<WayneGretzky,type,Sportsman>
<WayneGretzky,type,Person>
```

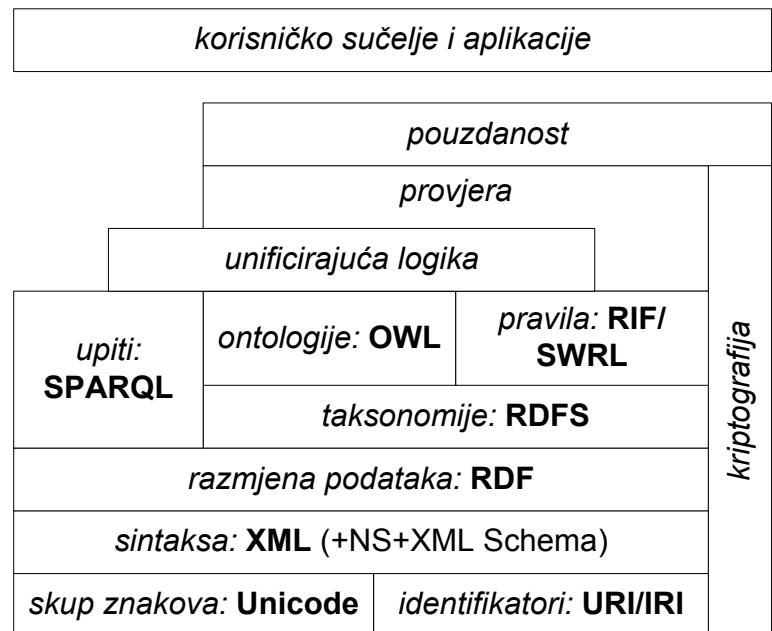
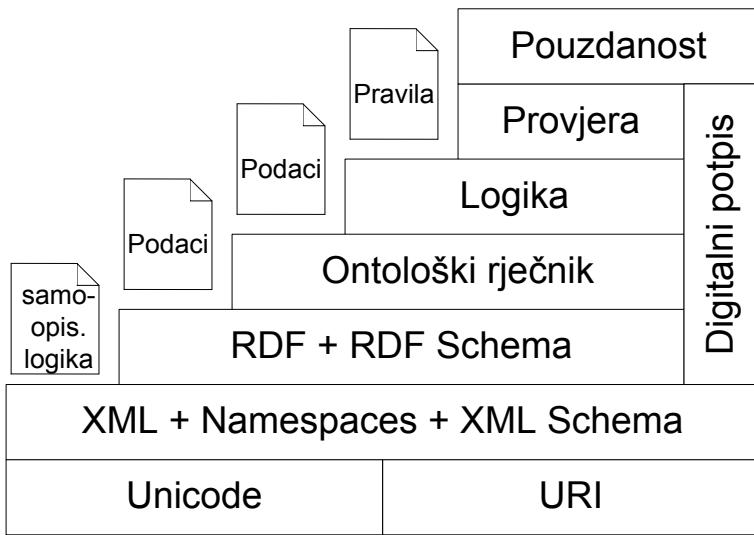
- ◆ Tim Berners-Lee:

- pojam prvi puta spomenut na predavanju u sklopu simpozija XML 2000, Washington, u prosincu 2000.
- članak T. Berners-Lee, J. Hendler, O. Lassila: The Semantic Web. Scientific American, 284(5), pp. 34-43, 2001
 - *Semantički Web će dati strukturu sadržaju web-stranica u smislu njegovog značenja, stvarajući okolinu u kojoj će programski agenti, prelazeći sa stranice na stranicu, moći obavljati sofisticirane zadatke za svoje korisnike*

- članak T. Berners-Lee, J. Hendler, O. Lassila: The Semantic Web. Scientific American, 284(5), pp. 34-43, 2001
 - *Takav agent, dolazeći na stranicu klinike, neće samo znati da stranica sadrži ključne riječi kao što su "zahvat", "medicina", "fizički", "terapija" (što je i danas moguće ugraditi u kod stranice), nego također i da Dr. Hartman u navedenoj klinici radi ponедjeljkom, srijedom i petkom te da postoji skripta koja uzima datum u formatu DD.MM.GGGG. i vraća zakazano vrijeme pregleda za taj dan. Agent će sve ovo „znati“ bez potrebe za umjetnom inteligencijom na razini Hala iz filma 2001. – Odiseja u svemiru ili C-3PO iz serije filmova Ratovi zvijezda. Umjesto toga, semantiku će u web-stranicu ukodirati voditelj ureda klinike (koji nikad nije pohađao niti najosnovniji tečaj programiranja) koristeći jednostavni program za stvaranje stranica Semantičkog Weba i resurse navedene na stranici Udruženja specijalista fizikalne terapije*

Semantički Web: ideja

- ◆ predavanje u sklopu simpozija XML 2000, Washington, u prosincu 2000.: već je dan inicijalni sklop protokola



originalni složaj
iz 2000.

S. Bratt: Semantic Web, and Other Technologies to Watch.
Predavanje u siječnju 2007

Nedostaci "klasičnog" Weba

- ◆ informacije na Webu daju
 - rečenice prirodnog jezika
 - slike
 - multimedijski sadržaji (zvuk, video)
- ◆ kako "zaključuju" računala?
 - potrebna stroga i sintaksno specifikacija
 - već prirodni jezik predstavlja poteškoću (idiomi, slikovit govor...)
 - problem nadopune konteksta u slučaju nepotpunih informacija
 - problem integracije informacija (Magic, Gretzky i LA)
 - problem interpretiranja slike i zvuka odnosno njihovog povezivanja s pojmovima

Nedostaci "klasičnog" Weba: tražilice

- ◆ HTML
 - spojena vizualna prezentacija (izgled slova, novi red, veličina proreda) sa sadržajem
- ◆ traženje prema pojedinim ključnim riječima
 - visoki odziv (eng. *recall*), mala preciznost (eng. *precision*) tj. tražilice gotovo uvijek pronađu mnogo dokumenata, ali je njihova korisnost i upotrebljivost često mala tj. ne dobije se ono što se želi
 - problem jezika
 - slaba mogućnost pretraživanja slika i multimedije

Problemi klasičnog Weba

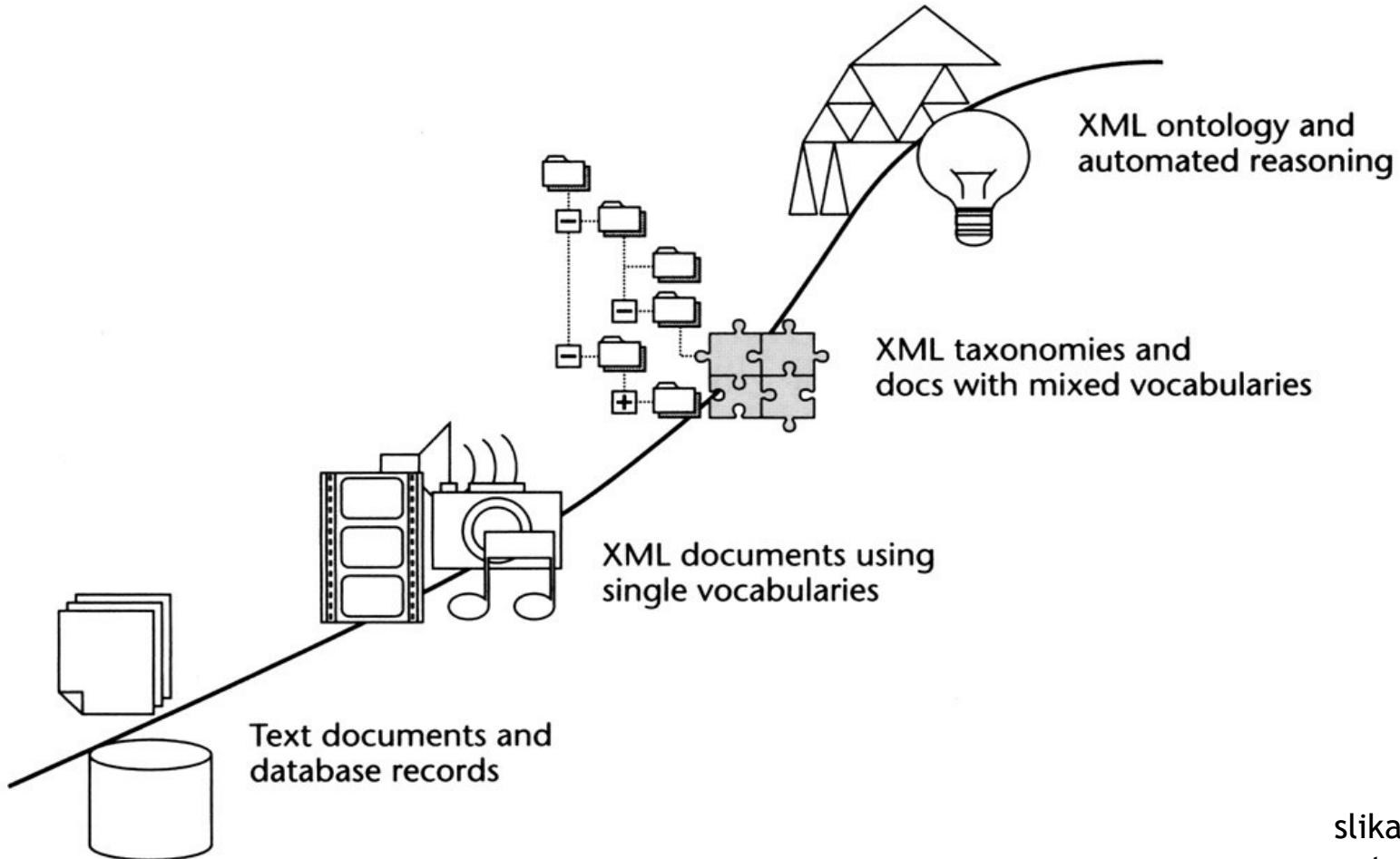
- ◆ problem Weba temeljenog na sintaksi:
 - kompleksni upiti koji zahtjevaju pozadinsko znanje:
 - pronaći informacije o svim životinjama koje upotrebljavaju sonar, a nisu šišmiši ili dupini!
 - lociranje podataka unutar podatkovnih repozitorija:
 - putni nalozi
 - cijene roba i usluga
 - dodjeljivanje kompleksnih zadataka Web agentima:
 - rezervirati odmor na nekom toplom mjestu, sljedeći vikend, ne jako daleko, i gdje govore francuski ili engleski

- ◆ Načiniti podatke “pametnijima”: dodati im metapodatke
 - kakvi trebaju biti metapodaci?

- ◆ RDF kao zapis trojki; RDF Schema kao metamodel
 - dobra podloga
 - taksonomije klasa i svojstava
 - domena i kodomena (*range*) za svako svojstvo
 - još uvijek ne omogućuje sofisticirane izraze:
 - osobaSBarTrojeDjece kao potklasa klase Osoba za koju postoje bar tri instance klase Osoba s kojima je povezana svojstvom imaDijete (u ulozi subjekta)
 - osobaBezSina kao potklasa klase Osoba za koju sve instance s kojima je povezana svojstvom imaDijete (u ulozi subjekta) spadaju u klasu Žena (Žena je potklasa klase Osoba)
 - disjunktnost pojedinih klasa
 - inverznost pojedinih svojstava

- ◆ **ontologija** (u računarstvu): **formalna, eksplisitna** specifikacija neke **dijeljene konceptualizacije**
- ◆ **konceptualizacija**: postojanje apstraktnog modela nekog fenomena kojem su identificirani pojmovi (eng. concepts) vezani uz njega.
- ◆ **eksplicitnost**: eksplisitno je definiran tip koncepta i dana su ograničenja za njegovu uporabu
- ◆ **formalnost** podrazumijeva čitljivost ontologije od strane računala
- ◆ **dijeljenost**: ontologija treba predstavljati konsenzus znanja (što šire) skupine ljudi, a ne isključivo pojedinca
Filozofija (Aristotel): znanost o bitku (grč. ὄντος)

Kvaliteta metapodataka



Slika: zapisivanje podataka - povijesni razvoj

slika:
autor dr. sc.
Damir Jurić

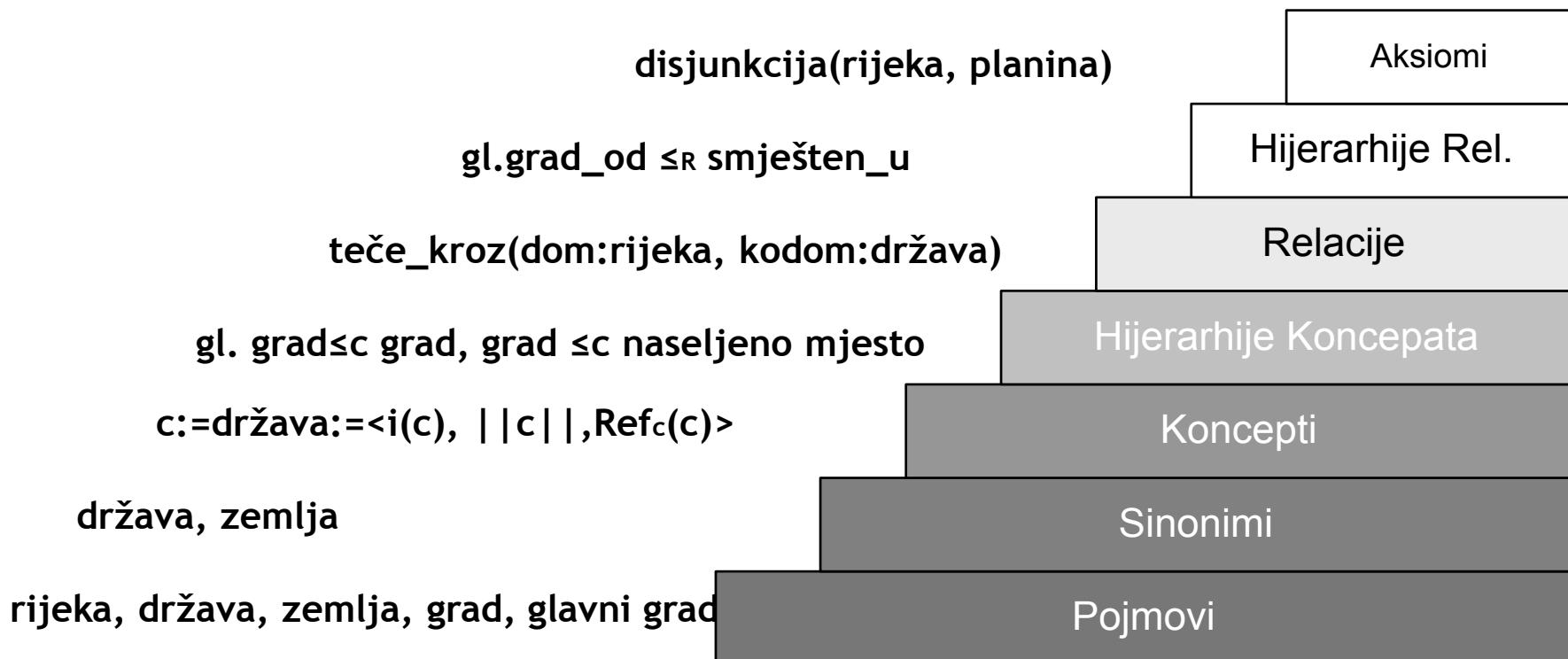
- ◆ Tekst i baze podataka
 - Početna faza
 - Logika se nalazi u aplikacijama, a ne u podacima
- ◆ XML dokumenti
 - Neovisnost podataka od aplikacije za određenu domenu
 - Podatke koristi više aplikacija
- ◆ Taksonomije i rječnici
 - Podaci se odnose na više različitih domena
 - Postoji precizna klasifikacija
 - Postoje jednostavne relacije između kategorija

Kvaliteta metapodataka

- ◆ Ontologije i pravila (eng. rules)
 - U ovoj fazi postoji mogućnost dobivanja novih podataka iz onih već postojećih
 - Podaci su “pametno” organizirani i opisane su konkretne relacije te sofisticirani formalizmi
 - Moguće je kombinirati i rekombinirati podatke na njihovoj elementarnoj razini
 - Primjer: automatsko prevođenje dokumenta iz jedne domene u ekvivalentni dokument iz druge domene

Ontologija

- ♦ Slojevi procesa učenja ontologija:



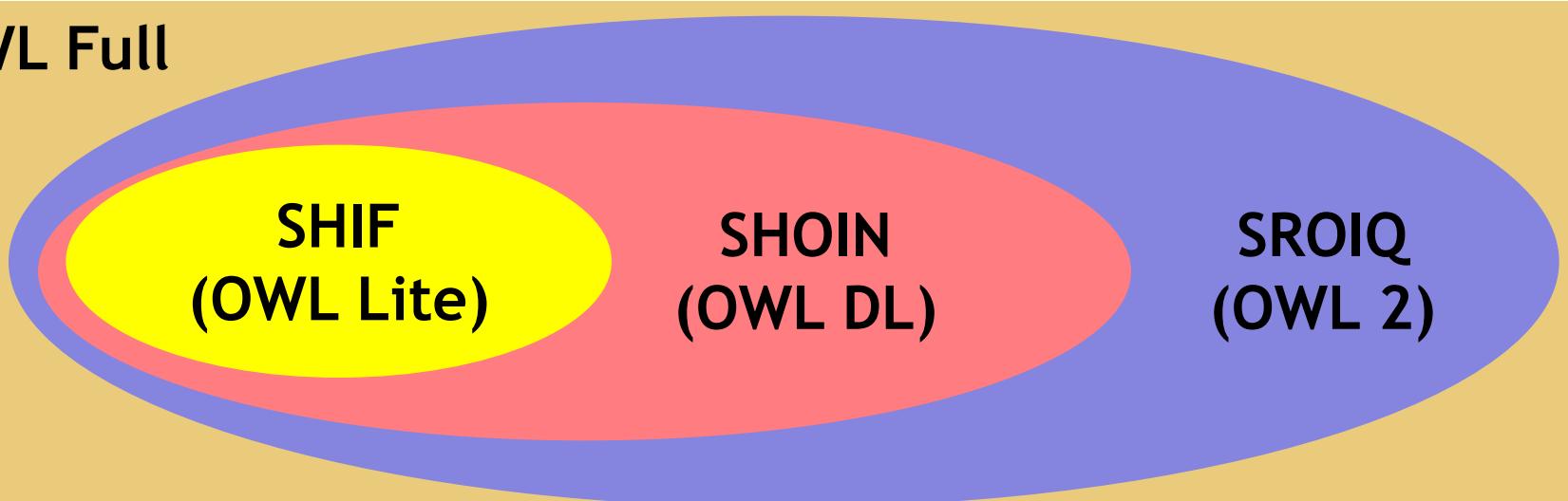
autor primjera
dr. sc. Damir
Jurić

- ◆ Web Ontology Language
- ◆ OWL 1 - Preporuka W3C od veljače 2004.
 - Temeljen je na starijim jezicima (DAML+OIL)
- ◆ OWL 2 - Preporuka W3C od prosinca 2012.
 - ispravlja neke nedostatke OWL 1
- ◆ OWL (1 i 2) matematički je zasnovan na opisnoj logici (*description logic*)

- ◆ Jezik za Web: temeljen na RDF(S)
- ◆ Jezik za ontologije: temeljen na logici

- ◆ ekspresivnost logike
 - raste s brojem konstruktora koje ona podržava
- ◆ porastom ekspresivnosti potencijalno se smanjuje
 - **izračunljivost** (eng. *computational completeness*) svojstvo da se svaki pojmovni opis može obraditi s logičke strane tj. na svako pitanje dati pozitivan ili negativan odgovor
 - **odlučivost** (eng. *decidability*), svojstvo da je izračunljivost garantirano ostvariva u konačnom vremenu.

OWL Full



- ◆ Dijelovi OWL ontologije:
 - Klase (Classes)
 - Svojstva (Properties)
 - Relacije između klasa (Relations)
 - Restrikcije nad svojstvima (Restrictions)
 - Karakteristike svojstava
 - Anotacije
 - Instance

OWL

- ◆ 5 različitih sintaksi

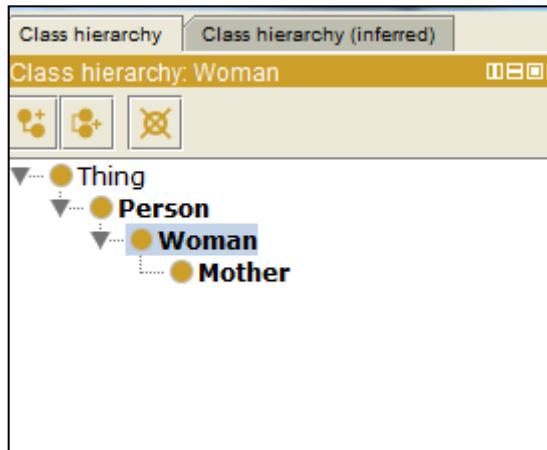
- funkcionalna
- RDF/XML
- mančesterska
- Turtle
- OWL/XML

| | |
|--|---|
| | Declaration(NamedIndividual(:John)) Declaration(Class(:Person)) Declaration(ObjectProperty(:hasWife)) Declaration(DataProperty(:hasAge)) |
| | <owl:NamedIndividual rdf:about="John"/> <owl:Class rdf:about="Person"/> <owl:ObjectProperty rdf:about="hasWife"/> <owl:DatatypeProperty rdf:about="hasAge"/> |
| | Individual: John Class: Person ObjectProperty: hasWife DataProperty: hasAge |
| | :John rdf:type owl:NamedIndividual . :Person rdf:type owl:Class . :hasWife rdf:type owl:ObjectProperty . :hasAge rdf:type owl:DatatypeProperty . |
| | <Declaration><NamedIndividual IRI="John"/> </Declaration> <Declaration><Class IRI="Person"/> </Declaration> <Declaration><ObjectProperty IRI="hasWife"/> </Declaration> <Declaration><DataProperty IRI="hasAge"/> </Declaration> |

OWL: klasa i njene potklase

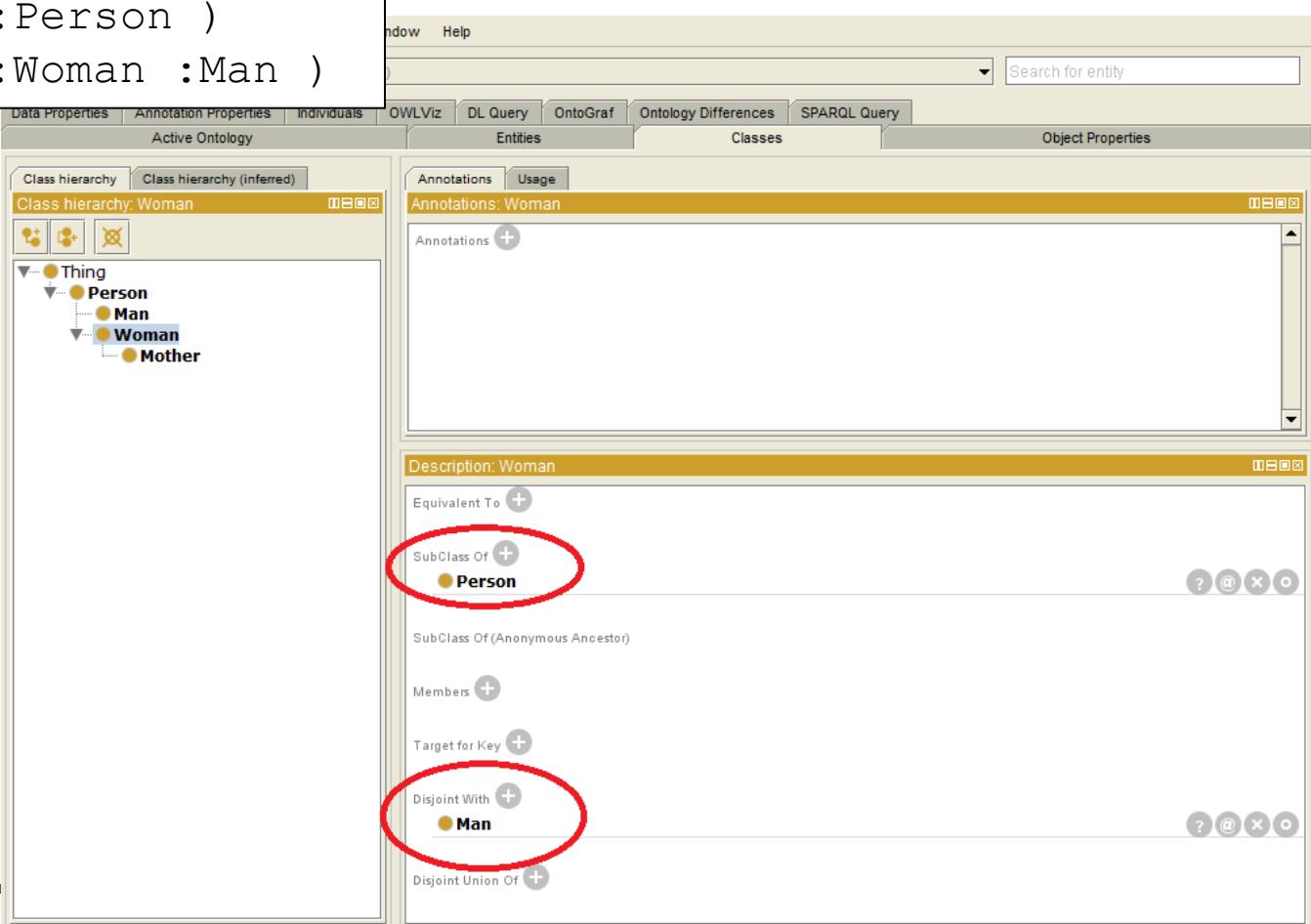
```
Declaration( Class( :Woman ) )
Declaration( Class( :Mother ) )
SubClassOf( :Woman :Person )
SubClassOf( :Mother :Woman )
```

Vizualizacija: koristimo alat Protege 4.3 (otvoreni kod; zasnovan na Javi)



OWL – disjunktne klase

```
Declaration( Class( :Woman ) )
Declaration( Class( :Man ) )
SubClassOf( :Woman :Person )
SubClassOf( :Man :Person )
DisjointClasses( :Woman :Man )
```



The screenshot shows the OntoGraf interface with the following panels:

- Top Bar:** Window, Help, OWLViz, DL Query, OntoGraf, Ontology Differences, SPARQL Query.
- Left Panel (Data Properties):** Entities, Classes, Object Properties.
- Middle Left Panel (Active Ontology):** Class hierarchy (inferred). Shows a tree structure: Thing -> Person -> Man -> Woman -> Mother.
- Middle Right Panel (Annotations):** Annotations tab. Shows Annotations for the Woman class.
- Bottom Panel (Description):** Description: Woman. Shows annotations for the Woman class, including:
 - Equivalent To: +
 - SubClass Of: + (highlighted with a red circle)
 - SubClass Of (Anonymous Ancestor):
 - Members: +
 - Target for Key: +
 - Disjoint With: + (highlighted with a red circle)
 - Disjoint Union Of: +

analiziramo klasu
Woman: vidimo da
je potklasa klase
Person i disjunktna
klasi **Man**

OWL - presjek

```

Declaration( Class( :Woman ) )
Declaration( Class( :Parent ) )

EquivalentClasses(
    :Mother
    ObjectIntersectionOf( :Woman :Parent )
)
    
```

Class hierarchy: Mother



```

graph TD
    Thing --> Parent
    Parent --> Person
    Person --> Man
    Person --> Woman
    Woman --> Mother
    
```

Annotations: Mother

Annotations +

Description: Mother

Equivalent To +

- Parent
- and Woman

SubClass Of +

- Woman

SubClass Of (Anonymous Ancestor)

Members +

Annotations: Mother

Annotations +

Annotations: Mother

Annotations +

Annotations: Mother

Annotations +

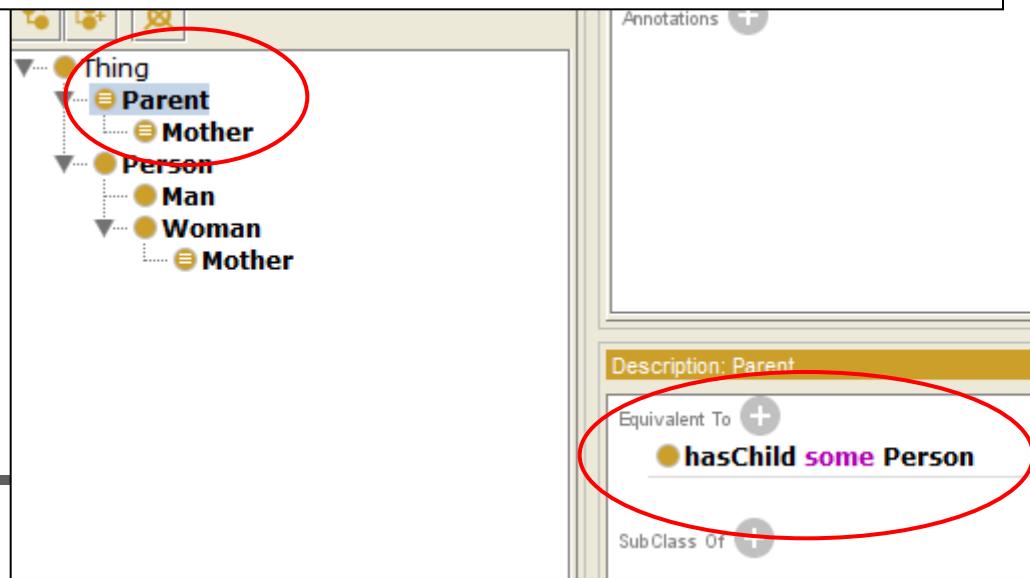
Mother: je presjek klasa Parent i Woman (time je ujedno i potklasa obiju klasa)

OWL – egzistencijalni kvantifikator

- ◆ egzistencijalni kvantifikator $\exists R.C$ zahtjeva da je svaka instanca zadane klase svojstvom R kao subjekt povezana s bar jednim objektom klase C
 - C se može izostaviti tj. promatrati kao kodomena se uzima *Thing*)

```
Declaration( ObjectProperty( :hasChild ) )
EquivalentClasses(
  :Parent
  ObjectSomeValuesFrom( :hasChild :Person )
)
```

Parent (roditelj) je nešto (potklasa klase *Thing*) što se referencira na bar jednu osobu (*Person*) svojstvom *hasChild*



OWL – univerzalni kvantifikator

- ◆ univerzalni kvantifikator $\forall R.C$ zahtijeva da je instanca zadane klase svojstvom R bude povezana isključivo s objektima klase C (a klasa C je potklasa klase koja je postavljena kao kodomena svojstva općenito)
 - ako za svojstvo nisu definirani domena i/ili kodomena, onda je to po definiciji klasa *Thing* (koja je nadređena svim klasama i obuhvaća sve instance ontologije)

```
EquivalentClasses (
    :PersonWithoutSons
    ObjectIntersectionOf (
        :Person
        ObjectAllValuesFrom( :hasChild :Woman )
    )
)
```

PersonWithoutSons (osoba bez sinova) je osoba (*Person*) koja se svojstvom *hasChild* referencira isključivo na ženske osobe (*Woman*) [ovdje su uključene i osobe koje uopće nemaju djece tj. ne referenciraju se ni na koji objekt svojstvom *hasChild*]

OWL – ograničenje broja

- ◆ poopćenje egzistencijalnog kvantifikatora: zahtjeva se da je svaka instanca zadane klase svojstvom R kao subjekt povezana s minimalno/maksimalno/točno n objekata klase C

```
EquivalentClasses (
    : PersonWithAtLeastThreeChildren
    ObjectIntersectionOf (
        :Person
        ObjectMinCardinality(3 :hasChild )
    )
)
```

PersonWithAtLeastThreeChildren (osoba s bar troje djece) je osoba (*Person*) koja se svojstvom *hasChild* referencira tri ili više objekata klase C ili pak bilo kakvih objekata (klase *Thing*) ako se C izostavi [ovdje je C izostavljen]

- ♦ upitni jezik za rad nad podacima pohranjenim u obliku RDF trojki subjekt-predikat-objekt
 - srođan SQL-u

```
PREFIX pr: <http://example.org/book/>
pr:book1 pr:title "SPARQL Tutorial".
```

Sadržaj se sastoji od jedne jedine trojke:
za resurs *book1* vrijednost svojstva *title* je *SPARQL Tutorial*

```
PREFIX pr: <http://example.org/book/>
SELECT ?title
WHERE
{
  pr:book1 pr:title ?title .
}
```

Ispisati sve vrijednosti svojstva *title* gdje je subjekt *book1*

- ◆ složeniji upiti: uvode se interne varijable

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .  
@prefix : <http://example.org/book/> .  
@prefix ns: <http://example.org/ns#> .  
  
:book1 dc:title "SPARQL Tutorial" .  
:book1 ns:price 42 .  
:book1 ns:author "John Smith" .  
:book2 dc:title "The Semantic Web" .  
:book2 ns:price 23 .
```

knjiga *book1* ima naslov (*title*), cijenu (*price*) i autora (*author*), a knjiga *book2* samo naslov i cijenu

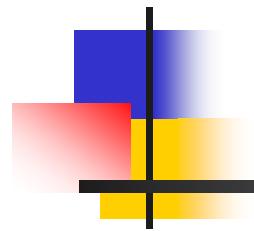
```
PREFIX dc: <http://purl.org/dc/elements/1.1/>  
SELECT ?title ?author  
WHERE {  
    ?x dc:title ?title  
    ?x ns:author ?author  
}
```

Za resurse koji imaju i naslov i autora ispisati taj naslov i tog autora:
ispiše se samo *SPARQL Tutorial - John Smith* (tj. podaci za *book1*)

Zaključak

Department of Telecommunications

- ◆ OWL je preporučeni standard za definiranje ontologija
- ◆ omogućava opisivanje semantike na način razumljiv računalima
- ◆ Očekuje se daljnji razvoj alata koji će biti u mogućnosti koristiti sve mogućnosti koje pružaju ontologije
- ◆ Semantički Web će zaživjeti tek kada se stvori kritičan broj resursa na Webu koji implementiraju neke od ovdje opisanih tehnologija



XML

(eXtensible Markup Language)

Upravljanje podacima
2015./2016.

Strukturirani i polustrukturirani podaci

- Kod **strukturiranih** podataka **shema je odvojena od podataka**. Najprije se definira shema, a potom se počinju unositi podaci koji odgovaraju shemi.
- **Primjer:** **relacijske** baze podataka gdje se najprije zadaje relacijska shema, a zatim stvara relacija koja se može popunjavati podacima.
- **Polustrukturirani** podaci mogu imati nepravilnu i/ili implicitnu strukturu.

Strukturirani i polustrukturirani podaci

Relacijski model – strukturirani podaci

**atribut
– primarni ključ**

**neklučni
atribut**

n-torka

| BRZAP | IMEZAP | POSAO | BRODJ |
|-------|--------|------------|-------|
| ----- | ----- | ----- | ----- |
| 1369 | DARKO | SLUŽBENIK | 20 |
| 1499 | ANA | PRODAVAČ | 30 |
| 1521 | VEDRAN | PRODAVAČ | 30 |
| 7566 | JOSIP | UPRAVITELJ | 20 |

Strukturirani i polustrukturirani podaci

Polustrukturirane podatke karakterizira barem jedno od sljedećih dvaju osnovnih **obilježja**:

- **Nepravilna struktura**
- **Implicitna struktura**

Strukturirani i polustrukturirani podaci

Nepравилна структура

- Kod polustrukturiranih podataka struktura nije uvek pravilna, dopuštene su varijacije u strukturi.
- Veliki skupovi podataka često se sastoje od manjih podatkovnih jedinica čija struktura je međusobno slična, ali ne i posve jednaka.

```
{autor: { prezime: "Poe"}, {ime: "Edgar"}, {srednje: "Allan"}}
```

```
{autor: { prezime: "Gide"}, {ime: "Andre"}}
```

```
{autor: { prezime: "Jesenjin"}, {ime: "Sergej"}, {očevo: "Aleksandrovič"}}
```

- Shema nije strogo ograničavajuća i bez iznimaka kao u slučaju potpuno strukturiranih podataka u relacijskoj bazi podataka.

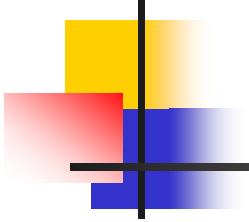
Strukturirani i polustrukturirani podaci

Implicitna struktura

- **Shema** podatkovne strukture **ne mora biti pohranjena odvojeno** (u posebnoj shemi ili dokumentu), nego može biti integrirana zajedno s njihovim sadržajem.
- Informacije koje su obično vezane uz shemu ovdje se nalaze zajedno s podacima.

Primjer: Značenje podatka "Verdi" nije napisano u odvojenom vanjskom dokumentu nego je upravo uz sadržaj podataka navedeno i što on predstavlja (prezime).

```
{kompozitor:  
 {prezime:"Verdi"}{ime="Giuseppe"}  
 {djelo: {opera:"Traviata"} }  
 {djelo: {opera:"Don Carlos"} } }
```



XML

- eXtensible Markup Language
- XML je **markup-jezik** kao i HTML
 - elektroničke oznake (*tagovi*) izmiješane su s tekstom/podacima
- Razlika: HTML određuje prikaz podataka, dok **XML opisuje podatke.**
- Tagovi u XML-u nisu unaprijed definirani. Potrebno je **definirati vlastite tagove.**
- XML – polustrukturirani podaci
- Korištenjem XML-a podaci se mogu razmijenjivati među raznorodnim sustavima.

XML

- Nastao kao podskup jezika SGML (Standard Generalized Markup Language), jednostavniji je od SGML-a
- XML je postao preporuka organizacije W3C (verzija XML 1.0) 1998. godine
- XML 1.1 (Second Edition) 2006. godine

- Primjena u slučajevima kad je potreban općeniti standard zapisa:
 - komunikacijski paketi na Internetu (SOAP poruke)
 - zapis aplikacijskih podataka neovisan o platformi
(zamjena za format *comma-separated values*)

Korištenje XML-a

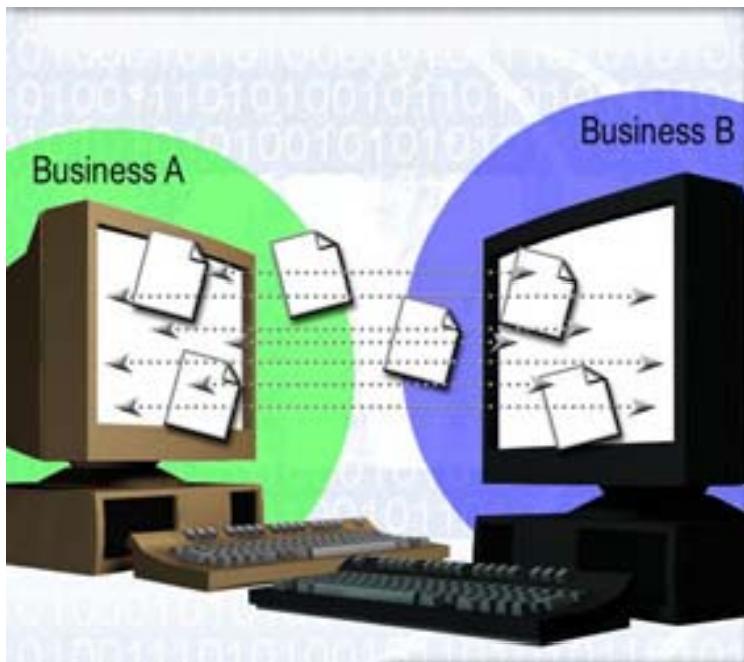
- **Razmjena podataka** među aplikacijama, neovisno o korištenom hardveru i softveru.



- Podaci mogu biti pohranjeni u XML formatu u baze podataka.
- Podaci su lakše dostupni raznim korisnicima, bilo na Web-u, bilo preko raznih aplikacija.

Korištenje XML-a

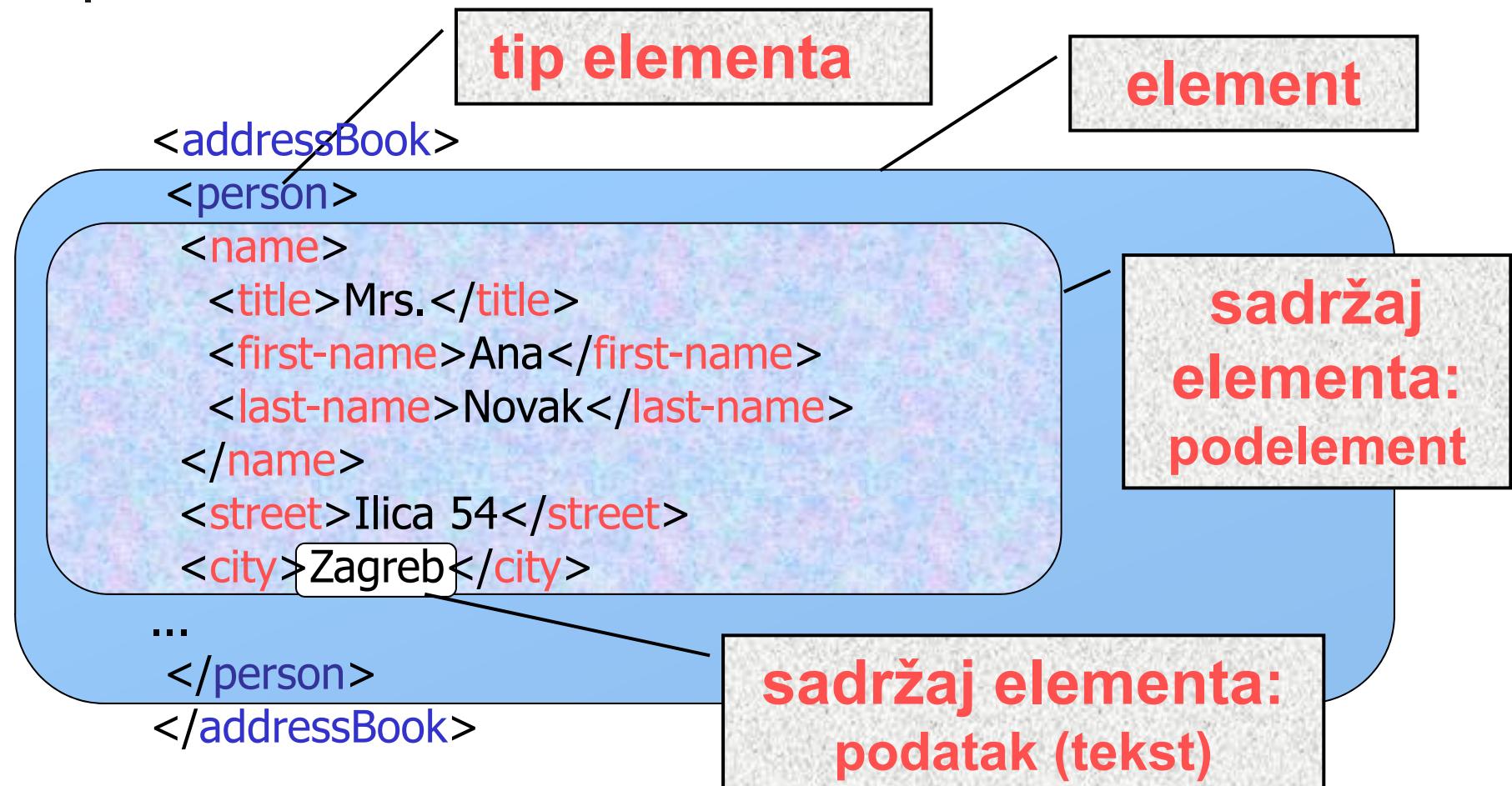
- Komunikacija tipa Business-to-Business
 - Razmjena podataka (npr. o narudžbama, računima...) između tvrtki uključenih u B2B.
 - Podaci se razmjenjuju između baze podataka i Web-sjedišta za elektroničku trgovinu



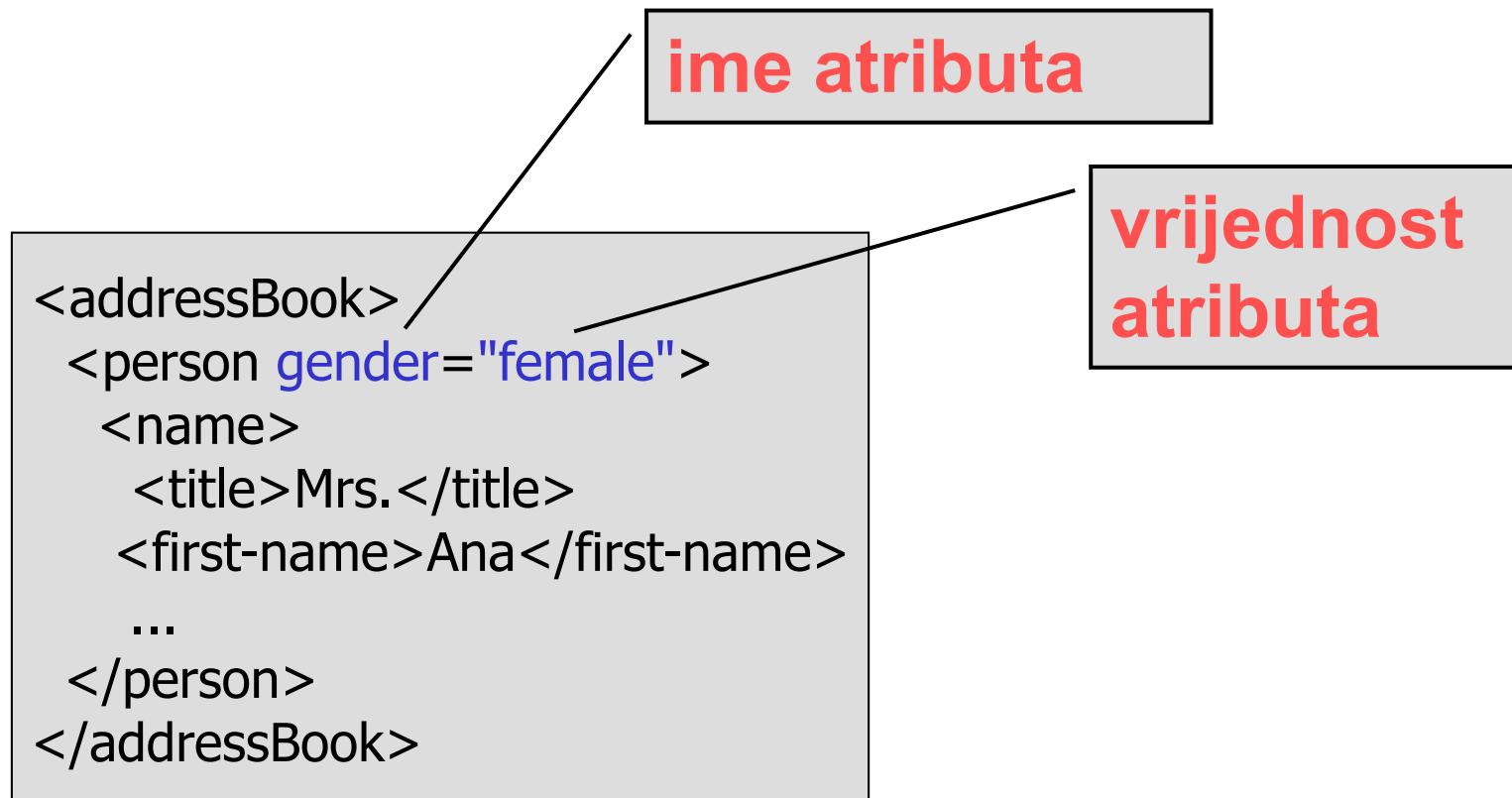
Korištenje XML-a

- XML se može koristiti za stvaranje novih jezika, npr:
 - MathML – uključuje matematičku notaciju i sadržaj. Integrira matematičke formule u dokumente

Sintaksa XML-a



Sintaksa XML-a



Dobro oblikovan dokument

- Dobro oblikovani dokument zadovoljava pravila oblikovanja XML dokumenta.
- Svaki element mora imati zatvarajući tag.
- Tagovi su “case sensitive”.

```
<Poruka>Ovo nije dobro</poruka>
<poruka>Ovo je dobro</poruka>
```

Dobro oblikovan dokument

- Ako neki element nema nikakav sadržaj, može se koristiti sljedeća sintaksa:

```
<kolicina mjerJed= "kg"/>
```

što je isto kao:

```
<kolicina mjerJed="kg"> </kolicina>
```

- Tagovi se smiju ugniježđivati, ali se ne smiju isprepletati.

```
<name>
  <first-name>Ana</first-name>
  <last-name>Novak</last-name>
</name>
```

~~<A atr1="x">
 <B atr1='y'>

 ~~

Dobro oblikovan dokument

- Vrijednosti atributa moraju biti u navodnicima.

```
<knjiga cijena = "450" valuta = "HRK">  
...  
</knjiga>
```

- Preporučuje se započeti XML dokument deklaracijom XML-a.

```
<?xml version="1.0" ?>
```

Dobro oblikovan dokument

Znakovi:

<

>

&

'

"

moraju se uvijek pisati na sljedeći način:

<

>

&

'

"

- **Primjer:**

```
<element> ovo je manji od &lt; /element>
```

Valjan XML dokument

- XML dokument je **valjan** ako:
 - je dobro oblikovan i
 - slijedi pravila koja propisuje njemu pridružen DTD ili XML Schema
- DTD i XML Schema omogućavaju opis strukture i postavljanje ograničenja nad sadržajem XML dokumenta.

Document Type Definition (DTD)

- definira:

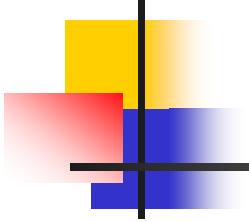
- koji elementi su na raspolaganju,
- slijed elemenata,
- kardinalnost pojavljivanja nekog elementa,
- dopuštene strukture ugnježđivanja elemenata,
- atribute koje određeni element može imati,
- vrijednosti koje atributi mogu poprimiti.

Document Type Definition (DTD)

- Korištenjem DTD-a, XML dokumenti uz sebe mogu imati i opis vlastite strukture.
- Struktura dokumenta može se vidjeti odvojeno od samih podataka.
- DTD može biti zajednički za više dokumenata. Moguće je dogovoriti zajednički DTD za razmjenu podataka.
- Različite aplikacije mogu pomoću standardnog DTD-a verificirati da su primljeni podaci valjni.

XML dokument - primjer

```
<addressBook>
  <person>
    <name>
      <title>Mrs.</title>
      <first-name>Ana</first-name>
      <last-name>Novak</last-name>
    </name>
    <street>Ilica 54</street>
    <city>Zagreb</city>
    <email>ana.novak@fer.hr</email>
  </person>
  <person>
    ...
  </person>
</addressBook>
```



DTD - primjer

```
<!ELEMENT addressBook (person)+>
<!ELEMENT person (name, street, city, email*)>
<!ELEMENT name (title?, first-name, last-name)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT last-name (#PCDATA)>
<!ELEMENT first-name (#PCDATA)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT e-mail (#PCDATA)>
```

- Samo je jedan korijenski element i čitav sadržaj XML dokumenta jest sadržaj tog elementa.

Elementi u DTD-u

- Sadržaj elementa:
 - složeni = izraz koji sadrži druge elemente (podelemente)
 - #PCDATA = parsed character data
 - EMPTY = nema podataka ni podelemenata
 - ANY = bilo koja kombinacija elemenata i podataka
 - (#PCDATA | A | B | C)* = miješani sadržaj

DTD elementi

| Definicija elementa | Značenje |
|---------------------|--|
| A? | Jedno ili nijedno pojavljivanje A; opcionalni A. |
| A+ | Jedan ili više pojavljivanja A. |
| A* | Nijedan ili više pojavljivanja A. |
| A B | A ili B, ali ne oboje. |
| A , B | A pa zatim B. |
| (A, B)+ | Jedan ili više pojavljivanja kombinacije (A pa B). Zagrade se koriste radi grupiranja. |
| #PCDATA | Ključna riječ koja označava znakovni niz. |

Atributi u DTD-u

Deklaracija atributa:

```
...
<!ELEMENT person (name, street, city, email*)>
<!ATTLIST person gender (male|female) #IMPLIED>
...
```

Valjani XML dokument:

```
<addressBook>
  <person gender="female">
    <name>
      <title>Mrs.</title>
      <first-name>Ana</first-name>
    ...
  </person>
</addressBook>
```

Atributi u DTD-u

- **Tipovi atributa:**

- CDATA
- (vrijednost1 | vrijednost2)
- ID
- IDREF
- IDREFS

...

- **Vrste atributa:**

- #REQUIRED - atribut se mora koristiti
- #IMPLIED - atribut se ne mora koristiti
- value - default vrijednost
- #FIXED "value" - jedina dopuštena vrijednost

- **Vrijednost:**

- character data
- samo 1 vrijednost iz popisa
- jedinstveni id
- id nekog drugog elementa
- popis drugih id-a

Provjera valjanosti korištenjem DTD-a

- U XML dokument može se uključiti:
 - ili cijeli DTD:

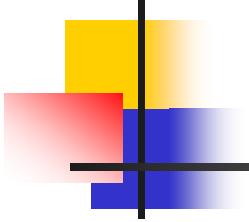
```
<!DOCTYPE korijenski_element [  
    .....  
]>
```

- ili referencu na DTD :

```
<!DOCTYPE korijenski_element SYSTEM "ime_datoteke">
```

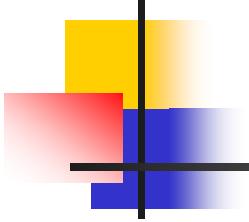
Provjera valjanosti korištenjem DTD-a

```
<?xml version="1.0" ?>
<!DOCTYPE addressBook [
<!ELEMENT addressBook (person)+>
<!ELEMENT person (name, street, city, email*)>
...
]>
<addressBook>
  <person>
    <name>
      ...
    </name>
  </person>
</addressBook>
```



XML Schema vs. DTD

- DTD
 - Ograničenja: tipovi podataka, ključevi, ...
 - Sintaksa drugačija od XML-a
- XML Schema
 - Napisana u XML-u
 - Podržava različite tipove podataka
 - Omogućava stvaranje novih tipova podataka
 - Kardinalnost se može detaljnije specificirati
 - Bolje riješeni ključevi i referenciranje
 - Nedostatak: obimna i složena



XML Schema

- preporuka W3C od 2001. godine
- Značajno proširuje mogućnosti DTD-a

- ***XML Schema Part 0: Primer***
<http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/>

- XML Schema sastoji se od:
 - definicija tipova elemenata (koje mogu biti izvedene iz drugih definicija)
 - i deklaracija elemenata.

XPath

Upravljanje podacima
2015./2016.

- ◆ sintaksa za opisivanje (dohvaćanje) dijelova XML dokumenta
- ◆ primjer: “u XML dokumentu *Narudžbenica* dohvati šifre svih proizvoda pod kategorijom *Kozmetika* ”

XML - stablo čvorova

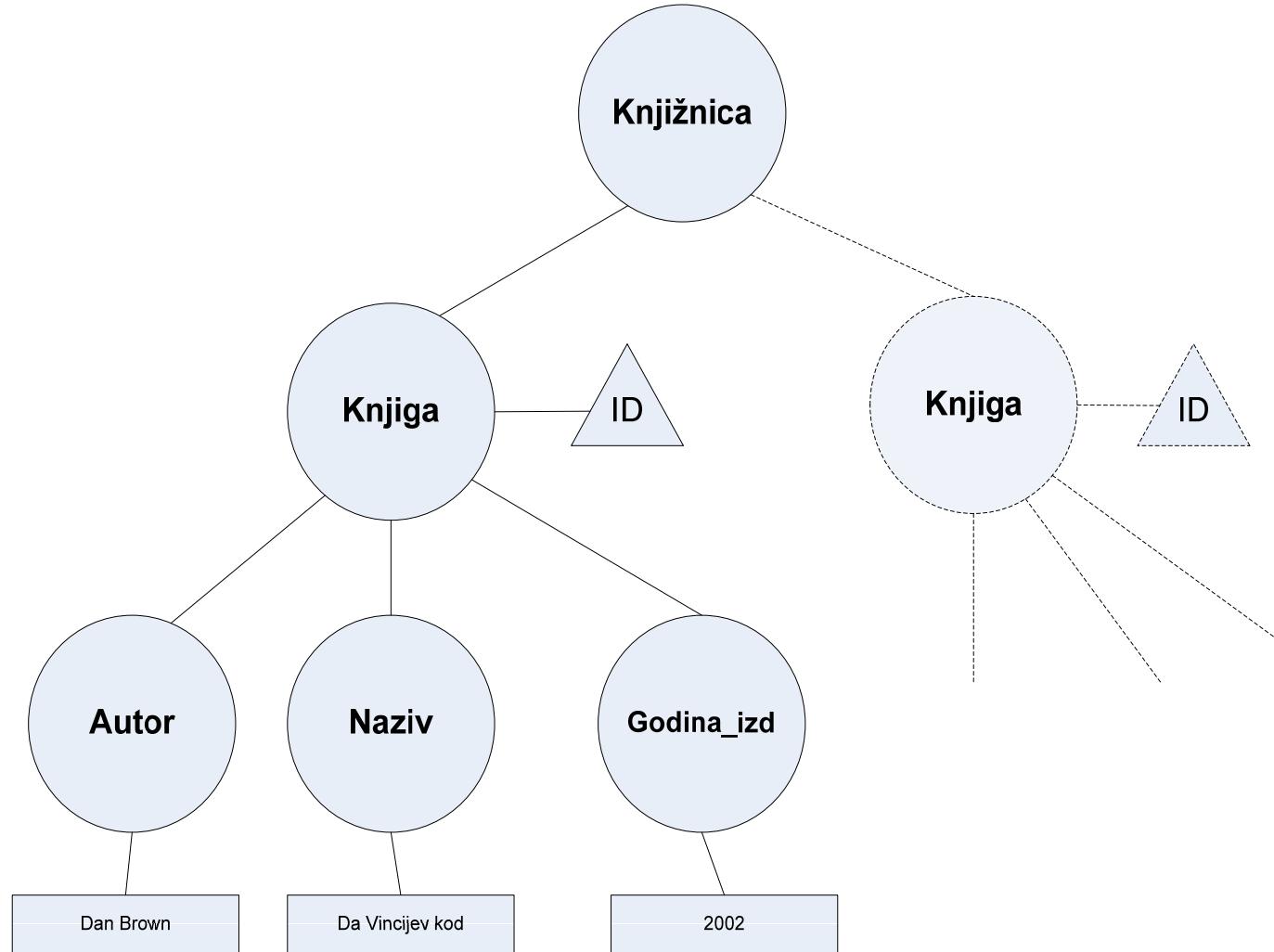
- ◆ XPath gleda XML dokument kao stablo čvorova
- ◆ 7 tipova čvorova:
 - korijenski čvor
 - element
 - atribut
 - tekst
 - komentar
 - procesorska instrukcija
 - imensko područje (imenik, *namespace*)

Primjer

Knjižnica.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<knjižnica>
    <knjiga ID="1001">
        <autor>Dan Brown</autor>
        <naziv>Da Vinciјev kod</naziv>
        <godina_izd>2002</godina_izd>
    </knjiga>
    <!-- drugе knjige -->
</knjižnica>
```

Stablo čvorova



Kako se koristi XPath?

- ◆ XPath je najčešće pomoćna tehnologija koju aplikacije koriste za dohvat informacija iz XML dokumenta
- ◆ stvara se tzv. *lokacijska staza* koja referencira dio dokumenta
- ◆ princip *lokacijske staze* vrlo je sličan strukturi staze direktorija kod UNIX ili Windows operativnih sustava

- ◆ XPath se uvijek referencira na **kontekst** - princip konteksta je sličan pozicioniranju u određeni direktorij
- ◆ najčešće se kao početni kontekst uzima korijenski čvor, te se od njega “putuje” kroz dokument
- ◆ korijenski čvor označava se simbolom “/”

Lokacijske staze - primjeri

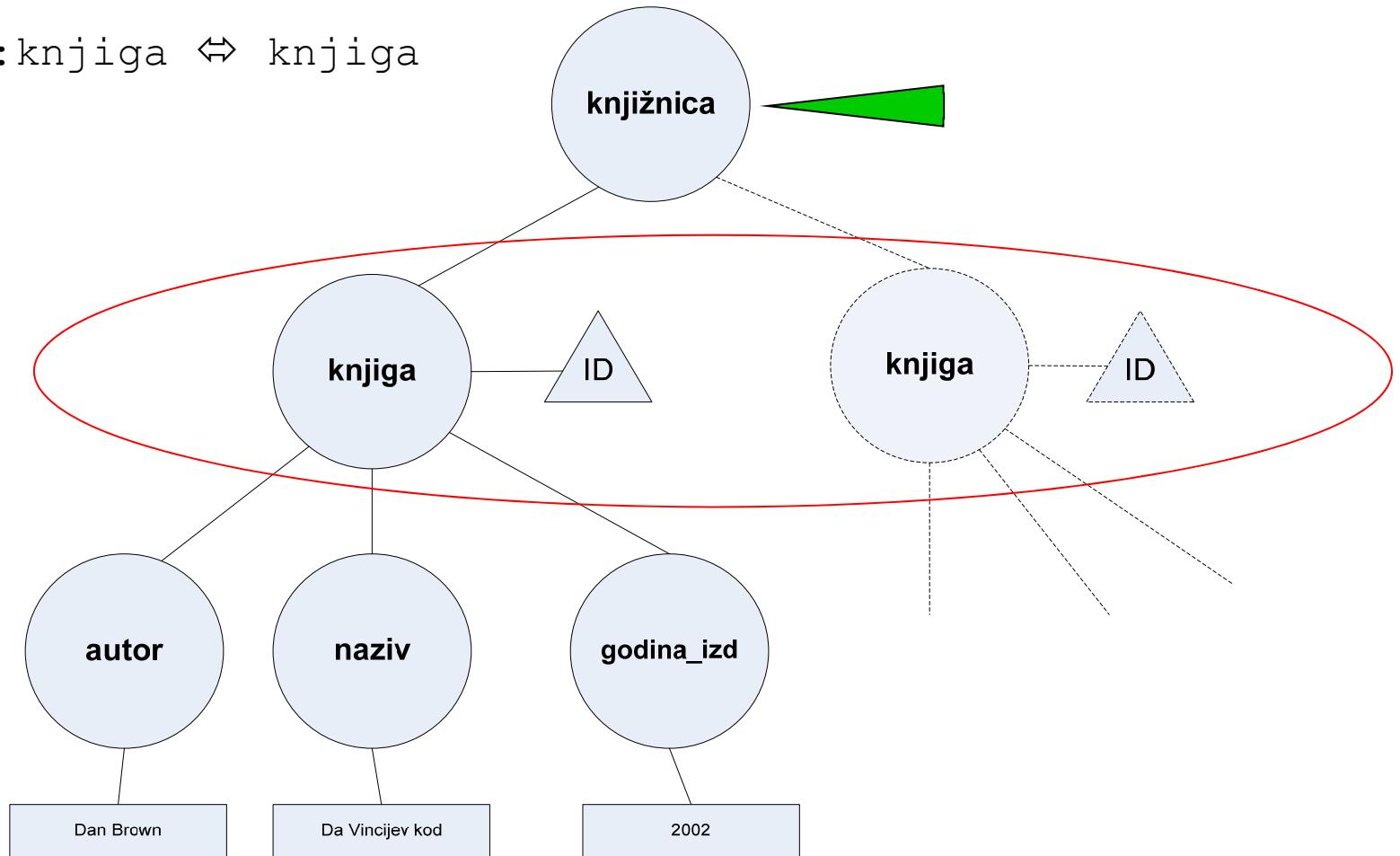
| | |
|---------------------------------------|---|
| / | korijenski čvor dokumenta |
| . | trenutni kontekst |
| .. | roditelj trenutnog konteksta |
| autor | elementi <i>autor</i> u trenutnom kontekstu (relativna staza) |
| /knjižnica/knjiga | svi elementi <i>knjiga</i> – apsolutna staza od korijenskog čvora |
| knjiga/@ID | atribut <i>ID</i> elemenata <i>knjiga</i> |
| //autor/text() | tekst elemenata <i>autor</i> ; nije bitna staza do elemenata <i>autor</i> u dokumentu |
| /knjižnica/* ili /knjižnica/node() | svi elementi ispod elemenata <i>knjižnica</i> |

Lokacijske staze (2)

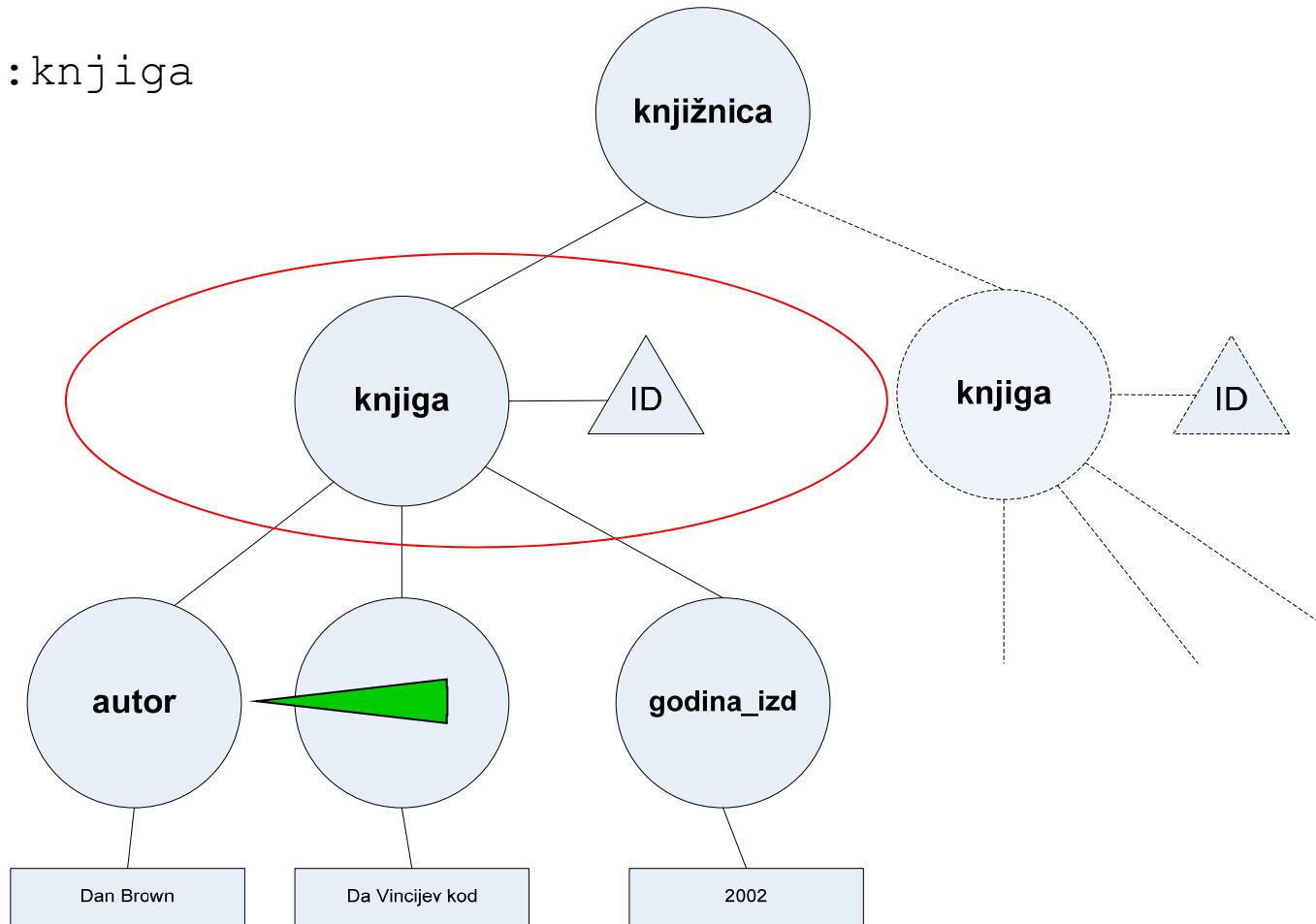
| | |
|---|---|
| <code>child::knjiga</code> ↔ <code>knjiga</code> <code>parent::*</code> ↔ <code>..</code> <code>self::*</code> ↔ <code>.</code> <code>knjiga/attribute::ID</code> ↔ <code>knjiga/@ID</code> | <p>potpuni oblik XPath sintakse</p> <p><code>os :: element</code></p> |
| <code>ancestor::*</code> <code>ancestor-or-self::*</code> | svi nadređeni čvorovi konteksta (uključujući ili ne uključujući i sam čvor konteksta) |
| <code>descendant::knjiga</code> <code>descendant-or-self::knjiga</code> | svi podređeni čvorovi konteksta koji su <i>knjiga</i> (ne uključujući ili uključujući i sam čvor konteksta ako je <i>knjiga</i>) |
| <code>preceding-sibling::*</code> <code>following-sibling::*</code> | svi prethodni/sljedeći čvorovi na istoj razini kao i čvor konteksta |
| <code>preceding::knjiga</code> <code>following::knjiga</code> | svi prethodni/sljedeći čvorovi od konteksta koji su <i>knjiga</i> |

Lokacijske staze - primjeri

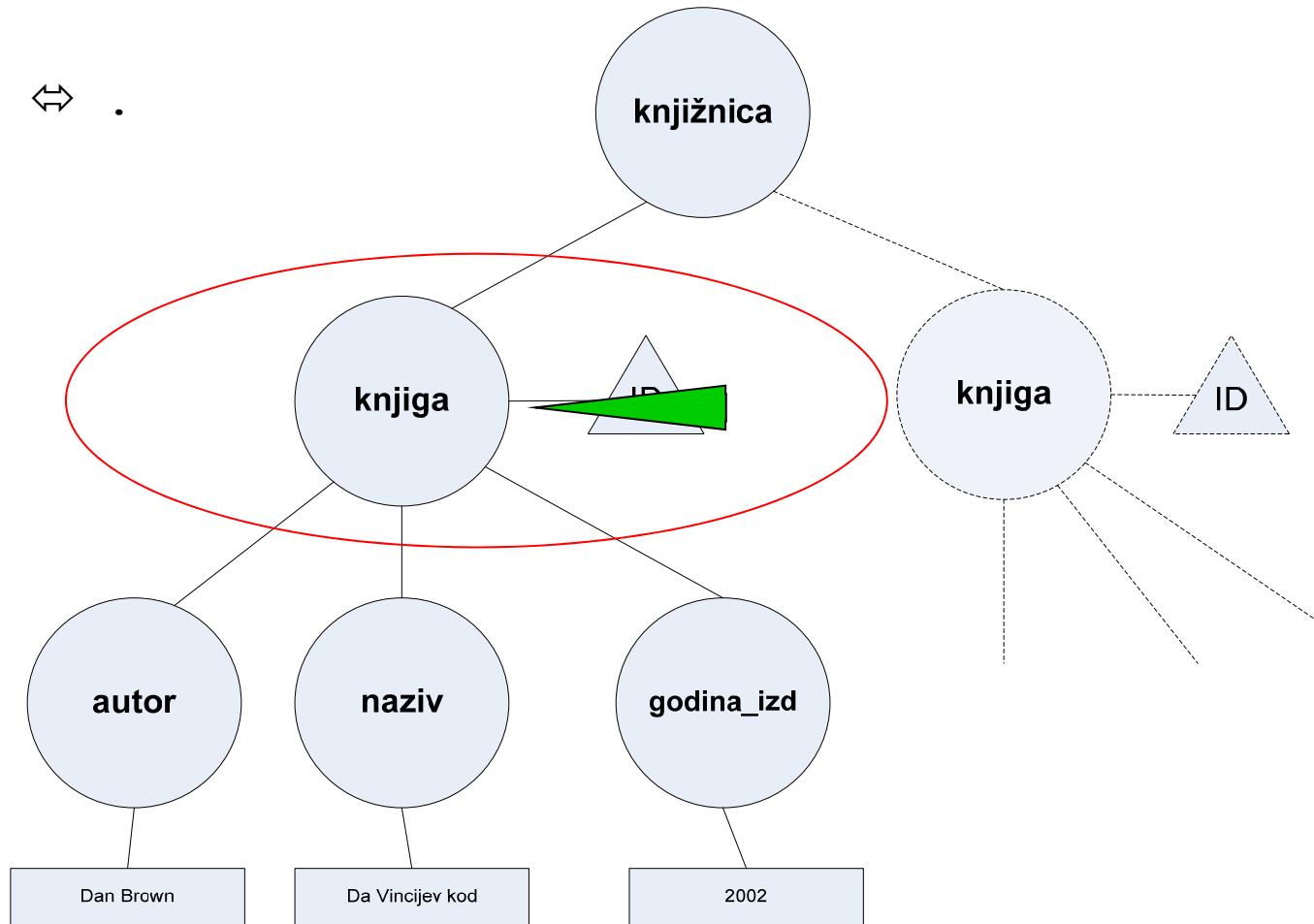
`child::knjiga` \Leftrightarrow `knjiga`



parent::knjiga



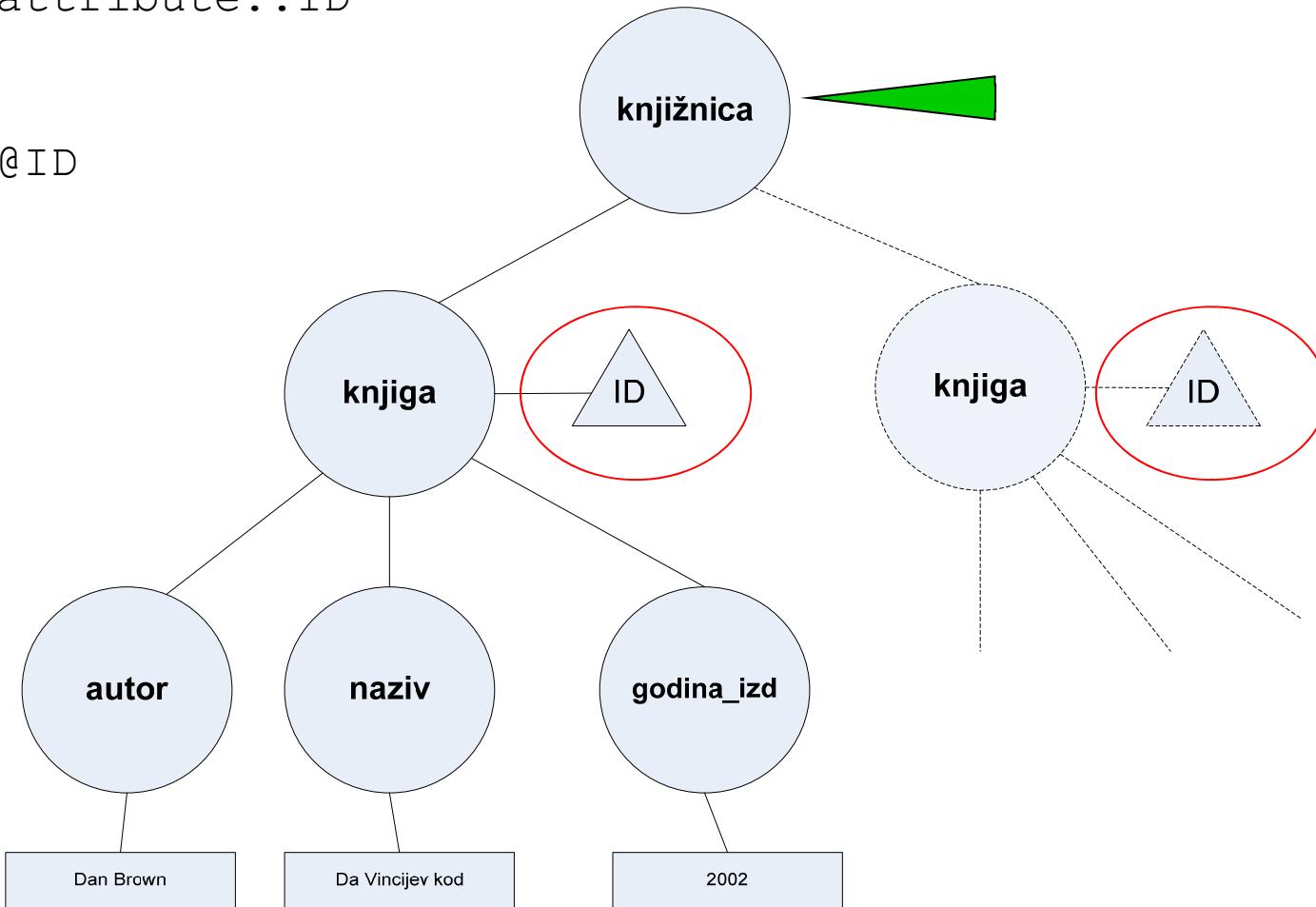
self::*: * ⇔ .



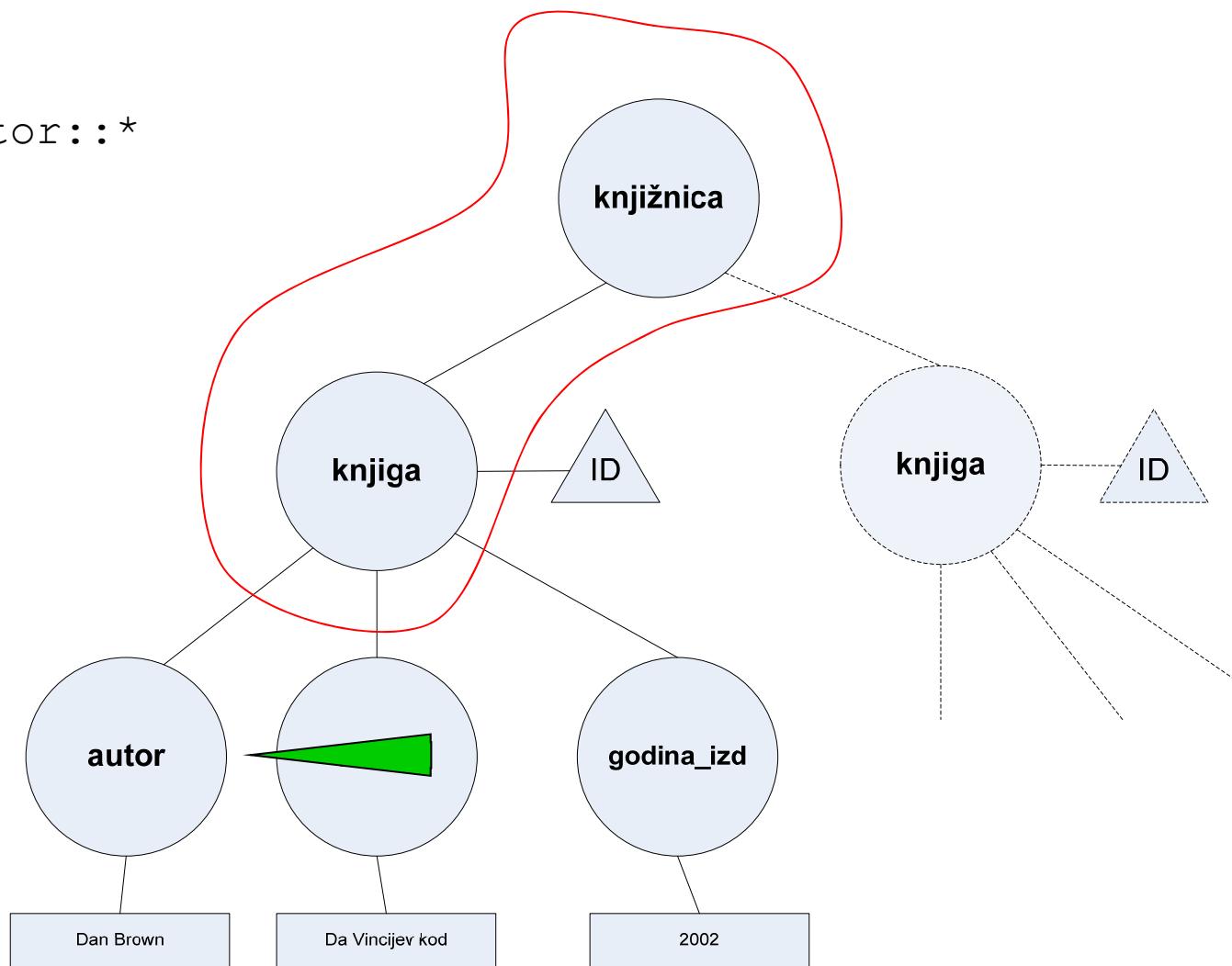
knjiga/attribute::ID

↔

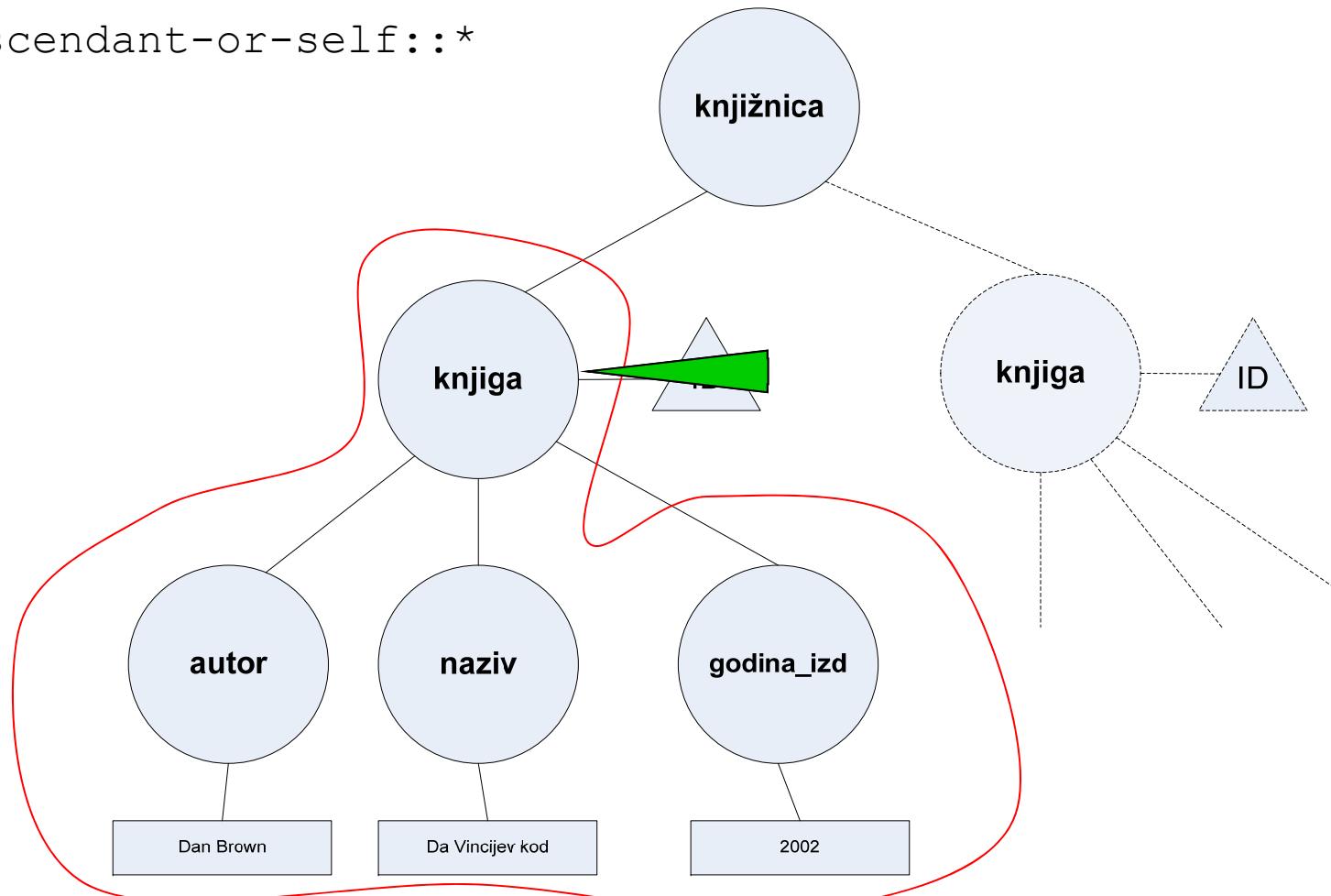
knjiga/@ID



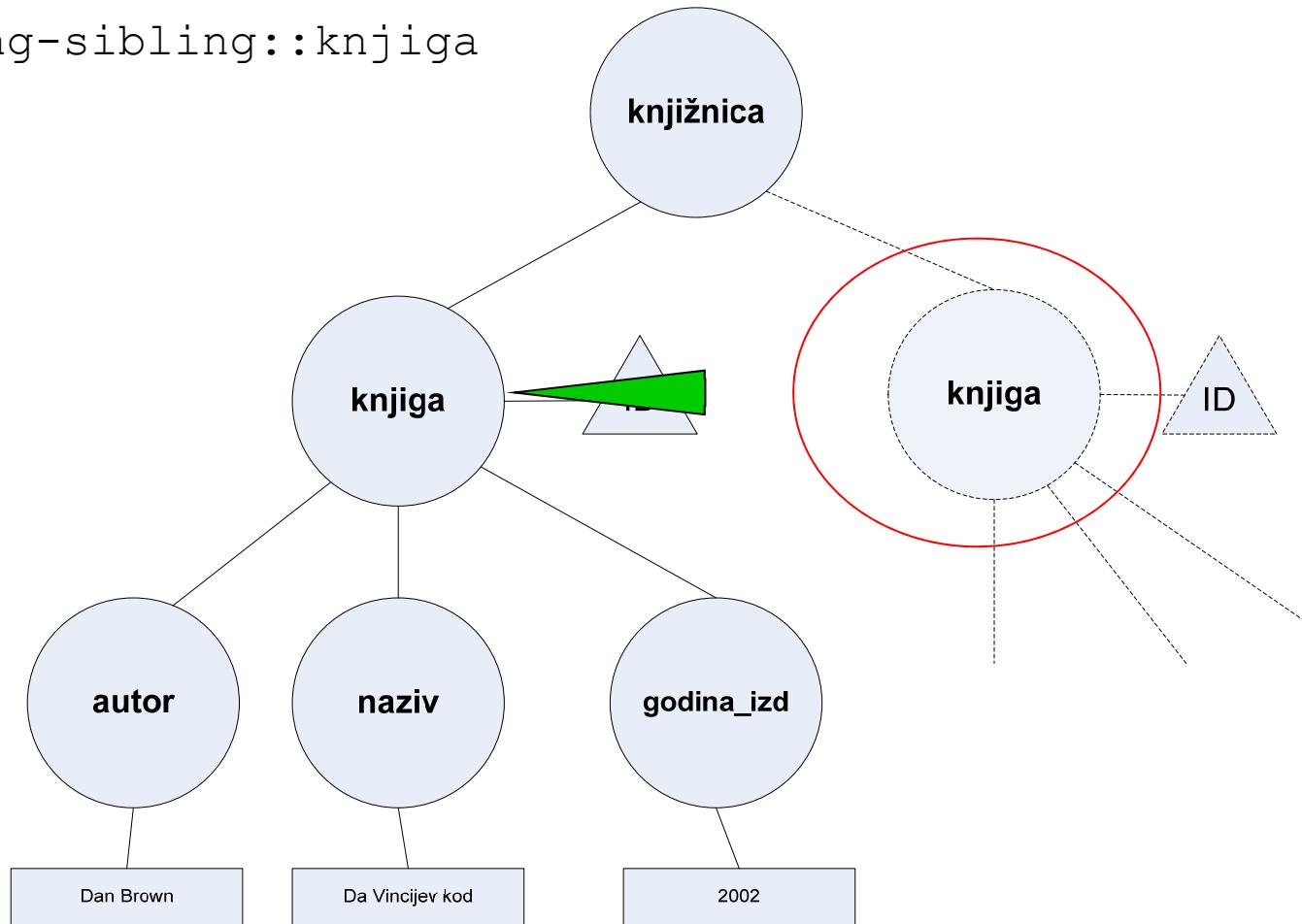
ancestor::*



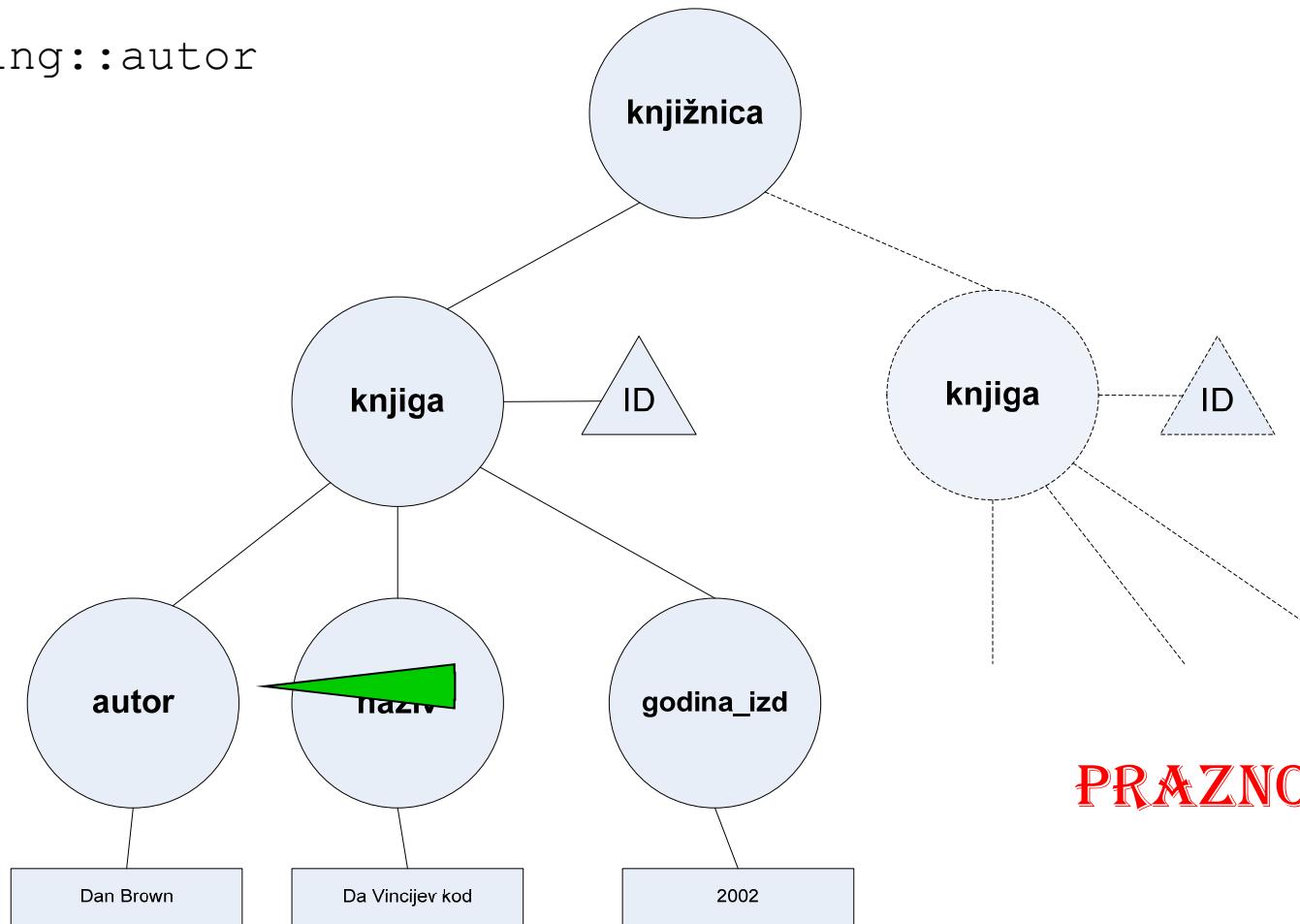
descendant-or-self::*



following-sibling::knjiga



preceding::autor



XPath predikati

- ◆ predikati se koriste za “filtriranje” rezultata dohvaćenih lokacijskom stazom
- ◆ predikat ima sljedeću sintaksu -
`element[izraz]`
- ◆ izraz u uglatim zagradama se evaluira u *Boolean* vrijednost - ako je istinita, čvor se dohvaća, a ako ne, čvor se izuzima iz rezultata
- ◆ izraz može biti funkcija, matematički ili logički izraz ili kombinacija

Lokacijske staze - predikati i funkcije

| | |
|---|---|
| <code>knjiga[3]</code> | dohvaća treći element <i>knjiga</i> (ako postoji) |
| <code>//knjiga/autor[position()=2 and @ID]</code> | svi elementi <i>autor</i> koji se pojavljuju drugi, a imaju u sebi atribut pod nazivom ID |
| <code>ancestor::knjiga[@ID="1001"]</code> | dohvaća sve pretke konteksta koji su <i>knjiga</i> i imaju atribut ID vrijednosti 1001 |
| <code>knjiga[(3 5) and @ID="1002"]</code> | knjiga koja je treća ili peta u kontekstu i ima ID vrijednosti 1002 |
| <code>count(//knjiga[@ID])</code> | broj svih elemenata <i>knjiga</i> u cijelom dokumentu koji imaju atribut pod nazivom ID |
| <code>//knjiga[last()]</code> | zadnji element <i>knjiga</i> bilo gdje u dokumentu |

XQuery

upitni jezik za XML

XQuery - osnovne značajke



- ◆ upitni jezik za rad s polustrukturiranim podacima u formatu XML
- ◆ razvijen od strane W3C
- ◆ deklarativni jezik s nekim elementima proceduralnih programskih jezika
- ◆ analogija sa SQL-om po načinu formiranja upita i osnovnoj logici korištenja

Relacije i XML (1)

- ♦ podaci o knjigama i njihovim autorima

BOOK

| bookKey | title | year | publisher | price |
|---------|--|------|----------------------------|--------|
| 1111 | TCP/IP Illustrated | 1994 | Addison-Wesley | 65.95 |
| 2222 | Advanced Programming in the Unix environment | 1992 | Addison-Wesley | 65.95 |
| 3333 | Data on the Web | 2000 | Morgan Kaufmann Publishers | 39.95 |
| 4444 | The Economics of Technology and Content for Digital TV | 1999 | Kluwer Academic Publishers | 129.95 |

| authorKey | last | first |
|-----------|-----------|-------|
| 50 | Stevens | W. |
| 60 | Abiteboul | Serge |
| 70 | Buneman | Peter |
| 80 | Suciu | Dan |

| bookKey | authorKey |
|---------|-----------|
| 1111 | 50 |
| 2222 | 50 |
| 3333 | 60 |
| 3333 | 70 |
| 3333 | 80 |

BOOK_AUTHOR

AUTHOR

- ◆ isti podaci u XML dokumentu

```
<bib>
  <book year="1994">
    <title>TCP/IP Illustrated</title>
    <author><last>Stevens</last><first>W.</first></author>
    <publisher>Addison-Wesley</publisher>
    <price>65.95</price>
  </book>
  <book year="1992">
    <title>Advanced Programming in the Unix environment</title>
    <author><last>Stevens</last><first>W.</first></author>
    <publisher>Addison-Wesley</publisher>
    <price>65.95</price>
  </book>
  <book year="2000">
    <title>Data on the Web</title>
    <author><last>Abiteboul</last><first>Serge</first></author>
    <author><last>Buneman</last><first>Peter</first></author>
    <author><last>Suciu</last><first>Dan</first></author>
    <publisher>Morgan Kaufmann Publishers</publisher>
    <price>39.95</price>
  </book>
</bib>
```

Upiti na relacije i upiti na XML

- ◆ ZADATAK: iz relacija ispisati:
 - za svakog autora prezime, ime i listu naslova koje je objavio
- ◆ upit:
 - `SELECT a.last, a.first, b.title
FROM author a, book b, bookauthor ba
WHERE a.authorkey=ba.authorkey AND
b.bookkey=ba.bookkey`
- ◆ ZADATAK: isto ispisati iz XML dokumenta u upitnom jeziku XQuery

Zahtjevi na upitni jezik za XML

- ◆ postavljati upite i nad meta podacima i nad korisničkim podacima
- ◆ postavljati upite nad duboko ugniježđenim strukturama
- ◆ tražiti objekte prema absolutnom i relativnom poretku
- ◆ očuvati poredak objekata u ulaznim dokumentima
- ◆ uspostaviti novi poredak na izlazu
- ◆ izvesti transformacije strukture
- ◆ očuvati postojeće hijerarhijske međuodnose

Standard i dokumenti

- ◆ **XQuery 1.0: An XML Query Language**
 - W3C Recommendation 23 January 2007
 - <http://www.w3.org/TR/2007/REC-xquery-20070123/>
 - zadnja verzija uvijek na <http://www.w3.org/TR/xquery>

- ◆ **XQuery XML Query Use Cases**
 - W3C Working Group Note 23 March 2007
 - <http://www.w3.org/TR/2007/NOTE-xquery-use-cases-20070323>
 - Zadnja verzija uvijek na
<http://www.w3.org/TR/xquery-use-cases>
 - **DOBRO PROUČITI primjere upita od Q1 do Q7 (poglavlje 1.1) !**

XQuery - model podataka

- ◆ XQuery percipira XML dokument kao **stablo čvorova**: 7 osnovnih tipova
 - element
 - atribut
 - tekstualni sadržaj
 - komentar
 - procesna instrukcija
 - imenik (*namespace*)
 - korijenski čvor dokumenta
- ◆ rad s cjelokupnim elementima, ali i dijelovima dokumenata

XQuery - ostale značajke

- ◆ koristi **XPath** za navigaciju u stablu
- ◆ podržava sve tipove podataka koje definira XML Schema
- ◆ podržava značajke SQL-a
 - agregacijske funkcije
 - sortiranje elemenata
- ◆ sama specifikacija **XQuery 1.0: An XML Query Language** NE omogućuje promjenu izvornog sadržaja
- ◆ **XQuery Update Facility 1.0** je od 17. ožujka 2011. *W3C Recommendation*

<http://www.w3.org/TR/xquery-update-10/>

- omogućava dodavanje i brisanje čvorova te izmjene unutar čvorova u stablu čvorova kojim je predstavljen XML dokument

Osnovni izgled upita: FWLOR (1)

- ◆ FLWOR (čita se kao *flower*)
 - FOR
 - LET
 - WHERE
 - ORDER BY
 - RETURN
- ◆ analogija sa SQL-om: SELECT-FROM-WHERE-ORDER BY

Osnovni izgled upita: FWLOR (2)

- ◆ FLWOR: FOR - LET - WHERE - ORDER BY - RETURN
- ◆ WHERE i ORDER BY: jednako kao u SQL-u
 - WHERE - odabir nekih čvorova u većem skupu čvorova (filtriranje liste čvorova)
 - ORDER BY - poredak čvorova
- ◆ FOR
 - veže vrijednosti elemenata za određenu varijablu
 - varijabla predstavlja listu vrijednosti nad kojima se vrše iteracije

Osnovni izgled upita: FWLOR (3)

- ◆ LET
 - povezuje vrijednosti elemenata u listu, ali bez iteracija
 - također predstavlja definiciju variable

- ◆ RETURN
 - oblikuje rezultat
 - SQL: rezultat upita nad relacijama je relacija
 - XQuery: rezultat može biti XML dokument ili pojedinačna vrijednost
 - moguće je konstruirati strukture posve različite od izvornih

Radni dokument i njegov DTD

```
<bib>
  <book year="1994">
    <title>TCP/IP Illustrated</title>
    <author><last>Stevens</last><first>W.</first></author>
    <publisher>Addison-Wesley</publisher>
    <price>65.95</price>
  </book>
  <book year="1992">
    <title>Advanced Programming in the Unix environment</title>
    <author><last>Stevens</last><first>W.</first></author>
    <publisher>Addison-Wesley</publisher>
    <price>65.95</price>
  </book>
  <book year="2000">
    <title>Data on the Web</title>
    <author><last>Abiteboul</last><first>Serge</first></author>
    <author><last>Buneman</last><first>Peter</first></author>
    <author><last>Suciu</last><first>Dan</first></author>
    <publisher>Morgan Kaufmann Publishers</publisher>
    <price>39.95</price>
  </book>
  <book year="1999">
    <title>The Economics of Technology and Content for Digital TV</title>
    <editor>
      <last>Gerbarg</last><first>Darcy</first>
      <affiliation>CITI</affiliation>
    </editor>
    <publisher>Kluwer Academic Publishers</publisher>
    <price>129.95</price>
  </book>
</bib>
```

```
<!ELEMENT bib (book* )>
<!ELEMENT book (title, (author+ | editor+ ), publisher, price )>
<!ATTLIST book year CDATA #REQUIRED >
<!ELEMENT author (last, first )>
<!ELEMENT editor (last, first, affiliation )>
<!ELEMENT title (#PCDATA )>
<!ELEMENT last (#PCDATA )>
<!ELEMENT first (#PCDATA )>
<!ELEMENT affiliation (#PCDATA )>
<!ELEMENT publisher (#PCDATA )>
<!ELEMENT price (#PCDATA )>
```

Jednostavan upit FLWOR (1)

- ◆ QXuery Use Case XMP 1:
 - za knjige izdane nakon 1991, a čiji je izdavač Addison-Wesley ispiši godinu i naslov:

```
<bib>
{
    for $b in doc("data/xmp-data.xml")/bib/book
    where $b/publisher = "Addison-Wesley"
    and $b/@year > 1991
    return
        <book year="{ $b/@year }">
            { $b/title }
        </book>
}
</bib>
```

Jednostavan upit FLWOR (2)

```
<bib>
{
    for $b in doc("data/xmp-data.xml")/bib/book
        where $b/publisher = "Addison-Wesley"
        and $b/@year > 1991
    return
        <book year="{ $b/@year }">
            { $b/title }
        </book>
}
</bib>
```

```
<bib>
    <book year="1994">
        <title>TCP/IP Illustrated</title>
    </book>
    <book year="1992">
        <title>Advanced Programming in the Unix environment</title>
    </book>
</bib>
```

Jednostavan upit FLWOR (3)

```
<bib>
{
  for $b in doc("data/xmp-data.xml")/bib/book
  where $b/publisher = "Addison-Wesley"
  and $b/@year > 1991
  return
    <book year="{ $b/@year }">
      { $b/title }
    </book>
}
</bib>
```

◆ **doc("data/xmp-data.xml")/bib/book**

datoteka ili URL (tip čvora:
korijen dokumenta)

Elementi bib i book

datoteka:

- apsolutna staza (C:/Documents and Settings/Pero/My Documents/QXquery/data/xmp-data.xml)
- relativna staza (data/xmp-data.xml): u odnosu na datoteku u kojoj je XQuery

Jednostavan upit FLWOR (4)

```
<bib>
{
  for $b in doc("data/xmp-data.xml")/bib/book
    where $b/publisher = "Addison-Wesley"
      and $b/@year > 1991
    return
      <book year="{ $b/@year }">
        { $b/title }
      </book>
}
</bib>
```

♦ **for \$b in doc("data/xmp-data.xml")/bib/book**

značenje izraza:

- u svim vršnim elementima **bib** (zapravo postoji samo jedan **bib** jer je on dijete nevidljivog korijena) pronađi sve podelemente **book** i spremi ih u varijablu **\$b**
- izvrši iteraciju za sve **\$b**

Jednostavan upit FLWOR (5)

```
<bib>
{
    for $b in doc("data/xmp-data.xml")/bib/book
    where $b/publisher = "Addison-Wesley"
    and $b/@year > 1991
    return
        <book year="{ $b/@year }">
            { $b/title }
        </book>
}
</bib>
```

- ◆ **for \$b in**
- where \$b/publisher = "Addison-Wesley" and \$b/@year > 1991**

značenje izraza WHERE:

- među svim pronađenim čvorovima **book** u **\$b** spremi samo one čiji podelement **publisher** ima vrijednost Addison-Wesley, a atribut **year** vrijednost veću od 1991

Jednostavan upit FLWOR (6)

```
<bib>
{
  for $b in doc("data/xmp-data.xml")/bib/book
  where $b/publisher = "Addison-Wesley"
  and $b/@year > 1991
  return
    <book year="{ $b/@year }">
      { $b/title }
    </book>
}
</bib>
```

- ♦ **for \$b in doc("data/xmp-data.xml")/bib/book**
return...

značenje izraza RETURN:

- za svaku pojedinu iteraciju **\$b** vrati XML strukturu određenu onime što se nalazi iza RETURN

Jednostavan upit FLWOR (7)

```
<bib>
{
    for $b in doc("data/xmp-data.xml")/bib/book
    where $b/publisher = "Addison-Wesley"
    and $b/@year > 1991
    return
        <book year="{ $b/@year }">
            { $b/title }
        </book>
}
</bib>
```

- ◆ izlaz ovog upita u XQueryju zapravo je jedan jedini element **bib**
- ◆ unutar vitičastih zagrada {} se nalazi XQuery kod
- ◆ cijelokupna iteracija vrši se unutar tog elementa **bib**
- ◆ povratne vrijednosti **bib** i **book** podudaraju se s elementima u XML dokumentu, ali nisu nužne
 - cijela povratna struktura, imena njenih elemenata mogu se proizvoljno definirati

Povratna struktura

```

<bib> → drugo ime: rezultat
{
    for $b in doc("data/xmp-data.xml")/bib/book
    where $b/publisher = "Addison-Wesley"
    and $b/@year > 1991
    return
        <book year="{ $b/@year }">
            { $b/title }
        </book>
    }
</bib>
  
```

drugo ime: rezultat

drugo ime: godina

drugo ime: naslov (umjesto title)

drugo ime: knjiga

```

<bib>
    <book year="1994">
        <title>TCP/IP Illustrated</title>
    </book>
    <book year="1992">
        <title>Advanced Programming in the Unix environment</title>
    </book>
</bib>
  
```

Ponovno upit i odgovor

```
<rezultat>
{
    for $b in doc("data/xmp-data.xml")/bib/book
    where $b/publisher = "Addison-Wesley" and
    $b/@year > 1991
    return
        <knjiga godina="{$b/@year}">
            <naslov>{$b/title/text()}</naslov>
        </knjiga>
}
</rezultat>
```

atribut je uvijek par ime-vrijednost:
ne postoji posebna oznaka za
vrijednost atributa

`text()`: označava tekstualni čvor koji
predstavlja sadržaj elementa `title`

```
<rezultat>
    <knjiga godina="1994">
        <naslov>TCP/IP Illustrated</naslov>
    </knjiga>
    <knjiga godina="1992">
        <naslov>Advanced Programming in the Unix environment</naslov>
    </knjiga>
</rezultat>
```

Dohvaćanje elementa relativnim izrazom u XPathu

```
<bib>
{
    for $b in doc("data/xmp-data.xml")//book
        where $b/publisher = "Addison-Wesley"
        and $b/@year > 1991
    return
        <book year="{ $b/@year }">
            { $b/title }
        </book>
}
</bib>
```

relativno se dohvaća **book**:
svaki podelement book bez
obzira na položaj u stablu XML
dokumenta

```
<bib>
    <book year="1994">
        <title>TCP/IP Illustrated</title>
    </book>
    <book year="1992">
        <title>Advanced Programming in the Unix environment</title>
    </book>
</bib>
```

Odgovor je isti;
razlika bi postojala kad bi u
dokumentu postojali elementi
book koji nisu izravni
potčvorovi globalnog elementa
bib: tada bi i oni bili
dohvaćeni ovim upitom

Pohrana XML-a

Upravljanje podacima

2015./2016.

Pohrana podataka u XML formatu

1. Pohrana XML dokumenata u datotečni sustav
2. Pohrana XML dokumenata u velike objekte (Large OBject – LOB) u objektno-relacijskim bazama
3. Preslikavanje strukture XML podataka u relacijsku strukturu
4. Korištenje izvornih (“native”) sustava za upravljanje XML podataka

Pohrana u datotečni sustav

- ◆ Prednost: jednostavnost
- ◆ Nedostaci:
 - XML dokument **treba parsirati** svaki put kad mu se pristupa što oduzima vrijeme pri obradi upita i provjeri valjanosti dokumenata
 - sustavi za upravljanje bazama podataka nude veće mogućnosti pri upravljanju podacima (upitni jezici, provjera integriteta, sigurnost, upravljanje transakcijama...) nego datotečni sustavi

Pohrana u velike objekte (LOB)

- ◆ Objektno-relacijskim sustavima za upravljanje bazama podataka dodaju se nove mogućnosti radi što jednostavnijeg pohranjivanja XML podataka.
- ◆ Različiti proizvođači (Oracle, Microsoft, IBM, Sybase, ...) nude različita rješenja, ali kod svih njih je potpora za XML samo nadogradnja postojećih načina pohrane podataka.
- ◆ Da bi se omogućila pohrana XML-a u objektno-relacijske sisteme, uvedeni su novi tipovi podataka i proširen je jezik SQL. Tzv. veliki objekti (LOB – Large OBject) su prilagođeni za pohranu XML dokumenata ili njihovih dijelova.

Pohrana u velike objekte (LOB)

Sadržaj dokumenta se može spremiti **u jedan stupac tablice** koristeći tipove podataka CLOB (Character Large OBject) ili BLOB (Binary Large OBject), odnosno tipove podataka koji se temelje na CLOB-u ili BLOB-u.

Ako se radi pohrani na temelju CLOB-a:

- dokument se čuva kao tekstualna datoteka
- ovisno o implementaciji, za dohvata podataka mogu se koristiti:
 - upitni jezici za XML (**XQuery**, **XPath**)
 - **SQL/XML** - proširenje SQL-a za XML
 - metode za **pretraživanje teksta**

Preslikavanje u relacijsku strukturu

- ◆ relacijski sustavi – dobro poznata, zrela i raširena tehnologija
- ◆ ideja: iskoristiti relacijske sustave za pohranu XML-a
- ◆ problem:
 - XML – polustrukturirani podaci, mogu imati nepravilnu strukturu
 - relacije (tablice) uvijek imaju pravilnu strukturu
 - **relacijski model i XML nekompatibilni**
 - nema podudarnosti između elemenata i atributa u XML-u i relacija i atributa u relacijskoj bazi
 - **atribut iz relacijskog modela može u XML dokumentu biti predstavljen ili elementom ili atributom**

Preslikavanje u relacijsku strukturu

Glavni proizvođači relacijskih i objektno-relacijskih sustava za upravljanje bazom podataka (Oracle, Microsoft, IBM, Sybase...) pružaju potporu za preslikavanje XML podataka u relacijski format.

Dva osnovna pristupa kod preslikavanja (*mapiranja*):

- ◆ **SUBP automatski** radi preslikavanje na osnovu XML Scheme. XML elementi i atributi se preslikavaju u stupce tablica. Takvo mapiranje korisnik najčešće može promijeniti.
- ◆ **Korisnicima** se omogućuje da **sami** definiraju preslikavanje.

Detaljnije na:

<http://www.rpbourret.com/xml/ProdsXMLEnabled.htm>

Korištenje izvornog sustava za XML

Izvorni (eng. *native*) **sustavi za upravljanje XML bazom podataka** su prvenstveno namijenjeni pohrani XML dokumenata.

Često se u praksi koristi kraći naziv **IZVORNA XML BAZA PODATAKA** (eng. *native XML database*).

- ◆ Pohranjuju i obrađuju podatke u XML formatu, bez mapiranja i prebacivanja podataka u relacijsku ili neku drugu strukturu.
- ◆ Svojstva izvornih XML baza podataka:
 - **Osnovna logička jedinica pohrane je XML dokument.**
 - **Imaju definiran logički model za XML dokumente. Taj model mora uključivati barem elemente, atribute, tekst i poređak elemenata.**

Korištenje izvorne XML baze podataka

- ◆ Kao i drugi DBMS-ovi i izvorni XML DBMS-ovi podržavaju upitne jezike, sigurnost, API-je, upravljanje transakcijama itd.
- ◆ Većina proizvoda podržava velik broj W3C specifikacija za XML, kao što su XPath, XQuery te XSLT.
- ◆ Omogućavaju veću brzinu obrade podataka nego što je to slučaj kod mapiranja u relacijsku bazu podataka, jer su podaci kod izvornih baza smješteni na jednom mjestu na disku.
- ◆ Omogućavaju fleksibilnu pohranu i dohvata podataka.

Korištenje izvorne XML baze podataka

- ◆ Primjeri sustava:
 - **BaseX** (open source) <http://basex.org/>
 - razvoj sustava počeo na Sveučilištu Konstanz, Njemačka
 - podržava standardni jezik **XQuery** te **XQuery Update Facility** (W3C preporuka od 2011.)
 - **eXist** (open source) <http://exist.sourceforge.net/>
 - podržava XQuery te proširenja jezika XQuery za izmjenu podataka
 - **Tamino** (Software AG)
 - podržava upitne jezike: Tamino X-Query (koji koristi XPath) i W3C XQuery<http://www.softwareag.com/corporate/products/wm/tamino/>

Korištenje izvorne XML baze podataka

- ◆ Primjeri sustava (nastavak):
 - **Oracle Berkeley DB XML** (open source) – ranije Sleepycat Software
 - podržava XQuery te XQuery Update Facility na razini čvora

<http://www.oracle.com/technology/products/berkeley-db/index.html>
 - **Ipedo XML Database** (Ipedo) http://www.ipedo.com/html/ipedo_xml_db.html
 - podržava XQuery
 - **Apache Xindice** (open source) <http://xml.apache.org/xindice/>
 - koristi XPath i XUpdate (nije W3C standardni jezik)
 - razvijan do 2007. godine

Odabir načina pohrane XML-a

Dva tipa XML datoteka:

1. **datoteke pravilne strukture, s podatkovnim jedinicama manjeg formata** (nema velikih “znakovnih nizova”), datoteke namijenjene prvenstveno prijenosu podataka, kod kojih je sadržaj je važniji od strukture dokumenta. Primjer: narudžbe.
(eng. *data-centric XML files*)
 - preslikavanje u relacijsku strukturu
 - zrela i stabilna tehnologija, ušteda prostora
 - preslikavanje može biti složeno i zahtijevati dosta vremena
 - ne može se iz dobivenih relacija opet dobiti identičan XML dokument
 - pohrana u izvorne XML baze ako je XML “prirodni format” dokumenata (npr. poruke kod elektroničke trgovine)

Odabir načina pohrane XML-a

2. datoteke nepravilne strukture, krupnijih podatkovnih jedinica.

Primjer: knjige.

(eng. *document-centric* XML files)

- pohrana u datotečne sustave ili velike objekte u objektno-relacijskim sustavima
 - (za manje skupove dokumenata koji ne zahtijevaju česte izmjene)
- pohrana u izvorne XML baze
 - brzi dohvati podataka u izvornim bazama
- veće organizacije, s velikim investicijama u objektno-relacijske sustave, vjerojatno će radije koristiti nadodane funkcije za XML koje nude vodeći proizvođači tih sustava

Upravljanje podacima

SKLADIŠTENJE PODATAKA (1.dio)

sastavili:

izv. prof. dr. sc. Boris Vrdoljak

doc. dr. sc. Marko Banek

Teme

- ◆ Uvod u skladištenje podataka
- ◆ Definicije
- ◆ Svojstva skladišta podataka
- ◆ Struktura podataka u skladištu
- ◆ Procesi u skladištenju podataka
 - Izvlačenje transformacija i učitavanje podataka
 - Spremanje podataka i upravljanje podacima
 - Korištenje skladišta podataka

UVOD U SKLADIŠTENJE PODATAKA

Definicija baze podataka (ponavljanje)

BAZA PODATAKA je centralno mjesto informacijskog sustava. Pohranjeni podaci u bazi podataka **opisuju trenutno stanje** dijela realnog svijeta za koji je i razvijen informacijski sustav, naravno na način pogodan za računalnu obradu.

BAZA PODATAKA je skup međusobno povezanih podataka pohranjenih bez nepotrebne zalihosti s ciljem da na optimalni način posluže u raznim primjenama. Podaci se spremaju neovisno o programima koji ih koriste, zajedničkim pristupom dodaju se novi podaci te mijenjaju i premještaju postojeći.

Uvod u skladištenje podataka

Po osnovnoj definiciji **baza podataka** prikazuje **trenutno stanje** informacijskog sustava.

U toj definiciji se zapravo radi o tzv. **operacijskim ili transakcijskim bazama podataka** koje prikazuju trenutno stanje procesa kojeg opisuju.

Uvod u skladištenje podataka

Uvijek su se, naročito **radi donošenja strateških odluka**, tražili i odgovarajući **povijesni podaci** koji su se dobivali na različite načine, često posebnim vještinama administratora baze podataka i programera.

Također treba imati na umu da postoje **različite vrste korisnika**. Svaka vrsta korisnika ima **različite potrebe za podacima i različite načine korištenja informacijskih sustava**.

Uvod u skladištenje podataka - primjer

Koje bi podatke trebao sadržavati informacijski sustav?

primjer: **banka**

- trenutna stanja svih računa svih klijenata DA
- svi aktivni krediti svih klijenata DA
- jučerašnja stanja računa svih klijenata vjerojatno DA
- stanja računa svih klijenata prije 90 dana vjerojatno DA
- stanja od prije 17 godina za račune koji su u međuvremenu zatvoreni
trošenje memorijskih resursa: je li to uistinu potrebno???
- ◆ **konkretna IZVEDBA** ovisi o **POTREBAMA KORISNIKA** sustava
odnosno **PODRUČJU PRIMJENE** tog sustava

Uvod u skladištenje podataka - primjer

- ◆ Različite potrebe korisnika
- ◆ primjer: banka
 - osobni bankari
 - vidjeti trenutna stanja svih računa klijenata
 - vidjeti podatke o svim aktivnim kreditima klijenata
 - menadžeri
 - usporediti zaradu banke od namjenskih i nemajenskih kredita
 - saznati u koje doba godine najviše ljudi uzima kredit za automobil
 - saznati je li korisnički paket *3usluge-40kn* uspješan ili je bolje svaku uslugu plasirati zasebno po 20kn po usluzi
 - analitičari kreditnih rizika
 - vidjeti podatke o svim kreditima svih klijenata kroz dulje vremensko razdoblje i vršiti statističke analize

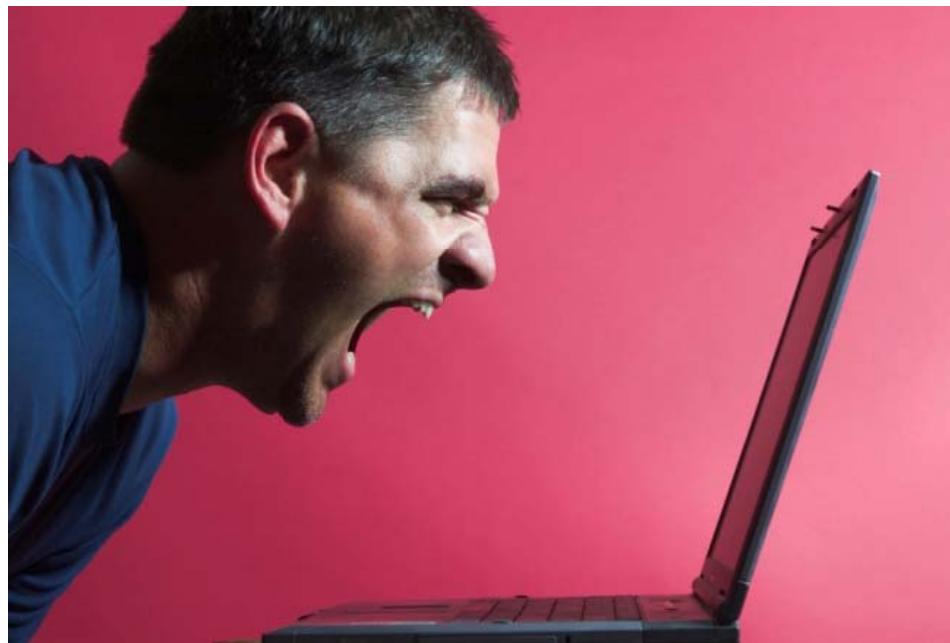
Uvod u skladištenje podataka - primjer

Različite potrebe korisnika u banci:

- osobni bankari
 - relativno strogo definiran skup operacija (uplate, isplate, odobrenje kredita)
 - obujmom najmanji skup podataka (većinom trenutni podaci)
- menadžeri
 - široko područje mogućeg interesa
 - skup više upita; struktura kasnijih upita ovisna o rezultatima ranijih
 - očekivan skup podataka širi; uključuje povijesne (arhivirane) podatke
- analitičar kreditnih rizika
 - skup više upita; struktura kasnijih upita ovisna o rezultatima ranijih
 - upotreba statističkih alata – još širi skup podataka

Analitička obrada podataka - problem

- Analitičari i menadžeri često ne mogu **brzo i jednostavno pristupiti postojećim podacima** u transakcijskim bazama podataka i pri tome **dobiti točne i potpune podatke**



Analitička obrada podataka - problem

- ◆ Podaci unutar jednog poduzeća pohranjeni:
 - na raznorodnim platformama
 - u različitim tipovima baza podataka i datoteka
 - u različitim podatkovnim formatima
 - s nekonzistentnim ključevima, mjernim jedinicama...
- ◆ za analizu su zanimljivi **sumarni i povijesni podaci**, analitički upiti obuhvaćaju veliku količinu podataka
- ◆ **analitička obrada narušava učinkovitost rada transakcijskih (operacijskih) baza**

Analitička obrada podataka - problem

Transakcijske baze omogućavaju dobivanje podataka o pojedinim poslovnim funkcijama, ali ne mogu dati u odgovarajućem vremenu, a da ne **naruše performanse sustava**, relevantne podatke za cjelovitu sliku poslovanja za koju su bitni i povijesni podaci.

Transakcijska baza opisuje trenutno stanje sustava i mora biti uvijek **dostupna**, a **složeni upiti ruše njene performanse**.

Rješenje je u implementaciji **skladišta podataka**.

Transakcijska baza i skladište podataka



U suvremenom informacijskom sustavu velikih poslovnih subjekata - dvije komponente:

■ **transakcijski sustav**

- još se koriste i nazivi: operacijski sustav, OLTP sustav

(OLTP - *On-Line Transaction Processing*)

■ **sustav skladištenja podataka**

- Da bi se što više smanjio negativan utjecaj analitičke obrade na rad transakcijskih sustava, a i zbog brojnih razlika tih dvaju sustava, nužno ih je **logički i fizički odvojiti**.

Transakcijska baza i skladište podataka

- ◆ Za **transakcijsku** (operacijsku) bazu podataka bitan je element **transakcija**. Sustav za upravljanje bazom podataka mora obaviti **veliki broj transakcija** od kojih svaka sadržava **malu količinu podataka**.
- ◆ Kod **sustava skladištenja podataka** težište je na **analizi** podataka i izradi izvješća pri čemu se obraduju **velike količine podataka**.

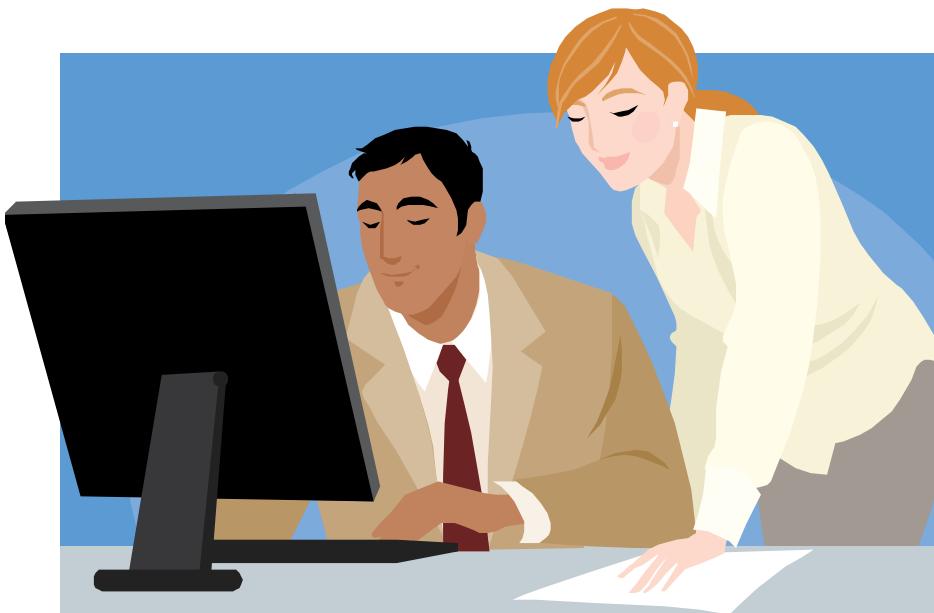
Cilj skladištenja podataka



- ◆ Osnovni cilj
skladištenja
podataka
NIJE
pohraniti velike
količine starih
podataka

Cilj skladištenja podataka

- ◆ što bolje i što brže iskoristiti postojeće podatke za dobivanje korisnih informacija
- ◆ analitičarima, menadžerima, donositeljima odluka omogućiti fleksibilan i učinkovit pristup svim relevantnim podacima



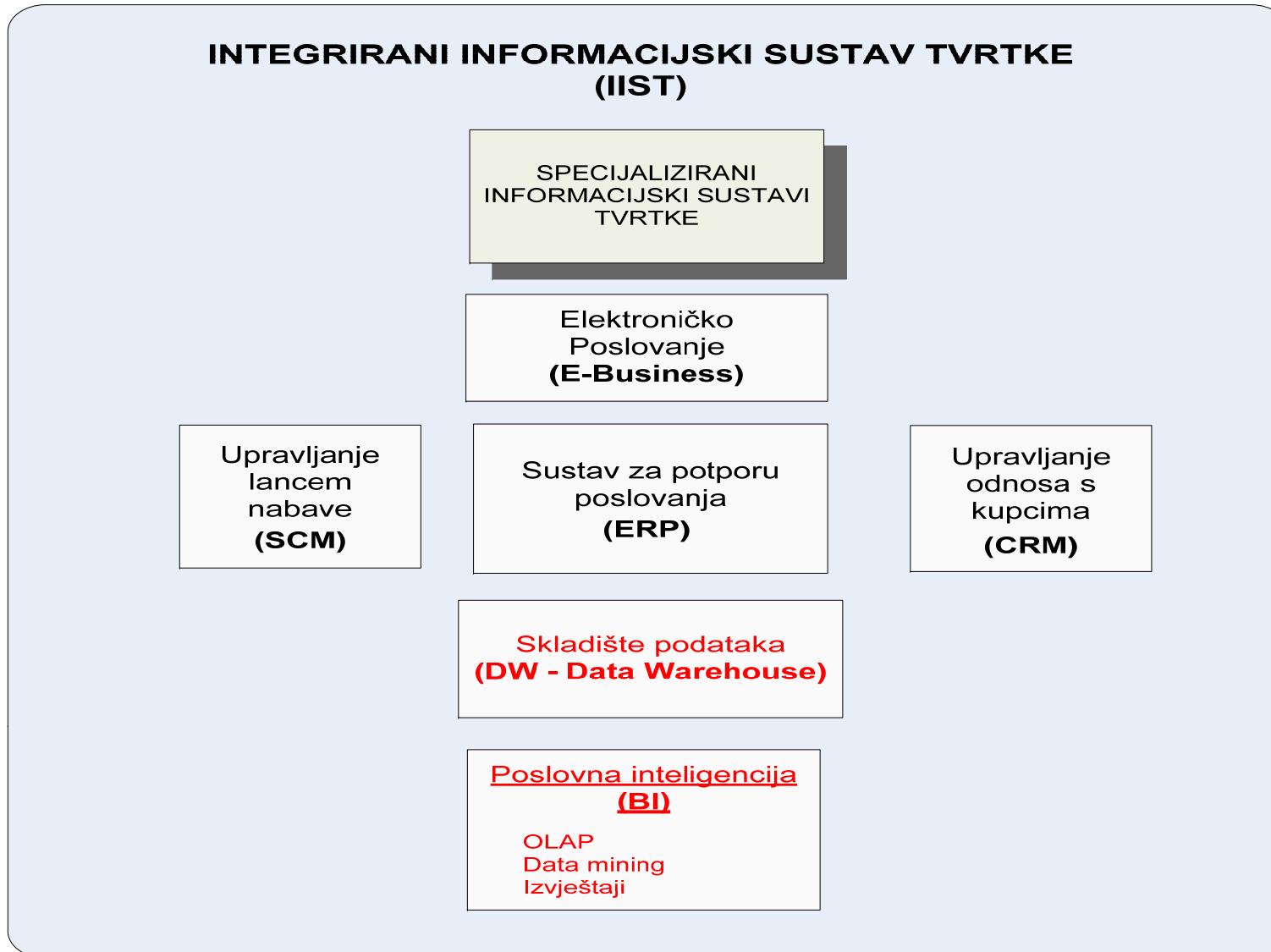
Cilj skladištenja podataka

- ◆ omogućiti dobivanje cijelovite slike o poslovanju poduzeća
- ◆ analizirati i razumijeti poslovna kretanja, pratiti i predviđati situaciju na tržištu, što bolje isplanirati sljedeće korake poslovanja, što brže reagirati na promjene, poboljšati odnose s poslovnim partnerima i kupcima

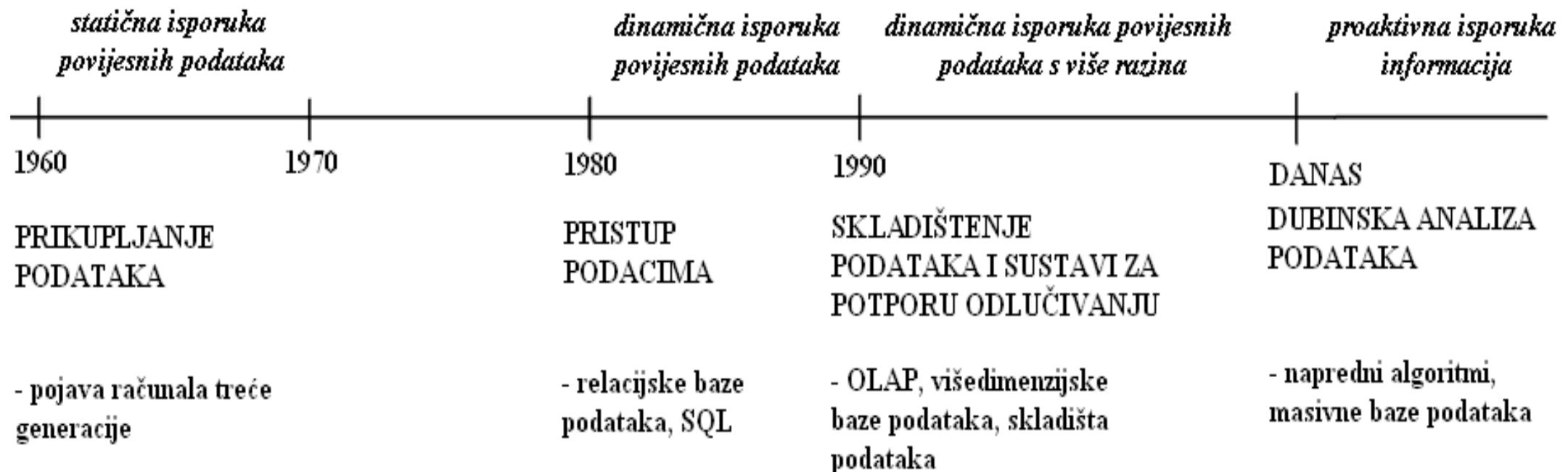


- ◆ Kvaliteta odluka ovisi o kvaliteti dostupnih podataka, razumijevanju tih podataka i brzini njihova dohvata.

Integrirani informacijski sustav tvrtke



Povijesni razvoj



Skladište podataka

DEFINICIJE

Skladište podataka - definicija



SKLADIŠTE PODATAKA je baza podataka koja **sadrži povijesne nepromjenjive podatke** koji se prikupljaju i obrađuju radi **potpore poslovnom odlučivanju**. Podaci se izvlače iz raznovrsnih izvora te se u skladu s definiranim modelom podataka učitavaju u skladište podataka i integriraju s postojećim podacima.

Skladište podataka

- u logičkom smislu centralizirana baza podataka
- sadržaj: povijesni nepromjenjivi podaci koji se prikupljaju i obrađuju radi potpore poslovnom odlučivanju
- podaci se izvlače iz raznovrsnih izvora te se u skladu s definiranim modelom podataka učitavaju u skladište podataka i integriraju s postojećim podacima (podaci učitani iz različitih izvora trebaju biti konzistentni)

Skladište podataka

- izvor podataka: transakcijske (operacijske) baze podataka, datoteke, vanjski izvori, novi sustavi razmjene informacija (elektroničko poslovanje: B2B, B2C, web-usluge)
- periodički se osvježava
- omogućava izvlačenje korisnih i sažetih informacija iz ogromne količine podataka pohranjene informacijskim sustavima

Skladištenje podataka - definicija

SKLADIŠTENJE PODATAKA (*data warehousing*) je proces integracije podataka o poslovanju neke organizacije u jednu bazu podataka (u logičkom smislu), iz koje krajnji korisnici - analitičari, menadžeri, donositelji poslovnih odluka - mogu raditi izvješća, postavljati upite i analizirati podatke.

Sustav skladištenja podataka - definicija

SUSTAV SKLADIŠTENJA PODATAKA

(*Data Warehousing System*) je skup tehnologija i alata koji omogućavaju **prikupljanje, integraciju i fleksibilnu analizu podataka** dobivenih iz različitih izvora, s ciljem iskorištenja postojećih podataka **za brzo dobivanje korisnih informacija**.

Područno skladište podataka

Skladište podataka često sadrži manje skupove podataka koje nazivamo **područnim skladištima podataka** (eng. *data mart*).

- ◆ samo **jedno tematsko područje**, odnosno **homogena grupa korisnika**
- ◆ pojedine funkcije poslovanja, male usko specijalizirane poslovne grupe, poslovanje pojedinih odjela poduzeća
- ◆ u poduzeću - jedno skladište podataka (*data warehouse*) i više područnih skladišta podataka (*data mart*)

Područno skladište podataka – definicija

PODRUČNO SKLADIŠTE PODATAKA je skup podataka dizajniran i konstruiran radi potpore odlučivanju pri čijem se dizajniranju slijede principi dizajna skladišta podataka s tim da taj skup podataka poslužuje potrebe **homogene grupe korisnika**.

Područno skladište podataka

Skladišta podataka donose poslovnu inteligenciju koja obuhvaća **velik broj funkcija poslovanja** nekog poduzeća, velik broj poslovnih grupa i odjela poduzeća.

Za razliku od toga, **područno skladište podataka (data mart)** orijentirano je na **pojedine funkcije poslovanja**, na male usko **specijalizirane poslovne grupe**, odnosno na **posovanje pojedinih odjela** poduzeća kao što su npr. prodaja, marketing i sl.

Organizacija će obično imati jedno skladište podataka i više područnih skladišta podataka, izgrađenih prema potrebama pojedinih odjela.

Područno skladište podataka

- ◆ Područno skladište podataka može biti:
 - **ovisno**
 - izvor podataka mu je središnje skladište podataka
 - **neovisno**
 - uzima podatke izravno iz operacijskih sustava

Svojstva skladišta podataka

- ◆ Skladište podataka
 - središnji dio sustava skladištenja podataka
 - Osnovna svojstva:
 1. Tematska orijentacija
 2. Integriranost
 3. Skup vremenskih snimaka
 4. Nepromjenjivost podataka

SVOJSTVA SKLADIŠTA PODATAKA

Svojstva skladišta podataka

Tematska orijentacija

subject-oriented

- ◆ Skladište podataka je orijentirano na **glavna tematska područja** poslovanja neke organizacije (izdavanje računa, naručivanje robe, prijam pacijenata, pristup web-stranici), odnosno **važne entitete** iz stvarnog svijeta (korisnik, proizvod, korisnički račun, ...).
- ◆ Nije orijentirano na procese.

Svojstva skladišta podataka

Tematska orijentacija

- ◆ Tematska orijentacija kod skladišta podataka je u suprotnosti s klasičnom procesnom, odnosno funkcijskom orijentacijom aplikacija kakvu nalazimo kod operacijskih sustava baza podataka.

Svojstva skladišta podataka

Tematska orientacija

- **Klasični operacijski sustavi** su organizirani oko područja djelovanja poduzeća. Za osiguravajuće društvo to su npr. područja: **sklapanje police, produljenje police, ostvarivanje popusta...**
- **Glavne teme skladišta podataka** za osiguravajuće društvo ovdje mogu biti: **korisnik, polica osiguranja, premija** i sl., odnosno teme kao što su **prodano osiguranje, isplaćene štete**, i sl.

Svojstva skladišta podataka

Tematska orientacija

- Operacijsko okruženje automatizira specifične funkcije poslovanja. U bankarskom poslovanju, operacijsko je okruženje dizajnirano oko *zahtjeva za kredit, isplate kredita, primjera mješevne uplate za kredit, zatvaranja kredita, otvaranja i zatvaranja računa, izdavanja kartice*, a okruženje skladišta podataka oko glavnih tema, odnosno važnijih entiteta iz stvarnog svijeta, kao što su: *korisnik, bankarski proizvod, financijska transakcija* itd.

Svojstva skladišta podataka

Tematska orientacija

- ◆ operacijsko okruženje zaokupljeno i **dizajnom baze podataka** (koje relacije postoje u bazi?) i **dizajnom procesa** (nad kojim se relacijama provodi unos, izmjena i brisanje za pojedini slučaj uporabe – use case?)
okruženje skladišta podataka se fokusirano samo na modeliranje podataka i dizajn baze podataka.
- ◆ Dizajn procesa nije dio okruženja skladišta podataka
- ◆ U skladištu nema podataka koji se ne koriste za analitičku obradu i potporu procesu donošenja odluka.

Svojstva skladišta podataka

Tematska orientacija

Operacijski sustavi prate poslovne transakcije, a kod skladišta podataka se iz svih raspoloživih izvora pokušava za određena tematska područja **skupiti podatke, ali samo one koji su zanimljivi i potrebni za analizu.**

Svojstva skladišta podataka

◆ Integriranost

integrated

- više izvora podataka u poduzeću i samo jedno skladište
- heterogeni izvori:
 - operacijske baze (relacijske i starije, naslijedene),
 - datotečni sustavi
 - novi sustavi razmjene informacija (elektroničko poslovanje: B2B, B2C, web-usluge)
- **Najvažnija značajka skladišta podataka jest da su podaci koji se prikupljaju iz raznorodnih izvora u njemu integrirani.**

Svojstva skladišta podataka

Integriranost

Operacijske podatke koji dolaze iz više baza podataka, a vjerojatno i podatke pribavljene izvan poduzeća koje izrađuje skladište podataka, treba transformirati i integrirati prije pohranjivanja u skladištu podataka.

Proces transformacije i integracije podataka može biti najzahtjevniji dio izgradnje skladišta podataka.

Svojstva skladišta podataka

Integriranost

- Pri integraciji heterogenih izvora potrebno je osigurati **konzistentnost** u
 - nazivima podatkovnih elemenata
 - mjernim jedinicama za varijable
 - strukturama šifriranja
 - ključevima podatkovnih elemenata

Svojstva skladišta podataka

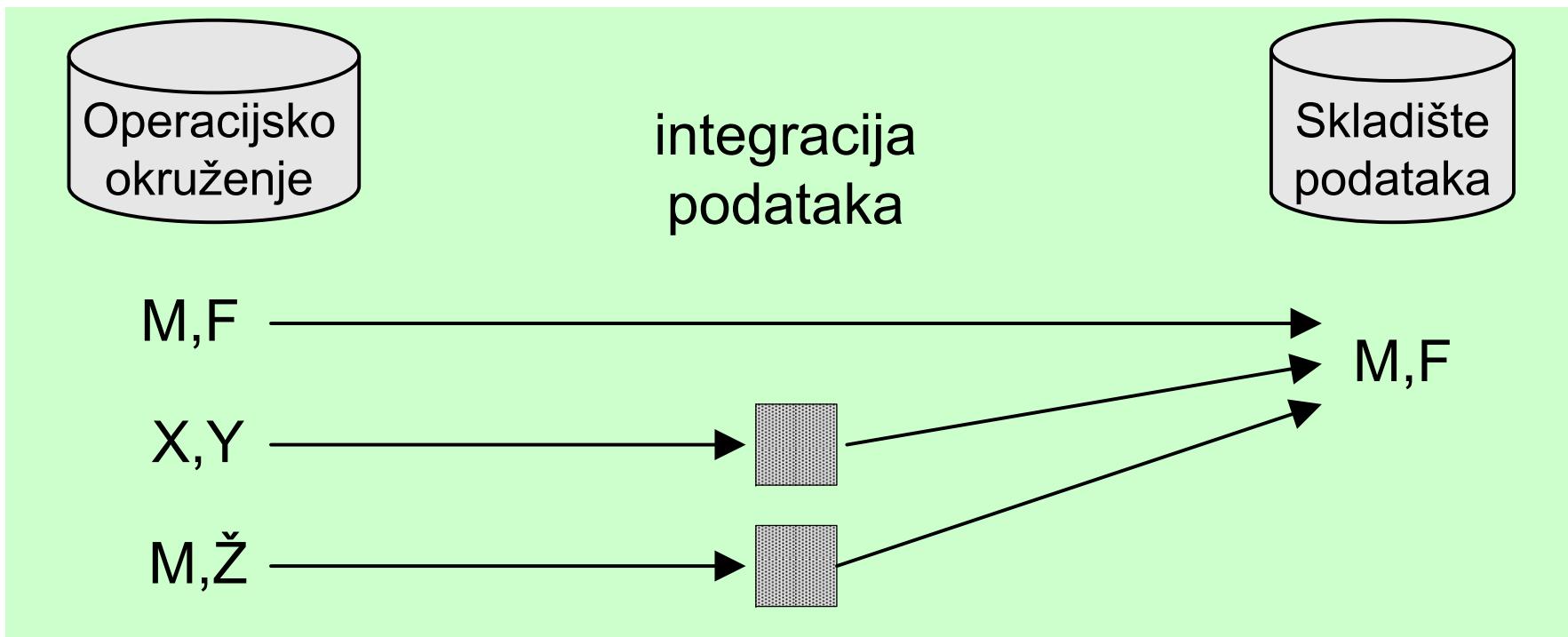
Integriranost

Osnovni problem: razni dizajneri baza podataka (odnosno aplikacija nad njima) tijekom godina donose različite odluke pri dizajnu, nema konzistentnosti među aplikacijama u pitanju šifriranja, konvencija imenovanja, fizičkih značajki, mjernih jedinica, struktura ključeva i sl.

Primjer: dizajneri mogu u operacijskim transakcijskim bazama podataka šifrirati polje koje opisuje spol osobe na različite načine.

Svojstva skladišta podataka

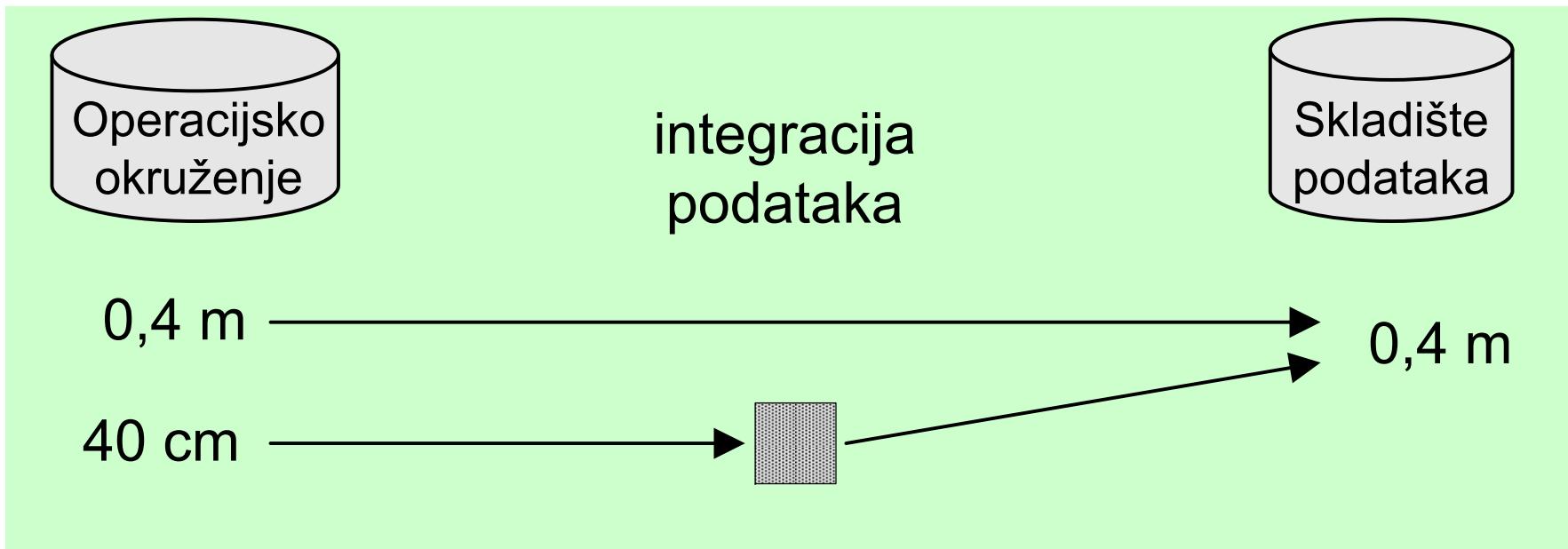
♦ Integriranost



Podaci se integriraju prije učitavanja u skladište podataka
- način šifriranja vrijednosti atributa

Svojstva skladišta podataka

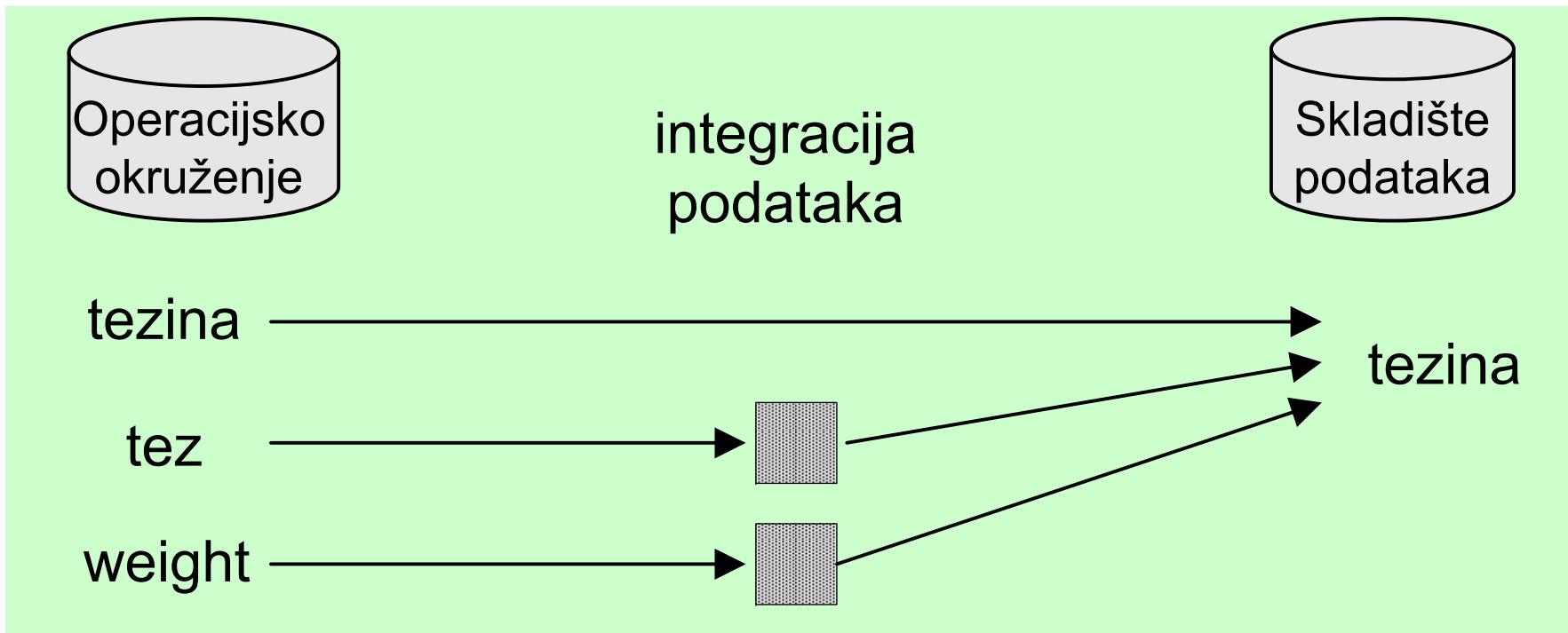
♦ Integriranost



*Podaci se integriraju prije učitavanja u skladište podataka
- mjerne jedinice*

Svojstva skladišta podataka

♦ Integriranost



*Podaci se integriraju prije učitavanja u skladište podataka
- imena atributa*

Svojstva skladišta podataka

Integriranost

Bilo da se radi o fizičkim značajkama podataka, mjernim jedinicama ili šifriranju, podaci trebaju biti spremjeni u skladište podataka na jedinstven, općeprihvatljiv način, i onda kada operacijski sustavi spremaju podatke na različite načine.

Svojstva skladišta podataka

Skup vremenskih snimaka

time-variant

- u operacijskom okruženju podaci su točni za trenutak pristupa podacima
- svrha skladišta podataka: povjesna analiza
- skupljanje povijesnih podataka
 - svaki povijesni podatak aktualan u određenom vremenskom trenutku
- redovno uzimanje aktualnih podataka kroz dugotrajno vremensko razdoblje

Svojstva skladišta podataka

Skup vremenskih snimaka

Skladište podataka je **dugi vremenski slijed zabilježenih snimaka**

- ♦ to je slijed slojeva podataka na kojima su spremljeni **podaci** o poslovanju poduzeća **za određene trenutke, odnosno vremenska razdoblja.**

Svojstva skladišta podataka

Skup vremenskih snimaka

Za podatke u skladištu podataka kaže se da su "vremenski različiti" (time-variant)

- ◆ svaki podatak je točan **za neki protekli trenutak ili razdoblje**
 - podaci stari 5 do 10 godina
 - u skladištu uvijek postoji element vremena
 - ako je snimak napravljen ispravno, nema kasnijih izmjena

Svojstva skladišta podataka

Skup vremenskih snimaka

Statički snimci operacijskog sustava spremaju se u skladište podataka na temelju određenog rasporeda, u isto vrijeme svakog dana, svakog tjedna ili svakog mjeseca – ovisno o zahtjevima poslovanja.

Metapodaci – podaci koji opisuju podatke u skladištu podataka – moraju također imati oznaku vremena.

Svojstva skladišta podataka

Skup vremenskih snimaka

Problem kod spremanja

Poteškoće unose dani i tjedni, budući da je broj dana u mjesecu i godini promjenjiv, a između tjedana i mjeseci nema pravilnih odnosa (mjesec može započeti bilo kojeg dana u tjednu).

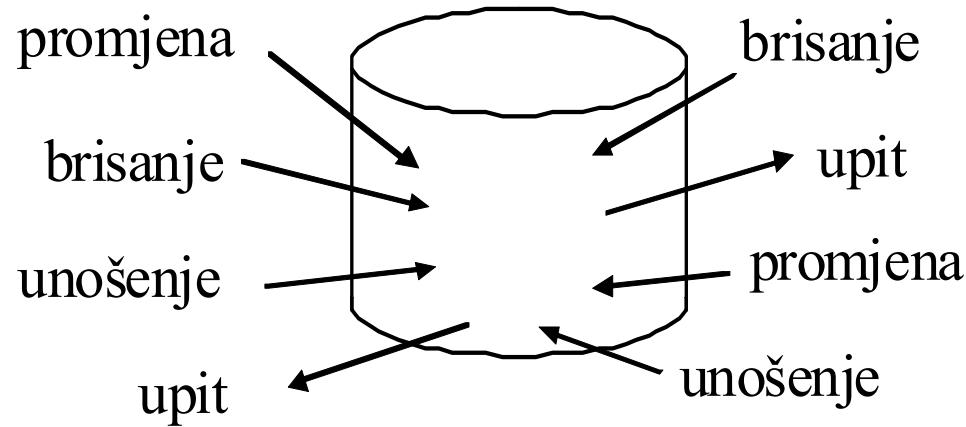
Svojstva skladišta podataka

Nepromjenjivost podataka

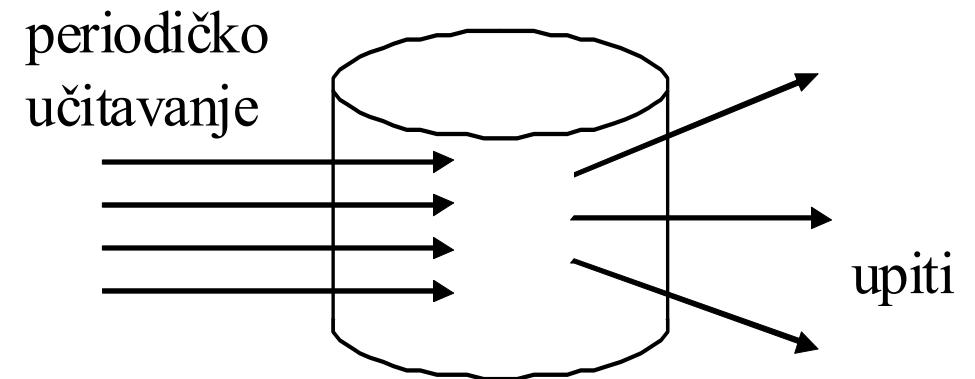
involatile

- ◆ U skladištu podataka dvije vrste operacija:
 - **učitavanje relevantnih podataka**
 - **upiti**
- ◆ Kad su podaci učitani, nad njima više **nema promjena**.
- ◆ Nema ni mijenjanja ni brisanja podataka (osim u slučaju da treba ispraviti pogrešan podatak)

Svojstva skladišta podataka



transakcijska baza podataka



skladište podataka

Svojstva skladišta podataka

Nepromjenjivost podataka

Izvor gotovo svih podataka u skladištu podataka je operacijsko okruženje (tj. operacijske baze podataka i datoteke). Iako bi ta činjenica mogla dovesti do zaključka da postoji velika zalihost između dva okruženja, takav zaključak nije ispravan.

Podaci se filtriraju pri prelasku iz operacijskog okruženja u okruženje skladišta podataka.

Samo oni **podaci** koji su **potrebni kao potpora odlučivanju** dolaze u okruženje skladišta podataka.

Svojstva skladišta podataka

Nepromjenjivost podataka

- Budući da nad podacima u skladištu **nema promjena**, u skladištu ne postoje anomalije unosa, brisanja i mijenjanja podataka, pa su dopuštene određene **slobode u pristupu podacima**, pogotovo **u pogledu normalizacije**.
- Posljedica jednostavnosti operacija nad skladištem podataka je jednostavnije načelo modeliranja

Svojstva skladišta podataka

Nepromjenjivost podataka

- Zbog toga što su podaci u skladištu podataka puno stariji nego oni u operacijskom okruženju, **ima vrlo malo preklapanja između dva okruženja**.
- Zatim, skladište podataka sadrži **sumarne podatke** kakve ne nalazimo u operacijskom okruženju.
- Mnoge podatke treba fizički promijeniti kad ih se prebacuje u skladište podataka tako da više ne možemo govoriti o istim podacima sa stanovišta integracije.

Transakcijska baza vs. skladište

| | OLTP | Skladištenje podataka |
|-----------------------|--|--|
| Sadržaj podataka | Trenutne vrijednosti, detaljni podaci | Povijesni podaci, detaljni i sumarni podaci |
| Vrijednost podataka | Vrlo promjenjiva | Postojana |
| Namjena | Vođenje poslovnog sustava, dnevne operacije | Izvješćivanje o stanju poslovnog procesa, analiza |
| Jedinica obrade | Transakcija | Upit |
| Korisnici | Službenici | Analitičari i menadžeri |
| Raspoloživost | Izuzetno važna | Manje važna |
| Izmjena podataka | Polje po polje | Nema direktne izmjene |
| Radne karakteristike | Čitanje/pisanje | Čitanje |
| Interakcija korisnika | Predodređena | Ad-hoc |
| Pristup zapisima | Desecima | Milijunima |
| Fokus | Spremanje podataka | Dobivanje informacija |

STRUKTURA PODATAKA U SKLADIŠTU PODATAKA

Struktura podataka u skladištu podataka

Razina podataka prema njihovoj zrnatosti:

- ◆ Tekući detaljni podaci
- ◆ Stariji detaljni podaci
- ◆ Blago sumirani podaci
- ◆ Jako sumirani podaci
- ◆ Metapodaci

Struktura podataka u skladištu podataka

Tekući detaljni podaci su podaci na najnižoj razini zrnatosti i odnose se na nedavno poslovanje poduzeća (stari su najviše 2 do 5 godina). Gotovo svi tekući detaljni podaci dolaze iz operacijskog okruženja učestalošću koja odgovara zahtjevima poslovnog okruženja. Obično je pri tome potreban značajan broj transformacija podataka.

Struktura podataka u skladištu podataka

Tekući detaljni podaci su daleko najvažniji u strukturi skladišta podataka jer:

- ◆ odražavaju nedavne događaje koji su uvijek najzanimljiviji za analizu,
- ◆ vrlo je velika količina tih podataka budući da su spremljeni na najnižoj razini zrnatosti,
- ◆ uvijek su spremljeni na disku,
- ◆ koriste se za sumiranje podataka i zato trebaju biti točni.

Struktura podataka u skladištu podataka

Stari detaljni podaci imaju jednaku razinu zrnatosti kao i tekući detaljni podaci. Obično nisu spremljeni na disku, već na nekom drugom mediju (magnetskoj vrpci, CD, DVD, ili naprosto drugom, zasebnom disku), jer se njima rijetko pristupa, a količina tih podataka je velika.

Pri postavljanju upita nad starim detaljnim podacima mora se prihvati slabija izvedba upita.

Stari podaci mogu, međutim, biti raspoloživi u sumarnom obliku na višim razinama zrnatosti gdje su spremljeni na disku i može im se brzo pristupiti.

Struktura podataka u skladištu podataka

Blago sumirani podaci su podaci koji nastaju sumiranjem tekućih detaljnih podataka. Ovaj je sloj skladišta podataka gotovo uvijek spremljen na disku. Pri dizajniranju skladišta podataka treba odrediti:

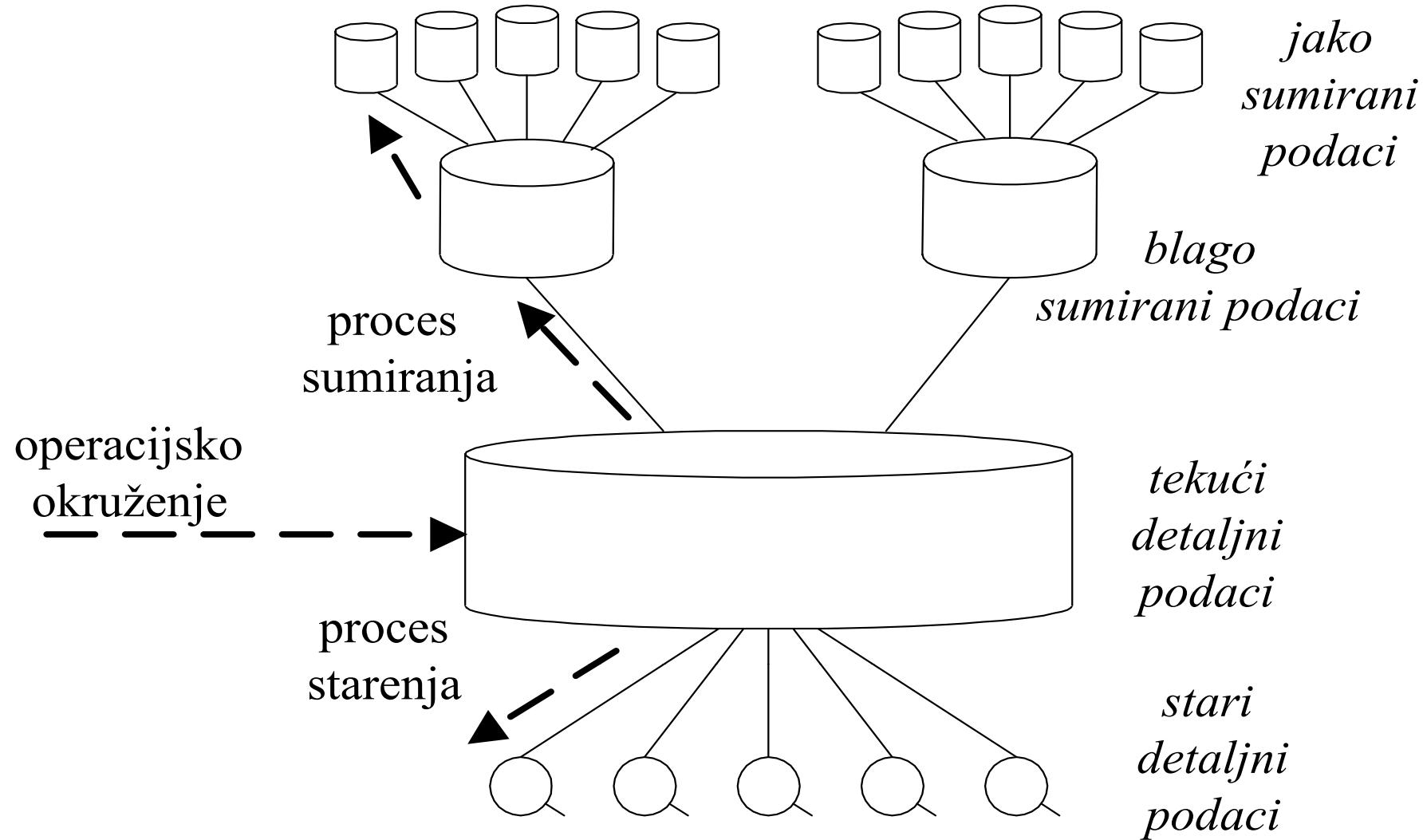
- ◆ po kojoj jedinici vremena se radi sumiranje?
- ◆ koje sadržaje - attribute - će sadržavati blago sumirani podaci?

Struktura podataka u skladištu podataka



Jako sumirani podaci čine sljedeći sloj podataka u skladištu podataka. Ovi su podaci kompaktni i lako im se pristupa. Gotovo uvijek su spremljeni na disku da bi se omogućio brz pristup.

Struktura podataka u skladištu podataka



Struktura podataka u skladištu podataka

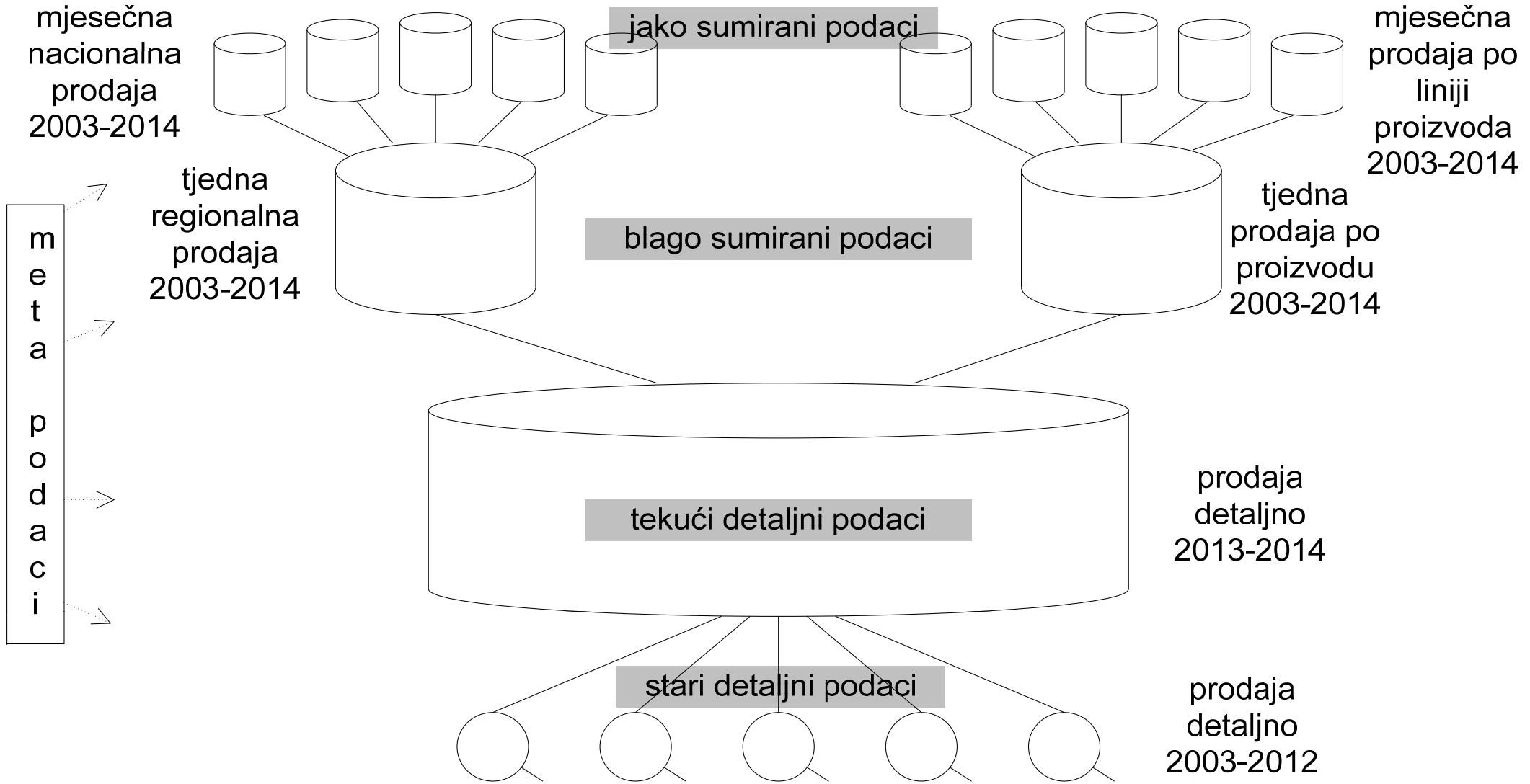
Metapodaci (podaci o podacima) imaju daleko važniju ulogu u okruženju skladišta podataka nego u operacijskom okruženju. Za njih se može reći da leže u dimenziji različitoj od drugih podataka u skladištu podataka jer ne sadrže podatke koji su uzeti direktno iz operacijskog okruženja.

Struktura podataka u skladištu podataka

Metapodaci određuju pravila u skladištu podataka i daju informaciju o:

- ◆ Sadržaju skladišta podataka (**struktura podataka, lokacija podataka, model podataka** itd.)
- ◆ Transformiranju i učitavanju podataka
- ◆ Korištenim algoritmima kod sumiranja podataka
- ◆ Kvaliteti podataka itd.
- ◆ **Metapodaci moraju također imati oznaku vremena.**

Struktura podataka u skladištu podataka

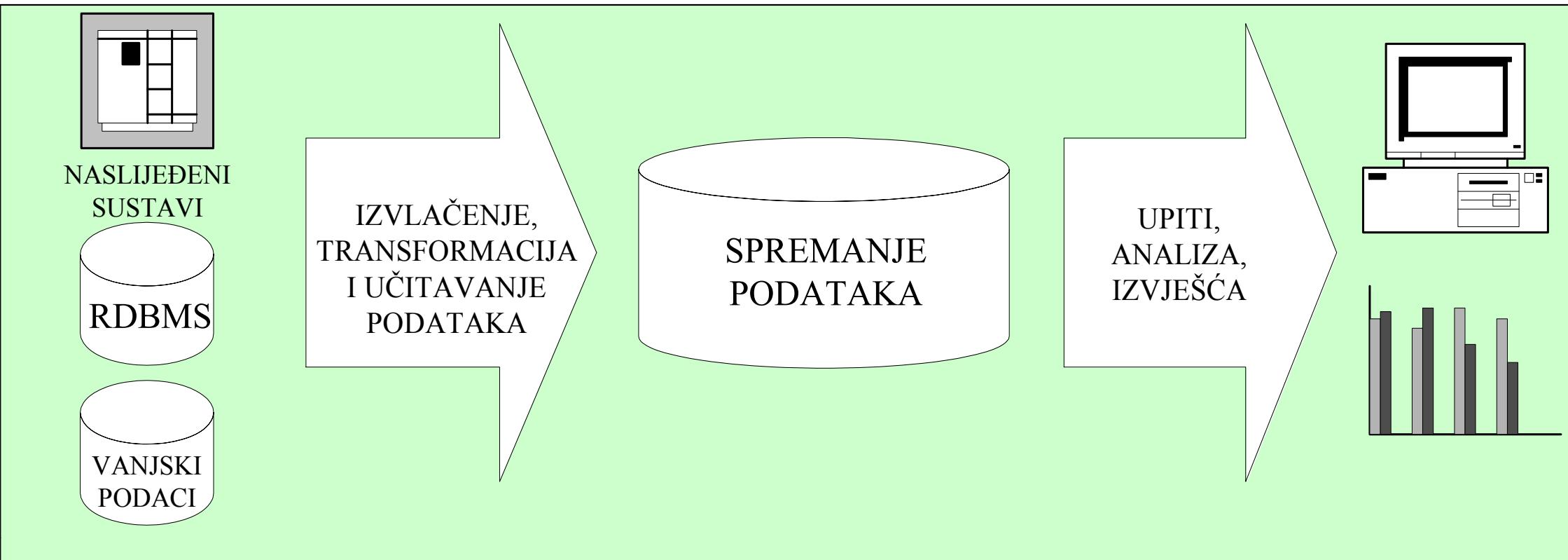


PROCESI U SKLADIŠTENJU PODATAKA

Procesi u skladištenju podataka

- ◆ U postupku skladištenja podataka prisutni su sljedeći **procesi**:
 - **izvlačenje, transformacija i učitavanje podataka**
 - **spremanje podataka i upravljanje podacima**
 - **korištenje skladišta podataka** kao potpora u procesu odlučivanja

Skladištenje podataka - procesi



Izvlačenje, transformacija i učitavanje

Procesi **izvlačenja, transformacije i učitavanja** (*Extraction, Transformation and Loading – ETL*) podataka uključuju niz složenih zadaća koje se odnose na identificiranje odgovarajućeg skupa podataka u operacijskom sustavima i vanjskim izvorima, izvlačenje, čišćenje i transformaciju podataka, prebacivanje podataka u skladište podataka te provjeru kvalitete podataka.

Izvlačenje, transformacija i učitavanje

- ◆ čitanje podataka iz raspoloživih izvora
 - nejasne i nedokumentirane vrijednosti podataka i odnosi među podacima; nekonzistentnost, zalihost
- ◆ izvlačenje podataka iz operacijskih sustava
 - dnevni, tjedni ili mjesecni snimak (u suvremenim skladištima i češće, bar za najvažnije podatke)
 - hvatanje promjena
- ◆ transformacija
 - prilagodba podataka modelu podataka skladišta
- ◆ učitavanje podataka

Izvlačenje, transformacija i učitavanje

Izvlačenje podataka (eng. *extracting*) podrazumijeva čitanje i razumijevanje izvornih podataka, odabir podataka koji su najkvalitetniji i najrelevantniji za poslovnu analizu, te preslikavanje tih podataka radi daljne obrade.



Analiza izvornih podataka

- Znamo li gdje su pohranjeni izvorišni podaci? U kojim sustavima? U kojim datotekama? U kojim bazama podataka?
- Postoje li višestruki izvori za iste podatke?
- Tko su vlasnici podataka?
- Postoji li raspoloživa dokumentacija za izvore podataka? Je li dokumentacija potpuna i jesu li u njoj praćene sve izmjene u sustavu?

Izvlačenje, transformacija i učitavanje

Analiza izvornih podataka:

- ◆ Kvaliteta podataka
 - Znamo li koliko su podaci čisti?
 - Jesu li podaci dovoljno čisti za sve vrste korisnika skladišta podataka?
 - Jesu li pogreške u podacima već dokumentirane?
 - Znamo li koji su podaci više, a koji manje važni?

Izvlačenje, transformacija i učitavanje

Problemi

- ◆ Čitanje podataka iz raspoloživih izvora koji posjeduju **nejasne i nedokumentirane vrijednosti** podataka, a odnosi među podacima imaju dosta **nekonzistentnosti i zalihosti**.



Izvlačenje, transformacija i učitavanje

Problemi

- ◆ Za zastarjele sustave koji se koriste tekstualnim datotekama, mrežnim i hijerarhijskim bazama podataka smještenim na *mainframe* računalima koristi se naziv **naslijedeni sustavi**. Ti su sustavi izvor povijesnih podataka, a **često ne podržavaju on-line režim rada**. (ovi sustavi su operativni čak i danas, 2015.)

Izvlačenje, transformacija i učitavanje

Problemi

- ◆ **Metapodaci**, tj. podaci o imenu datoteka, nazivu polja, ograničenjima i tipovima podataka, mijenjaju se tokom vremena. Te **promjene nisu uvijek dokumentirane**, te se ne može sa sigurnošću utvrditi kada je došlo do promjena u naslijedenoj aplikaciji. Ljudi koji su razvijali te aplikacije rijetko su još dostupni.

Izvlačenje, transformacija i učitavanje

Problemi

- ◆ Pri ispitivanju izvora podataka **problem nekonzistentnosti između više odvojenih sustava** dolazi u središte pozornosti.
- ◆ Podaci su raspoređeni u više operacijskih baza podataka i među njima postoji znatna **zalihost**.
- ◆ **Podaci** koji se odnose na iste entitete **razlikuju se po formi i sadržaju**. Svaki izvor podataka izrađen je na temelju vlastitog skupa zahtjeva - tijekom razvojnog procesa ne vodi se briga o drugim bazama pa nema usklađenog pogleda na podatke.

Izvlačenje, transformacija i učitavanje

Metode izvlačenja podataka iz izvora:

- ◆ Izvlačenje podataka iz **dnevničkih (log) datoteka**.
- ◆ Neki sustavi kreiraju posebnu **datoteku promjena** koja se pri izvlačenju koristi na isti način kao log datoteka.
- ◆ Ako izvorni podaci imaju **oznake vremena**, u procesu izvlačenja podataka može se odabrati samo one podatke koji su se promijenili od zadnjeg izvlačenja podataka.
- ◆ Metoda koja se godinama koristila je **uspoređivanje datoteka**.

Izvlačenje, transformacija i učitavanje



Budući da je skladište podataka slijed snimaka stanja transakcijskih baza, podaci se trebaju izvlačiti tako da se **izvlačenje** podataka iz jednog izvora **odnosi na isti trenutak** u vremenu kao i izvlačenje iz drugih izvora podataka.

Izvlačenje, transformacija i učitavanje

Transformacija podataka

- ◆ Kad se podaci izvuku iz operacijskih baza podataka i datoteka, te vanjskih izvora, treba ih pripremiti za učitavanje u skladište podataka – dakle treba ih **transformirati u prikladni format**.
- ◆ Podatke treba **prilagoditi modelu podataka** odredišne baze podataka (tj. skladišta podataka), **provjeriti njihovu točnost i kvalitetu** te ih **“očistiti” i integrirati**.

Izvlačenje, transformacija i učitavanje

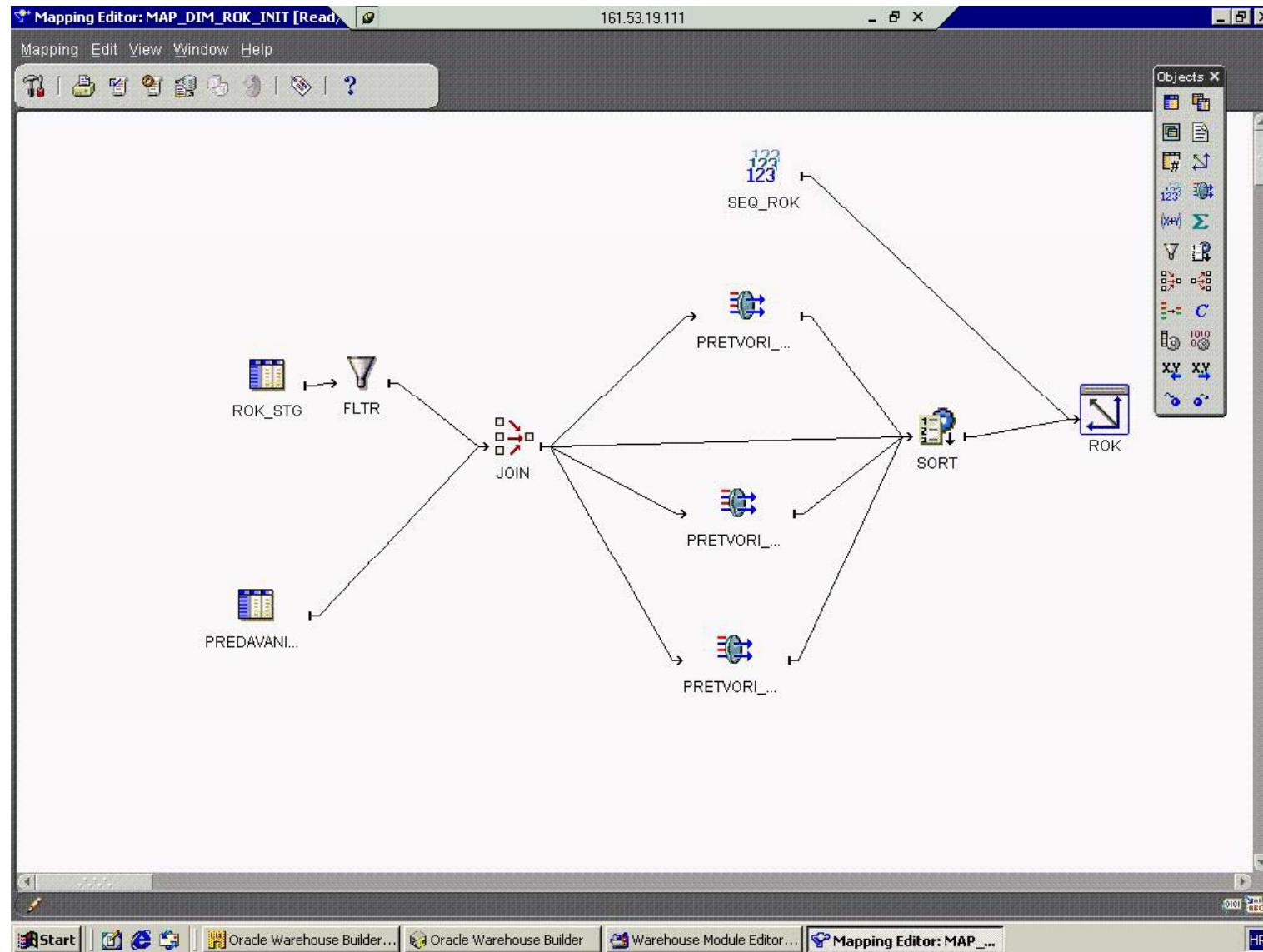
Transformirani podaci moraju biti:

- Točni
- Relevantni za poslovne potrebe
- Konzistentni za zalihosne izvore
- Potpuni, tj. moraju sadržavati informacije potrebne za odgovore na poslovna pitanja

Izvlačenje, transformacija i učitavanje

- ◆ Podaci izvučeni iz izvornih sustava obično se učitavaju u **pripremno skladište podataka** koje čine sve baze podataka i datoteke u koje se podaci spremaju na putu između izvornog i odredišnog sustava.
- ◆ Pripremno skladište podataka će često biti relacijska baza podataka.
- ◆ Ako je izvor podataka tekstualna datoteka, podaci se i u pripremnom skladištu podataka mogu držati u tekstualnim datotekama.

Izvlačenje, transformacija i učitavanje – alat



Transformacija podataka u *pripremnom spremištu podataka* uključuje:

- ◆ **čišćenje** podataka
- ◆ **kombiniranje više izvora** podataka i **integraciju** podataka
- ◆ **prilagodbu izvornih podataka modelu podataka skladišta**
- ◆ **nametanje novih ključeva**
- ◆ **izgradnju agregacija** (tj. organiziranje pohrane određenih podataka dobivenih korištenjem agregatnih funkcija SUM, AVG, COUNT) radi poboljšanja izvedbe čestih upita

Izvlačenje, transformacija i učitavanje

Loša kvaliteta podataka, prema istraživanjima SAS Instituta, **uzrok je neuspjeha u 70% projekata** izgradnje skladišta podataka što je dovoljan razlog da se tom pitanju posveti dužna pažnja prilikom svih faza izgradnje sustava.

U većini slučajeva pitanja kvalitete podataka su izvan utjecaja tima zaduženog za skladište podataka.

Kvaliteta podataka u skladištu uglavnom ovisi o ispravnosti izvora podataka.

Izvlačenje, transformacija i učitavanje

- ◆ **Pregled kvalitete podataka u operacijskim sustavima:**
 - zalihost podataka,
 - nedostatak zajedničkog standarda za pohranjivanje podataka,
 - jednostavne greške pri unosu podataka,
 - nedostatak određenih podataka,
 - raznovrsni podaci upisani u tekst slobodne forme
- ◆ **Praćenje stanja kvalitete podataka** u procesu izvlačenja, transformacije i učitavanja

Izvlačenje, transformacija i učitavanje

Kvaliteta podataka - primjer

- ◆ uzmimo da u transakcijskoj bazi podataka postoji tablica PARTNER u koju se pohranjuju podaci o dobavljačima proizvoda.
- ◆ ako tu transakcijsku bazu želimo iskoristiti kao izvor podataka za skladište podataka pomoću kojeg ćemo analizirati prodaju, potrebno je prvo ispitati kvalitetu podataka

| Column Name | Data Type |
|----------------------|-------------------|
| PARTNER_ID | NUMBER(12,0) |
| SIFRA | VARCHAR2(10 BYTE) |
| NAZIV | VARCHAR2(80 BYTE) |
| SKR_NAZIV | VARCHAR2(50 BYTE) |
| MAT_BROJ | VARCHAR2(20 BYTE) |
| ADRESA | VARCHAR2(80 BYTE) |
| VRSTA_PARTNERA_ID | NUMBER(2,0) |
| STATUS_AKTIV_ID | NUMBER(2,0) |
| POSL_STATUS_ID | NUMBER(2,0) |
| POSTA_ID | NUMBER(8,0) |
| DAT_PROMJENE | DATE |
| OZN_PROMJENE | VARCHAR2(8 BYTE) |
| DANA_DOSPIJECA | NUMBER(3,0) |
| MJESTO | VARCHAR2(25 BYTE) |
| PODRUCJE_ID | NUMBER(8,0) |
| ZATEZNA_KAM | NUMBER(8,3) |
| DRZAVA_ID | NUMBER(4,0) |
| POSLOVNA_JEDINICA_ID | NUMBER(4,0) |
| REFERENT_ID | NUMBER(4,0) |

Izvlačenje, transformacija i učitavanje

Kvaliteta podataka - primjer

- za atribut MAT_BROJ u tablici PARTNER čak 64.2% redaka nema definiranu vrijednost (tj. ima vrijednost NULL)
- također, u velikom broju primjera je za matični broj upisana besmislena vrijednost
- Zbog navedenih razloga,, atribut MAT_BR nećemo uključiti u skladište podataka!

Data Drill Panel

Here are drill results on PARTNERI column MAT_BROJ.

Distinct values:

| | MAT_BROJ | # Rows | % of 9269 |
|----|----------------------|--------|-----------|
| 1 | ✓ | 5950 | 64.2% |
| 2 | ✓ 3203077 | 24 | .3% |
| 3 | ✓ * | 13 | .1% |
| 4 | ✓ 9999999999999 | 8 | .1% |
| 5 | ✓ 3586243 | 5 | .1% |
| 6 | ✓ 0714178 | 4 | 0% |
| 7 | ✓ 0980633 | 4 | 0% |
| 8 | ✓ 3276147 | 4 | 0% |
| 9 | ✓ 1414887 | 4 | 0% |
| 10 | ✓ 3207595 | 4 | 0% |
| 11 | ✓ 222222222222222222 | 3 | 0% |
| 12 | ✓ 1414895 | 3 | 0% |
| 13 | ✓ 0487422 | 3 | 0% |
| 14 | ✓ 0567167 | 3 | 0% |
| 15 | ✓ 3271471 | 3 | 0% |

Izvlačenje, transformacija i učitavanje

Kvaliteta podataka – primjer

- ◆ Točnost – podaci moraju biti u skladu sa stanjem realnog svijeta
 - *ista osoba?*

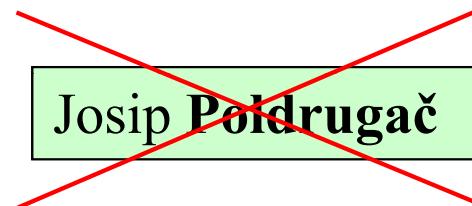
| | |
|----------------|--------------|
| Antun Petrović | Slavonska 35 |
|----------------|--------------|

| | |
|----------------|--------------|
| Antun Petrović | Slovenska 35 |
|----------------|--------------|

- *susjedi?*

| | |
|-------------|---------------|
| Ivan Sertić | Gundulićeva 5 |
|-------------|---------------|

| | |
|-----------------|-------------------|
| Josip Poldrugač | Ivana Gundulića 5 |
|-----------------|-------------------|



Izvlačenje, transformacija i učitavanje

Kvaliteta podataka - primjeri

- ◆ Točnost – podaci moraju biti u skladu sa stanjem realnog svijeta
 - *zaustavljeno vrijeme?*

| Ime | Prezime | Strucna_sprema | Prosj_mjesecni_prihod |
|-----|---------|----------------|-----------------------|
| Ana | Ilić | SSS | 3967,08 |

Podaci iz
2008.

| Ime | Prezime | Strucna_sprema | Prosj_mjesecni_prihod |
|-----|---------|----------------|-----------------------|
| Ana | Ilić | SSS | 3967,08 |

Podaci iz
2014.

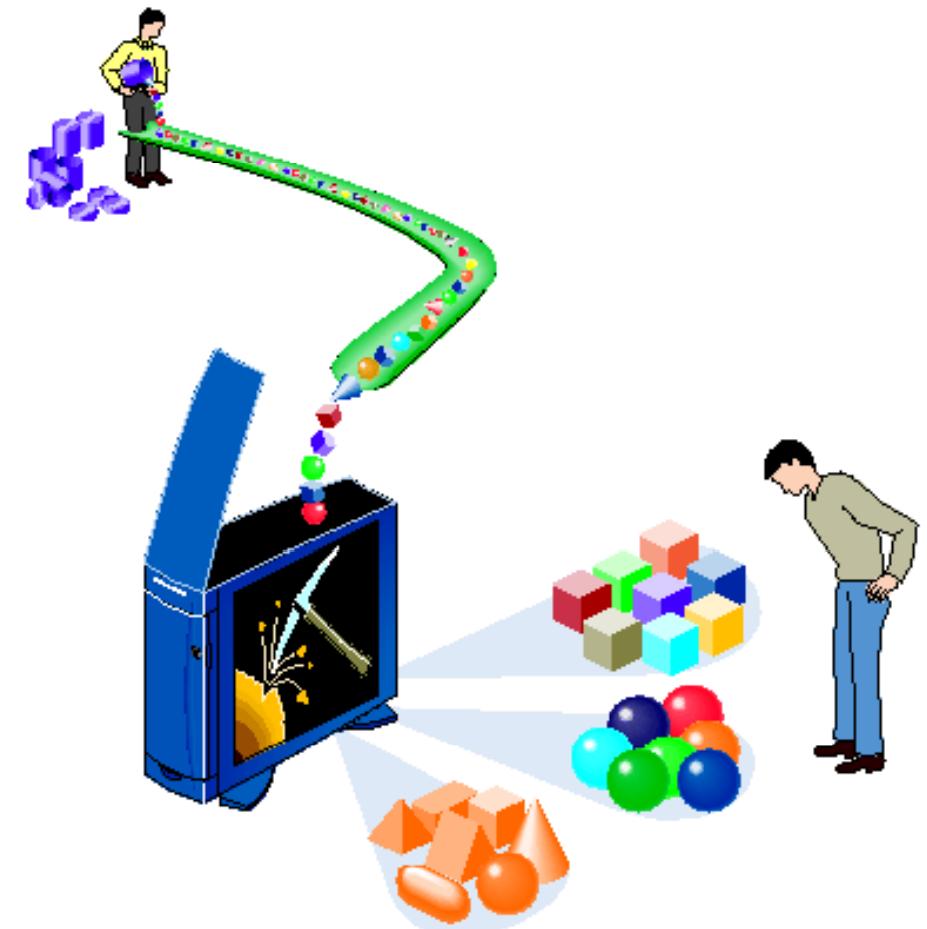
*Promjena
prezimena? (do)školovanje?*

*napredovanje na radnom
mjestu?*

Izvlačenje, transformacija i učitavanje

Većina operacijskih sustava gradila se **bez podatkovnih standarda na razini čitave organizacije**.

To znači da ima vrlo malo konzistentnosti među sustavima, tj. koliko ima operacijskih baza podataka u organizaciji toliko ima i načina spremanja podataka npr. o imenima i adresama poslovnih korisnika.



Izvlačenje, transformacija i učitavanje

Problem konzistentnosti postoji i unutar jednog sustava.

Na primjer, podaci o korisnicima se pohranjuju **u jednom polju** i nema standarda o velikim slovima, kraticama i poretku naziva.

Ako nema ograničenja na unos podataka u to polje, krajnji korisnici mogu unijeti različite tekstualne opise za isti podatkovni element.

Atributi važni za poslovanje mogu tako biti skriveni u tekstu slobodne forme.

Anomalije neispravnog upisivanja podataka (pogreške u *spellingu*) su česte i rezultiraju udvostručenjem podataka.

Izvlačenje, transformacija i učitavanje

Netočni i udvostručeni podaci, te podaci koje se ne uklapaju u opis polja i poslovna pravila, **posljedica su loših sustava ili loših praksi hvatanja podataka.**

Moguće je da sustav nema ugrađenu potporu za provjeru integriteta podataka ili da osoba odgovorna za unos podataka nije motivirana za održavanje stopostotne kvalitete podataka.

Neki će podaci, koje se želi pohraniti u skladište podataka, nedostajati u izvornim sustavima.

Izvlačenje, transformacija i učitavanje

Karakteristike kvalitetnih podataka:

- ◆ **Ispravnost** – podatak u bazi skladišta podataka odgovara podatku iz izvora, a ako ne odgovara postoji dokumentiran razlog različitosti.
- ◆ **Potpunost** – podaci u bazi skladišta podataka predstavljaju cijeli skup relevantnih podataka, npr. element pod nazivom *ukupan_prihod* treba sadržavati i podatke iz podružnice u Austriji, ako nema podataka za tu podružnicu on očito ne opisuje ukupan prihod; potrebno je preimenovati ga (i provjeriti njegov opis u dokumentaciji)

Izvlačenje, transformacija i učitavanje

Karakteristike kvalitetnih podataka:

- ◆ **Konzistentnost** – podaci ne smiju biti kontradiktorni, npr. agregacija (sumiranje) podataka mora odgovarati sumi detaljnih podataka.
- ◆ **Jedinstvenost** – npr. partner pod nazivom IN2 i IN_2 predstavljaju ime jedne tvrtke te stoga u skladištu podataka može postojati samo jedan oblik.

Izvlačenje, transformacija i učitavanje

Karakteristike kvalitetnih podataka:

- ◆ **Pravovremenost** – podaci u skladištu podataka odgovaraju određenom vremenskom trenutku, npr. stanje broja korisnika u 3/2015 godine treba usporediti s jednakim takvim izvještajem iz izvora podataka.
Druga komponenta pravovremenosti je sam proces prijenosa koji u pravilnim vremenskim periodima dopunjuje bazu skladišta podataka. U slučaju prekida procesa potrebno je znati vrijeme zadnjeg uspješnog prijenosa odnosno datum valjanosti baze skladišta podataka.

Izvlačenje, transformacija i učitavanje

Pri izgradnji skladišta podataka se otkrije više problema što se tiče kvalitete podataka u izvornim bazama podataka nego kod bilo koje druge aktivnosti.

Proces izvlačenja, transformacije i učitavanja podataka eksponira problem kvalitete, ali ne osigurava nužno zadovoljavajuću kvalitetu podataka. Promjene usmjerene na **poboljšanje kvalitete podataka** su važne za skladište podataka jer **skladište sadrži podatke koji su ključni za proces donošenja odluka**.

Izvlačenje, transformacija i učitavanje

Velik je broj organizacija u kojima se shvaća da postoji problem nekvalitetnih podataka koji **zahtijevaju čišćenje**, ali se izbjegava suočavanje s tim problemom u žurbi da se izgradi prvo skladište podataka.

Poboljšanje kvalitete podataka se redovito ističe kao **jedan od najvažnijih ciljeva** izgradnje skladišta podataka. Međutim, gotovo redovito se **ne investira dovoljno** u tu svrhu i tako kvaliteta podataka ostaje jedan od najproblematičnijih dijelova procesa skladištenja podataka.

Izvlačenje, transformacija i učitavanje

Zbog svih navedenih problema u vezi s nezadovoljavajućom kvalitetom podataka, **treba provesti čišćenje podataka prije njihovog učitavanja** u skladište podataka.

Čišćenje podataka prije ubacivanja u skladište podataka od životne je važnosti za projekt skladištenja podataka budući da **o kvaliteti podataka pohranjenih u skladište podataka ovisi ispravnost odluka** koje se donose na temelju tih odluka.

Izvlačenje, transformacija i učitavanje

Čišćenje podataka je postupak **otklanjanja nekonzistentnosti , pogrešaka i anomalija** koje bi imale nepovoljan utjecaj na daljnju obradu i korištenje podataka.



Izvlačenje, transformacija i učitavanje

Čišćenje podataka uključuje:



- ispravljanje pogrešaka nastalih pri unosu podataka
- rad s podatkovnim elementima koji nedostaju
- raščlanjivanje podataka u standardne formate
- osiguravanje domenskog i entitetskog integriteta

Izvlačenje, transformacija i učitavanje

Čišćenjem podataka smanjuje se rizik nevaljanih odluka i troškovi povezani s netočnim, nepotpunim i zalihosnim podacima.

Osigurava se da podaci budu:

- konzistentni unutar sebe
- konzistentni s drugim podacima u istom izvoru podataka
- konzistentni s podacima u drugim izvorima
- konzistentni s podacima koji su već u skladištu podataka

Izvlačenje, transformacija i učitavanje

- ◆ Čišćenje podataka je korisno je pogotovo za područje marketinga i to **za obradu popisa imena i adresa**.
- ◆ Odjel marketinga treba imati pouzdane podatke o poslovnim korisnicima da bi se znalo tko su korisnici i da bi se učinkovito komuniciralo s njima.
- ◆ Korištenjem besmislene ili krivo napisane adrese ili slanjem više jednakih pisama istoj osobi tvrtka gubi na ugledu, a ako pismo ni ne stigne na odredište zbog nevaljane korisničke adrese, loši podaci izravno će dovesti do poslovnih troškova.

Izvlačenje, transformacija i učitavanje

- ◆ **Podaci o poslovnim korisnicima** često se prikupljaju iz više unutarnjih i vanjskih izvora.
- ◆ **Spajanja podataka o imenima i adresama korisnika** iz više izvora bit će problematično u slučajevima kad postoje sitne razlike u njihovom načinu pisanja.
- ◆ Budući da će anomalije obično biti najčešće oko podataka o najčešćim poslovnim korisnicima, onda će čak i mali postotak nepravilnosti u izvornim podacima imati **značajan negativan utjecaj na poslovanje**.

Izvlačenje, transformacija i učitavanje

Važni podaci o poslovnim korisnicima, kao što su imena i adrese korisnika, **često su upakirani u nekoliko općih polja** gdje pravila upisa nisu strogo određena.

Proces čišćenja se u takvim slučajevima može rastaviti na šest koraka:

- raščlanjivanje,
- standardiziranje,
- verifikacija,
- spajanje ekvivalentnih podataka,
- spajanje po kućanstvima,
- dokumentiranje.

Izvlačenje, transformacija i učitavanje

- ◆ U sljedećem primjeru adresa se ubacuje u tri slobodna polja nazvana *Adresa_1*, *Adresa_2* i *Adresa_3*:

Adresa_1: Fakultet elektroteh i

Adresa_2: rač Unska 3

Adresa_3: 10000 Zagreb

- ◆ Proces čišćenja obavit će se u navedenih šest koraka.

Izvlačenje, transformacija i učitavanje

1. Raščlanjivanje

- ◆ Sadržaj ovih općih polja treba rastaviti na sve sastavne dijelove.
- ◆ Adresa upisana u ovom primjeru može se raščlaniti na sljedeće elemente:

Ustanova: Fakultet elektroteh i rač

Ulica: Unska

Kućni broj: 3

Poštanski broj: 10000

Mjesto: Zagreb

Izvlačenje, transformacija i učitavanje

2. Standardiziranje

- ◆ Elementi zapisa se stavljaju u standardniji oblik da bi sadržavali identične vrijednosti za ekvivalentne podatkovne elemente.
- ◆ U danom primjeru promijenit će se podatkovni element “**Fakultet elektroteh i rač**” u standardniji oblik: “**Fakultet elektrotehnike i računarstva**”.

Izvlačenje, transformacija i učitavanje

3. Verifikacija

- ◆ Provjerava se jesu li podaci ispravni prema specifičnim pravilima (npr. odgovaraju li međusobno poštanski broj i naziv mjesta).
- ◆ Uspoređuje se podatke s poznatim popisom (obično adrese) i označava polja kao dobra, loša ili automatski ispravljiva.

Izvlačenje, transformacija i učitavanje

4. Spajanje ekvivalentnih podataka.

- ◆ Nakon što se imena i adrese očiste i stave u standardizirane formate, sortiranjem datoteke o korisnicima pronađaze se podaci o određenom korisniku u više zapisa, te se provjerava jesu li svi elementi svih adresa identični. Ako jesu, znači da zapisi za koje se činilo da opisuju više korisnika zapravo opisuju samo jednog korisnika.
- ◆ Problem legitimno promijenjene adrese može se riješiti označavanjem odgovarajućih tipova elemenata oznakama tipa "prethodna" i "sadašnja" (adresa).

Izvlačenje, transformacija i učitavanje

5. Spajanje po kućanstvima

- ◆ Složeniji oblik spajanja podataka je povezivanje podataka o više osoba pod identifikator jednog kućanstva. Ispituje se čini li više osoba koje dijele istu adresu jedno kućanstvo pri čemu treba biti pažljiv da se ne uključi rođake (isto prezime!) u različitim stanovima u istoj zgradici. Najuobičajeniji slučaj čine supružnici koji imaju različite pojedinačne i zajedničke bankovne račune s različitim manjim razlikama u poljima imena i adrese.

Izvlačenje, transformacija i učitavanje

6. Dokumentiranje

- ◆ Sva pravila i rezultate prethodno navedenih koraka treba dokumentirati u obliku metapodataka. Metapodaci će se moći koristiti u budućim postupcima čišćenja istih podataka čime će se osigurati bolje prepoznavanje adresa.

Izvlačenje, transformacija i učitavanje

Da bi podaci u skladištu podataka bili točni, konzistentni i pouzdani potrebno je **baviti se kvalitetom podataka tijekom čitavog procesa** skladištenja podataka. To **uključuje**:

- ◆ pregled kvalitete podataka u operacijskim sustavima,
- ◆ razvoj i izvršenje plana djelovanja za čišćenje podataka,
- ◆ praćenje stanja kvalitete u procesu izvlačenja, transformacije i učitavanja,
- ◆ osiguravanje da podaci u određenom okruženju budu pouzdani, konzistentni i da odgovaraju potrebama poslovnih korisnika.

Izvlačenje, transformacija i učitavanje

Tehnike **čišćenja, transformacije i integracije** moraju se primjenjivati **na iterativan način**.

Treba **specificirati logiku** za transformaciju podataka i **automatizirati proces** što više moguće (korištenjem specijaliziranih alata i programske podrške).

Složenost procesa ovisi o dizajnu skladišta podataka, strukturi izvornog sustava, čistoći izvornog sustava i zahtjevima za integracijom izvornih sustava.

Informacije o izvlačenju i transformaciji treba **dokumentirati** jer one čine važne metapodatke.

Izvlačenje, transformacija i učitavanje

- ◆ **Kad se podaci učitaju** u skladište podataka treba napraviti **provjeru stanja** učitanih podataka.
- ◆ **Prebrojavanje i zbroj odgovarajućeg skupa podataka donosi jednostavnu i brzu provjeru podataka.**
- ◆ Ako su sva prebrojavanja ispala prema očekivanjima, a zbrojevi ili prosječne vrijednosti se nalaze unutar specificirane gornje i donje granice, onda se može s određenom pouzdanošću tvrditi da je učitavanje visoke kvalitete te da krajnji korisnici mogu vjerovati sadržaju skladišta.

Izvlačenje, transformacija i učitavanje

- ◆ **Kvaliteta podataka u skladištu se može redovito provjeravati uzimanjem slučajnih uzoraka podataka.**
- ◆ Provjerava se jesu li podaci unutar granica koje imaju smisla za poslovanje
- ◆ I nad 1% skladišta može se procijeniti sveukupna kvaliteta podataka u skladištu.
- ◆ Provjerava se jesu li vrijednosti podataka konzistentne s vremenskim slijedom sličnih vrijednosti koje su im prethodile. Također, uspoređuju se podaci s vrijednostima u operacijskim sustavima.

Izvlačenje, transformacija i učitavanje

Za osiguranje kvalitete podataka u skladištu podataka važno je da **krajnji korisnici skladišta daju informatičkom osoblju informacije o kvaliteti podataka.**

Krajnji korisnici će bolje od informatičkog osoblja razumjeti poslovne razloge za moguću besmislenost nekih podataka u skladištu i bolje će otkrivati podatke čija vrijednost nema smisla.

Izvlačenje, transformacija i učitavanje

Alati za pregled i poboljšanje kvalitete podataka

Velik broj alata se može koristiti za automatiziranje čišćenja podataka i drugih oblika popravljanja kvalitete podataka.

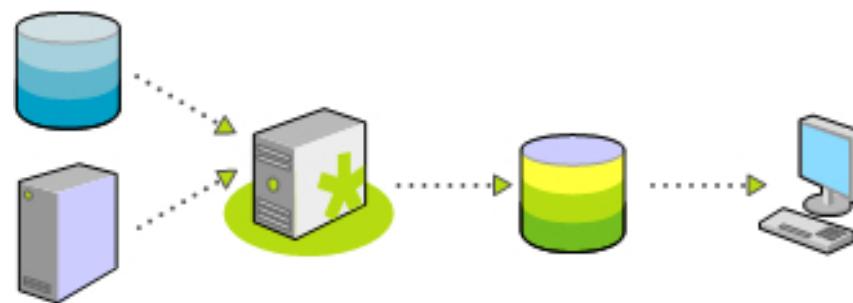
Ovi proizvodi izvode jednu ili više sljedećih funkcija:

- **pregled kvalitete**
- **otkrivanje pravila među podacima**
- **čišćenje podataka**

Izvlačenje, transformacija i učitavanje

◆ Učitavanje podataka

- **Dnevno, tjedno ili mjesечно** (u suvremenim sustavima i češće, bar za najvažnije podatke)
- Ovaj proces nastoji se automatizirati – obavlja se obično noću, prema određenom rasporedu
- Treba nametnuti referencijski integritet podataka
- Provjera **kvalitete** učitanih podataka



Spremanje podataka i upravljanje

- ◆ **Podaci se spremaju** u bazu podataka odvojenu od postojećih operacijskih (OLTP) baza.
- ◆ Posebnu pozornost treba posvetiti **omogućavanju brzog i učinkovitog pristupa podacima** – oblikovanje **odgovarajućeg modela podataka**.



- ◆ Upravljanje uključuje omogućavanje sigurnog pristupa podacima, suočavanje s problemom stalnog povećanja količine podataka te osiguravanje raspoloživosti podataka u slučaju pogreški u sustavu.

Korištenje skladišta podataka

Korištenje skladišta podataka

podrazumijeva:

postavljanje upita nad podacima pohranjenih u skladištu podataka,

analizu tih podataka,

izradu izvješća,

pronalaženje nepoznatih zakonitosti među podacima i

objavljivanje dobivenih rezultata.



Korištenje skladišta podataka

- ◆ Tri osnovna načina korištenja:
 - **izrada izvještaja (reports)**
 - **analitička obrada OLAP (On-line analytical processing)**
 - Podaci se organiziraju te pomoću tablica i grafova prikazuju na način koji odražava višedimenzionalnost poslovanja
 - Korisnik definira parametre i kriterije analize
 - **dubinska analiza podataka (data mining)**
 - Otkrivanje dotad nepoznatih informacija, podudarnosti ili pravilnosti u ogromnom skupu podataka
 - Koriste se različiti matematički i statistički algoritmi

Korištenje skladišta podataka

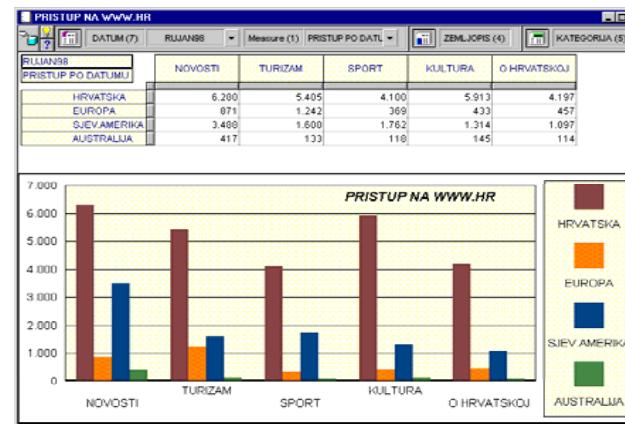
Alati za izradu izvještaja:

- ◆ generiranje izvještaja koji sadrže tablice i grafove
- ◆ prikaz podataka koji su filtrirani, sortirani ili sumirani po određenim kriterijima
- ◆ dovoljni samo u manje složenim dijelovima sustava poslovne inteligencije, često se koriste u kombinaciji sa drugim alatima
- ◆ nefleksibilnost - potrebno stalno sudjelovanje informatičara u procesu izrade i održavanja

Korištenje skladišta podataka

OLAP alati:

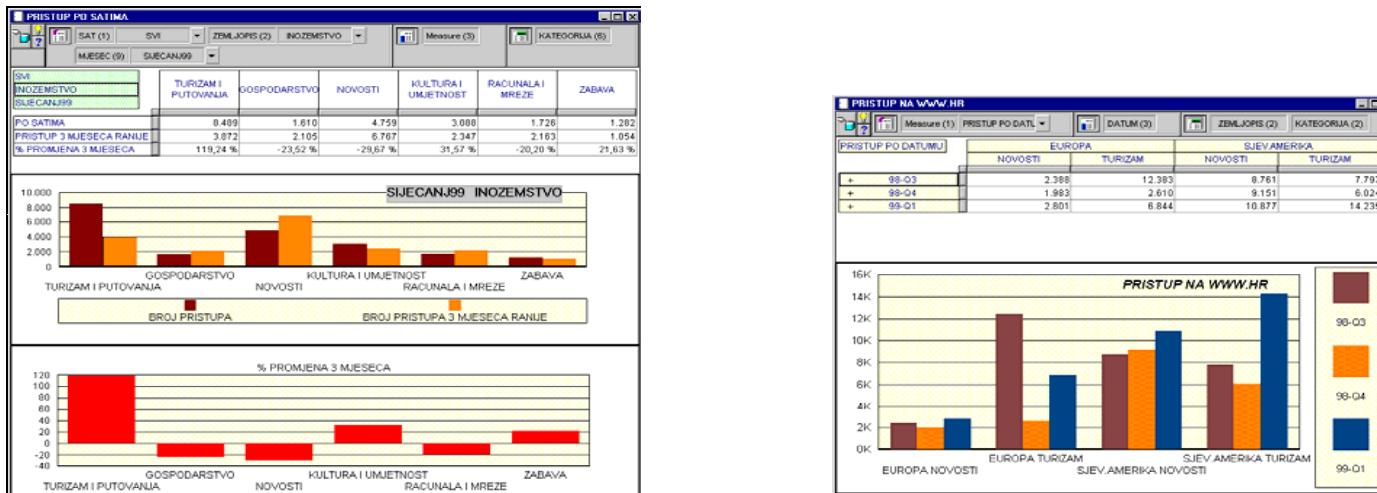
- ♦ analiza velike količine podataka na brz, konzistentan i **interaktivni** način
- ♦ podaci se organiziraju, te pomoću tablica i grafova prikazuju na način koji odražava dimenzionalnost poslovanja tj. način na koji krajnji korisnik razmišlja pri analizi poslovanja



Korištenje skladišta podataka

OLAP alati:

- ◆ dinamička **višedimenzionalna analiza** podataka
- ◆ prikazuju se sumarni i detaljni podaci
- ◆ pogled na podatke iz različitih perspektiva
- ◆ u pravilu **više upita** od kojih svaki sljedeći *izravno ovisi o rezultatu prethodnog*



Korištenje skladišta podataka

Primjeri - OLAP alati daju odgovore na pitanja:

- ◆ Kolika je prodaja proizvoda A za prošli mjesec u Hrvatskoj u odnosu na isti mjesec godinu dana ranije?

- ◆ Kojih 10 proizvoda je donijelo najveću zaradu u proteklom tromjesečju na području grada Zagreba? Koji su se proizvodi najslabije prodavali u istom razdoblju?

- ◆ Kad bi u čitavoj Hrvatskoj bio porast prodaje vrste proizvoda C jednak porastu u Zagrebu u 2012. godini, kolika bi bila prodaja u Hrvatskoj u toj godini?

Korištenje skladišta podataka



DUBINSKA ANALIZA PODATAKA *(data mining)*

Analiza **velikih** skupova podataka s ciljem pronalaženja **neočekivanih veza i uzoraka** u skupovima podataka ili sumarnog prikaza skupa podataka na način da vlasniku ili korisniku podataka pruža **nove, razumljive i korisne informacije**.

Korištenje skladišta podataka

- ◆ Pod pojmom **POSLOVNA INTELIGENCIJA** (*Business Intelligence* – **BI**) podrazumijevamo aplikacije i tehnologije za prikupljanje podataka, omogućavanje pristupa podacima i analize tih podataka u svrhu dobivanja korisnih informacija o poslovanju neke tvrtke, kao i znanja o čimbenicima koji utječu na poslovanje.

Sustavi poslovne inteligencije se obično oslanjaju na skladišta podataka koja garantiraju jednostavan i brz pristup informacijama.

Korištenje skladišta podataka

- ◆ Napomena:

Dok neki smatraju sustav poslovne inteligencije (*BI system*) sinonimom za sustav skladištenja podataka (*DW system*),
drugi taj pojam vežu samo uz alate za izvještavanje, OLAP i dubinsku analizu podataka.



Korištenje skladišta podataka

Sustavi BI (*Business Intelligence Systems*) obuhvaćaju široki skup alata za potporu poslovnom odlučivanju kao što su alati za:

- izvještavanje,
 - OLAP,
 - dubinsku analizu podataka (*data mining*).
-
- ◆ Često se upotrebljava i *MS Excel* (ostali alati obično imaju opciju prebacivanja podataka u *Excel*) i srodni alati drugih proizvođača

Korištenje skladišta podataka

- ◆ Suvremena istraživanja u području poslovne inteligencije bave se također integracijom sustava poslovne inteligencije sa:
 - sustavima za upravljanje rezultatima tvrtke (Corporate Performance Management - CPM),
 - uravnoteženim karticama postignuća (Balanced Scorecards)
 - upravljanjem poslovnim procesima (Business Process Management - BPM).

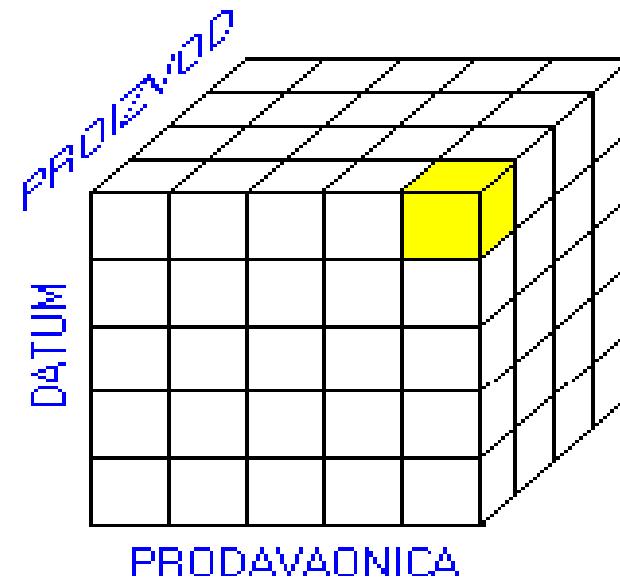
UPRAVLJANJE PODACIMA

SKLADIŠTENJE PODATAKA
(2.dio)

- ◆ Dimenzijski model podataka
 - Višedimenzionalni pogled na podatke
 - Dimenzijski konceptualni model
 - Zvjezdasta shema (dimenzijski model)
 - Izvedba upita
 - Pretvorba ER dijagrama u zvjezdaste sheme
- ◆ Različiti pristupi oblikovanju skladišta podataka

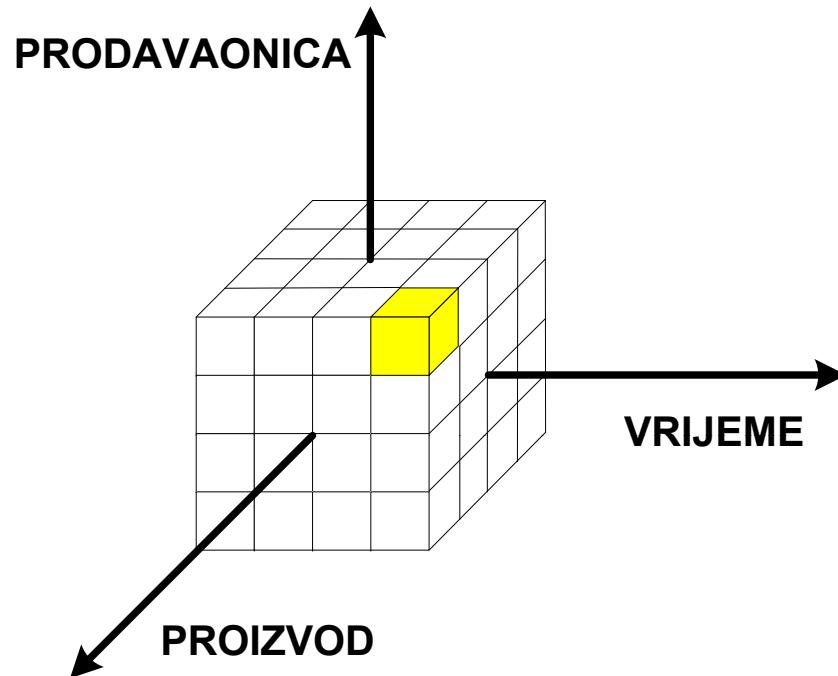
Višedimenzionalni pogled na podatke

- ♦ kod višedimenzionalnog pogleda na podatke svaki podatkovni element smješten je na presjeku dimenzijskih članova koji ga određuju
- ♦ primjer: kod analiziranja **prodaje**, dimenziije mogu biti: proizvod, vrijeme i prodavaonica
- ♦ Možemo zamisliti da je unutar svake kockice upisano koliko je komada određenog proizvoda prodano (ili kolika je bila zarada od prodaje tog proizvoda) za određeni dan u određenoj prodavaonici.



Višedimenzionalni pogled na podatke

- ◆ Vrijednosti datuma, prodavaonice i prodanog proizvoda u analizi prodaje mogu se postavljati međusobno nezavisno.
- ◆ Budući da su sve dimenzije međusobno nezavisne, one se mogu predstaviti kao n-dimenzijski koordinatni sustav.



- ◆ intuitivan koncept,
lako razumljiv analitičarima
i menadžerima

Višedimenzionalni pogled na podatke

- ◆ **OLAP alati**, koji se često koriste za analizu velikih količina podataka u skladištu podataka, pružaju krajnjem korisniku **višedimenzionalni pogled** na podatke.
- ◆ Organiziranjem i spremanjem podataka prema višedimenzionalnom konceptu omogućava se sljedeće:
 - korisnik može dobro razumjeti podatke,
 - korisnička sučelja su jednostavna za korištenje,
 - izvedba upita je na zadovoljavajućoj razini.
- ◆ Broj dimenzija je često veći od tri i tada se radi o “n-dimenzijskoj kocki” ili “hiperkocki”. Takvu strukturu nije lako vizualno predočiti, ali radi se o istom konceptu kao i za tri dimenzije.

Višedimenzionalni pogled na podatke

- ◆ Oblikovanje modela podataka za skladište podataka:
 1. **Konceptualno** – rezultat je konceptualna shema koja ne ovisi o odabranom DBMS-u
 2. **Logičko** – rezultat je logička shema - najpoznatija je zvjezdasta shema (eng. *star schema*)
 3. **Fizičko** – ovisi o odabranom DBMS-u i bavi se npr. optimalnim izborom indeksa

- ◆ Oblikovanje skladišta podataka (Golfarelli – Rizzi):
 - Analiza informacijskog sustava
 - Specifikacija korisničkih zahtjeva
 - Konceptualno oblikovanje
 - Definiranje skupa najčešćih upita i procjena količine podataka
 - Logičko oblikovanje
 - Izvlačenje, transformacija i učitavanje podataka
 - Fizičko oblikovanje

Modeli podataka za skladište podataka

Konceptualni višedimenzionalni model



- ◆ Nekoliko konceptualnih višedimenzijskih modela predloženo je u znanstvenoj literaturi:
 - Datta, H. Thomas. A Conceptual Model and Algebra for On-Line Analytical Processing in Data Warehouses. In Proc. of Workshop on Information Technology and Systems (WITS'97), Atlanta, USA, 1997.
 - P. Vassiliadis. Modeling multidimensional databases, cubes and cube operations. In Proc. of *10th Int'l. Conf. on Scientific and Statistical Database Management (SSDBM)*, Capri, Italy, 1998.
 - W. Lehner. Modelling large scale OLAP scenarios. In *Proc. of Int'l Conf. on Extending Database Technology EDBT'98, Lecture notes in Computer Science (LNCS)*, Springer, 1998.

Modeli podataka za skladište podataka

Konceptualni višedimenzionalni model

- ◆ M. Golfarelli, D. Maio, S. Rizzi, “The Dimensional Fact Model: a Conceptual Model for Data Warehouses”, *International Journal of Cooperative Information Systems*, vol. 7, 1998.
- ◆ [TBC99] N. Tryfona, F. Busborg, J. Christiansen. starER: A Conceptual Model for Data Warehouse Design. In Proc of the *ACM 2nd Int'l Workshop on Data Warehousing and OLAP (DOLAP'99)*, Kansas City, USA, 1999.
- ◆ E. Franconi, U. Sattler. A data warehouse conceptual model for multidimensional aggregation. In *Proc. of 1st Int'l Workshop on Design and Management of Data Warehouses (DMDW'99)*, Heidelberg, Germany, 1999.
- ◆ B. Hüsemann, J. Lechtenbörger, G. Vossen. Conceptual data warehouse design. In *Proc. of 2nd Int'l Workshop on Design and Management of Data Warehouses (DMDW'00)*, Stockholm, Sweden, 2000.

Modeli podataka za skladište podataka

Implementacija konceptualnog modela



Kod implementacije je bitno da krajnjim korisnicima omogućuje razumijevanje podataka, da korisnička sučelja budu jednostavna za korištenje te da izvedba upita bude na zadovoljavajućoj razini.

Implementacija:

- ◆ **relacijska** baza podataka (Relational OLAP)
 - osnovna struktura je tzv. **zvezdasta shema**
- ◆ **višedimenzionalna** baza podataka (Multidimensional OLAP)
 - podaci se spremaju u višedimenzionalna polja

Modeli podataka za skladište podataka

Implementacija konceptualnog modela

ROLAP (*Relational OLAP*)

- ◆ svi podaci se pohranjuju u relacijsku bazu podataka
- ◆ pri izvedbi upita OLAP alati generiraju SQL naredbe na temelju meta podataka.
- ◆ od OLAP alata se zahtijeva da se analiza može izvoditi brzo i to s jednakim mogućnostima i jednakom lakoćom što se tiče svake od dimenzija.
- ◆ normalizirana relacijska baza podataka dobivena po pravilima ER oblikovanja za takvu zadaću nije prikladna
- ◆ koristi se posebni **model kojim se oponaša podatkovna struktura višedimenzionalnog polja** i koji je **optimiziran za brzu i jednostavnu izvedbu složenih višedimenzijskih upita – zvjezdasta shema** (ili **zvjezdasti spoj** ili najkraće: **zvijezda**).

Modeli podataka za skladište podataka

Implementacija konceptualnog modela

MOLAP (*Multidimensional OLAP*)

- ◆ podaci se pohranjuju u bazu podataka koja nije relacijska i koja je posebno optimizirana za pohranu podataka namijenjenih za višedimenzionalnu analizu.
- ◆ u takvoj višedimenzionalnoj bazi podataka (*Multidimensional Database* – MDDB) podaci se pohranjuju u višedimenzionalna polja na način koji odražava konceptualni pogled u obliku kocke
- ◆ pri izvedbi upita na temelju vrijednosti dimenzija zadanih upitom izračunavaju se pozicije elemenata polja koji sadrže tražene podatke.
- ◆ Problemi:
 - velik dio elemenata polja je prazan
 - ne postoji standard za pohranu i upite

HOLAP (*Hybrid OLAP*)

- ◆ Riječ je o kombinaciji MOLAP i ROLAP načina rada.
- ◆ Budući da **relacijska baza ima veći kapacitet, a višedimenzijska bržu izvedbu upita**, detaljni se podaci spremaju u relacijskoj bazi (tj. skladištu podataka), a oni sumarni podaci koji se u upitima često koriste spremaju se u višedimenzionalnu bazu podataka.

Modeli podataka za skladište podataka

Dimenzijski model



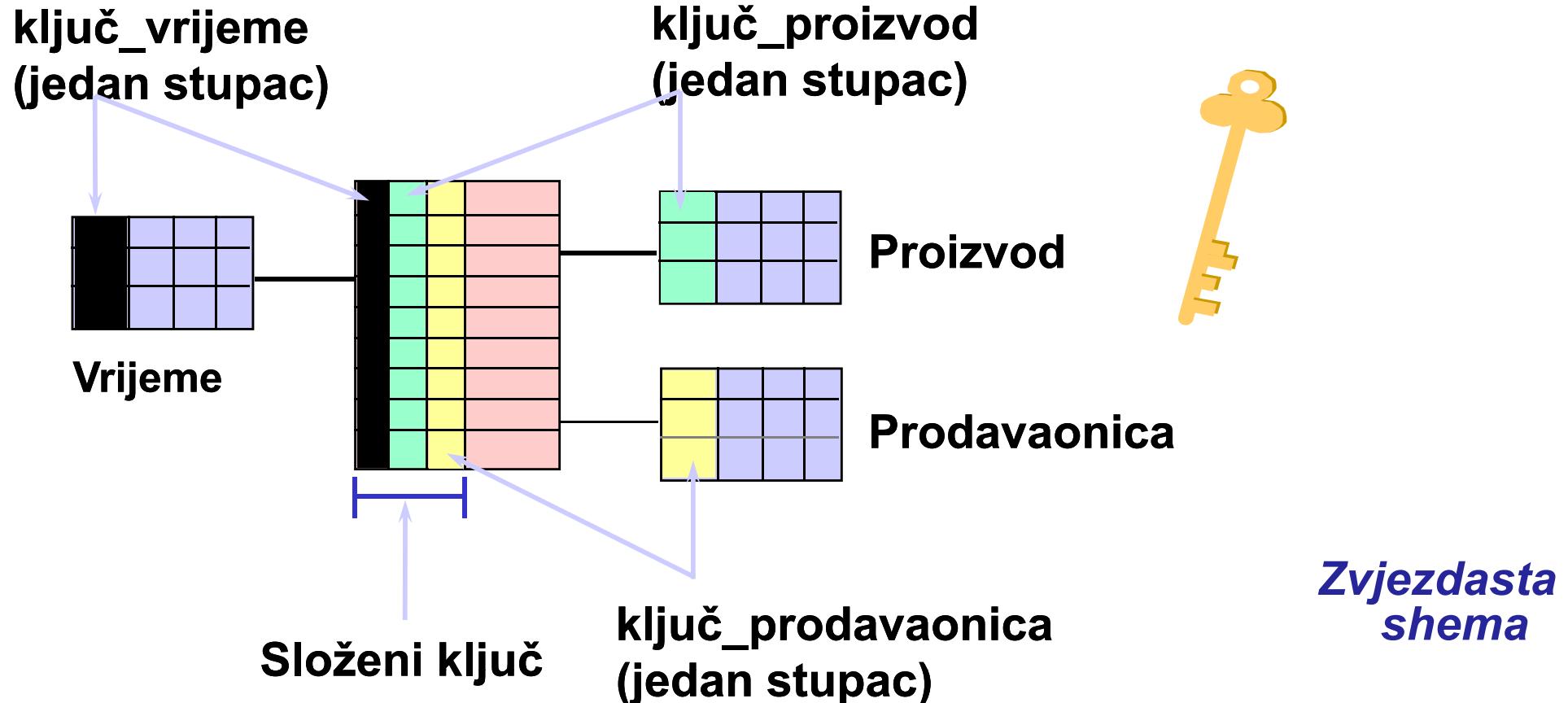
Najjednostavnija izvedba višedimenzionalnog konceptualnog modela u **relacijskoj** tehnologiji jest **zvjezdasta shema** (eng. *star schema*).

Zvjezdasta shema se sastoji od:

- jedne velike središnje tablice (tzv. **činjenične tablice**),
- **više dimenzijskih tablica.**

Modeli podataka za skladište podataka

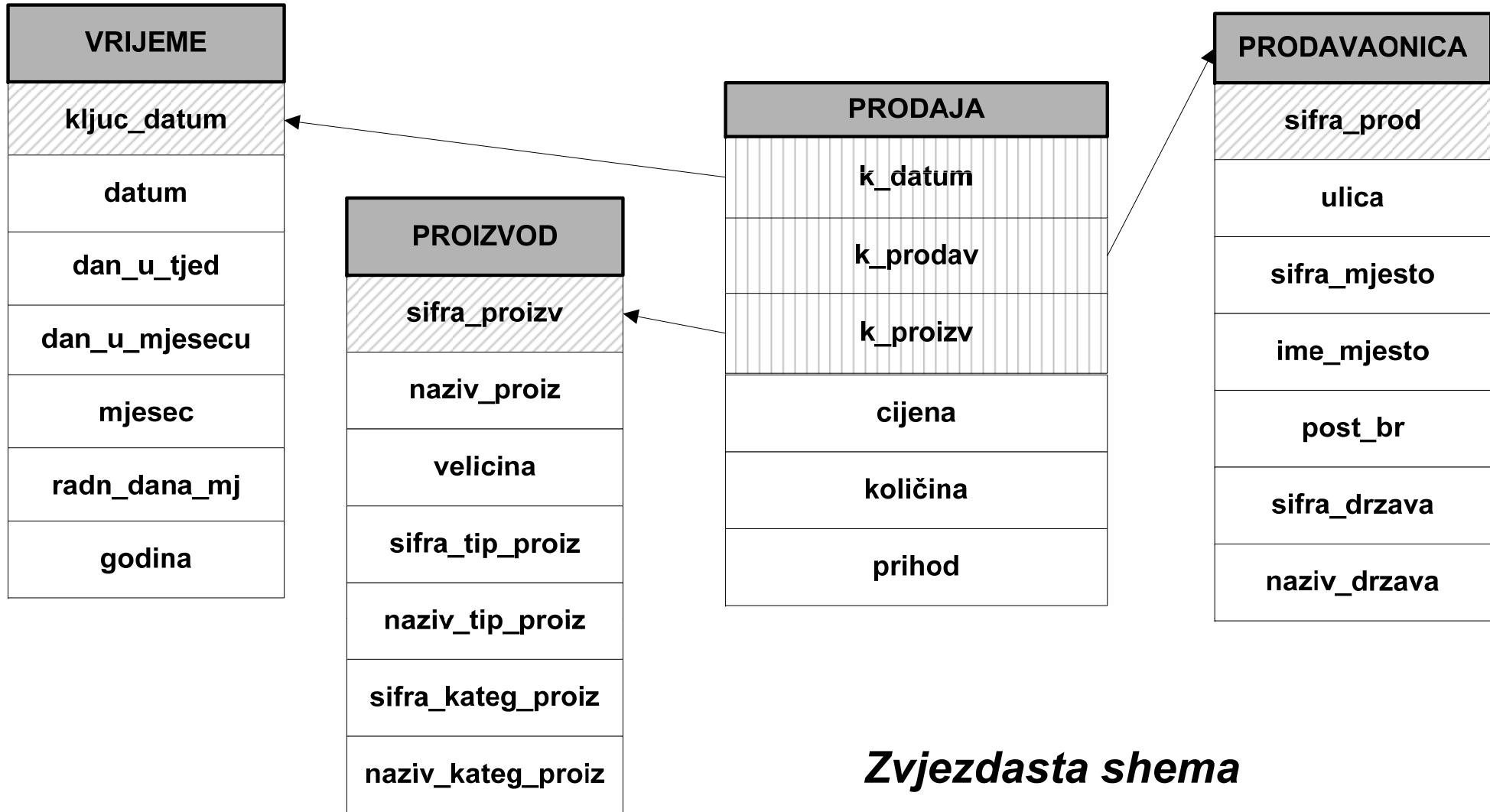
Dimenzijski model



- ♦ Svi strani ključevi na dimenzijske tablice zajedno čine **složeni primarni ključ činjenične tablice**. To znači da je redak činjenične tablice određen kombinacijom primarnih ključeva svih dimenzijskih tablica.

Modeli podataka za skladište podataka

Dimenzijski model



Modeli podataka za skladište podataka

Dimenzijski model

- ◆ **Činjenična tablica** sadrži po jedan **strani ključ** na svaku od dimenzija koje opisuju činjenicu.
- ◆ U **činjeničnoj tablici** (engl. *fact table*) nalaze se također **mjere**.

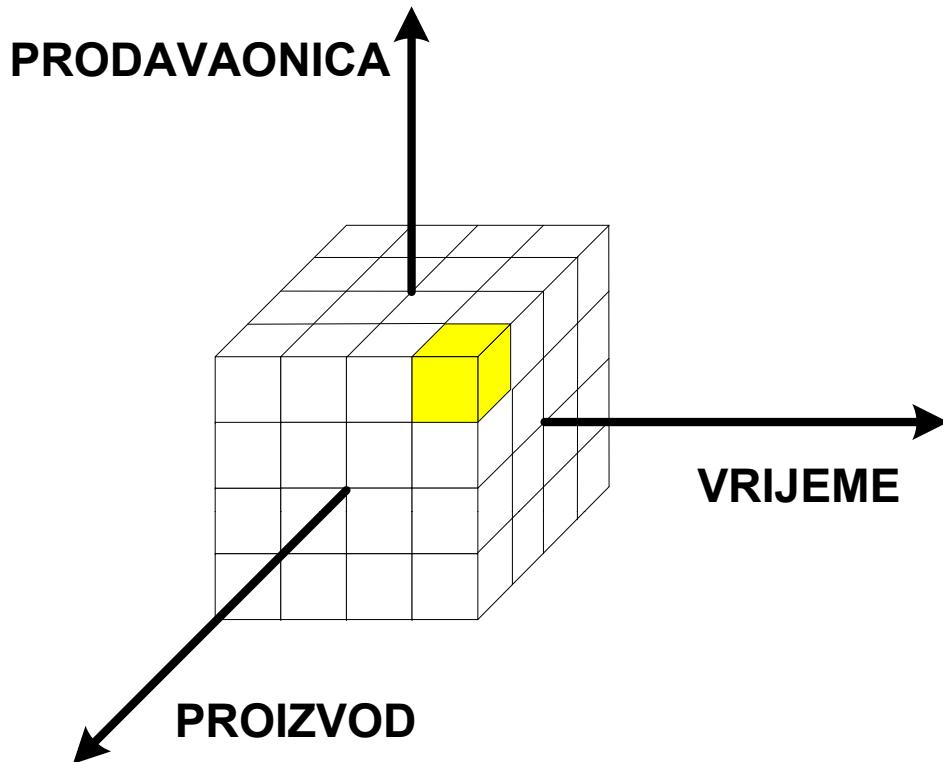
| PRODAJA |
|----------|
| k_datum |
| k_prodav |
| k_proizv |
| cijena |
| kolicina |
| prihod |

Modeli podataka za skladište podataka

Dimenzijski model

Zvjezdasta shema

- Svaki **redak činjenične tablice** odgovara jednoj kockici unutar kocke podataka. Vrijednosti upisane unutar kockice (određene koordinatama n-dimenzionalnog prostora) su vrijednosti atributa mera u činjeničnoj tablici.



| PRODAJA |
|----------|
| k_datum |
| k_prodav |
| k_proizv |
| cijena |
| količina |
| prihod |

Modeli podataka za skladište podataka

Dimenzijski model



Zvezdasta shema

- ♦ **atributi u dimenzijskim tablicama** su najčešće:
 - **tekstualni**
 - **diskretni** (tj. imaju određen skup mogućih vrijednosti
 - npr. atribut *dan_u_tjednu* ima 7 mogućih vrijednosti)

Dimenzijski model

- ◆ **Vremenska dimenzija** je gotovo uvijek nazočna u skladištu podataka, jer je zapravo svako skladište podataka vremenski niz.
- ◆ Da nema eksplisitne vremenske dimenzije, pri postavljanju upita mogla bi se postavljati ograničenja preko vremenskog ključa datumskog tipa u činjeničnoj tablici i tako bismo mogli analizirati poslovanje po mjesecima i godinama.
- ◆ Međutim, ako želimo npr. u analizi suprotstaviti radne dane neradnim danima i praznicima ili analizirati po sezonomama, tada nam treba eksplisitna dimenzija vremena.

Modeli podataka za skladište podataka

Dimenzijski model



Vremenska dimenzija može sadržavati atribute kao što su:

- datum
- dan u tjednu
- broj tjedna (npr. tjedan 12)
- mjesec
- tromjesečje
- indikator praznika
- radni dan / vikend
- specijalni događaj
- sezona
- ...

Modeli podataka za skladište podataka

Dimenzijski model

- ◆ preporučuje se za vremensku i ostale dimenzije koristiti **surogatni ključ**
- ◆ mada postoji atribut *datum* tipa DATE, preporučuje se za ključ vremenske dimenzije nametnuti atribut tipa INTEGER (*kljuc_datum*)
- ◆ surogatni ključevi trebaju biti **negovoreće šifre** cijelobrojnog tipa podataka
- ◆ vrijednosti surogatnih ključeva pridružuju se slijedno (npr. 1, 2, 3...)

Zašto su korisni surogatni ključevi?

- ◆ zaštita od promjena u izvornim sustavima (izmjene ključeva, šifri, tipova podataka)
- ◆ važni su za “sporopromjenjive dimenzije”



Modeli podataka za skladište podataka

Dimenzijski model

- ◆ **Hijerarhija** je vrlo važna u dimenzijskom modeliranju, budući da ona omogućuje dobivanje detaljnijeg ili sumarnijeg višedimenzijskog pogleda na podatke.
- ◆ Npr. u dimenzijskoj tablici **VRIJEME** hijerarhija je definirana preko razina:
datum, mjesec, tromjesečje i godina
- ◆ Krajni korisnik će obično prvo promatrati sumarne podatke, a zatim će dio podataka gledati detaljnije.
- ◆ **Dimenzijske tablice - denormalizirane** radi jednostavnosti dizajna i učinkovitijeg izvođenja upita
 - u dimenzijskim tablicama mogu postojati tranzitivne funkcione zavisnosti (tj. dimenzijske tablice **nisu u 3. normalnoj formi**).

Modeli podataka za skladište podataka

Dimenzijski model

- ◆ **DIMENZIJSKU TABLICU** čini **niz međusobno funkcijски zavisnih atributa** koji činjenicu opisuju na različitima razinama detaljnosti.

Na primjer, **dimenzijska tablica VRIJEME** sadrži:

- ◆ atribut **kljuc_datum** (na najdetaljnijoj razini - određuje razinu zrnatosti),
- ◆ atribut **mjesec** (koji funkcijski ovisi o atributu **kljuc_datum**),
- ◆ atribut **godina** (funkcijski ovisi o atributu **mjesec** i tranzitivno o atributu **kljuc_datum**).

| VRIJEME |
|---------------|
| kljuc_datum |
| datum |
| dan_u_tjed |
| dan_u_mjesecu |
| mjesec |
| radn_dana_mj |
| godina |

Dimenzijski model

- ◆ U **dimenzijsku tablicu vremena** često se na osnovnoj razini pohranjuju zapisi koji predstavljaju dane (tj. datume) i koji pokrivaju nekoliko proteklih i nekoliko idućih godina.
- ◆ Ako je potrebno pohranjivati podatke po satima, obično se u tim slučajevima stvara **nova dimenzijska tablica za sate**.
- ◆ Dimenzija može paralelno imati više hijerarhija. Osim hijerarhije *datum-mjesec-tromjesečje-godina*, može se raditi sumiranje i po tjednima ili posebno definiranim sezonom.

Modeli podataka za skladište podataka

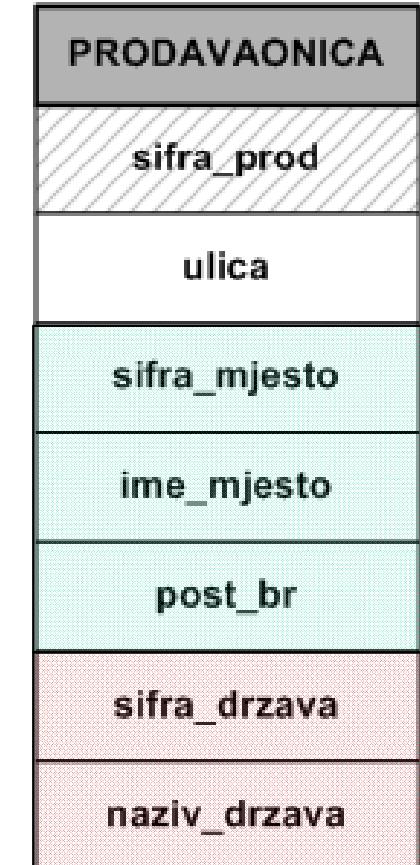
Dimenzijski model

- ♦ **funkcijske zavisnosti** predstavljaju **veze** među atributima tipa **više-prema-jedan** (kao što je npr. veza između atributa SIF_PROD i SIFRA_MJESTO te SIF_MJESTO i SIFRA_DRZAVA)
- ♦ takve veze omogućavaju **fleksibilno sumiranje podataka u upitima**

SIF_PROD -> SIFRA_MJESTO

SIF_MJESTO -> SIFRA_DRZAVA

- ♦ Dimenzijska tablica PRODAVAONICA ima tri **hijerarhijske razine**:
 1. PRODAVAONICA
 2. MJESTO
 3. DRŽAVA



Modeli podataka za skladište podataka

Dimenzijski model



- U dimenziji PROIZVOD najdetaljniju razinu određuje upravo proizvod, dok se zbrajanje može izvršiti s obzirom na tip proizvoda, a zatim i po kategoriji proizvoda.

SIFRA_PROIZV -> SIFRA_TIP_PROIZ

SIFRA_TIP_PROIZ -> SIFRA_KATEG_PROIZ

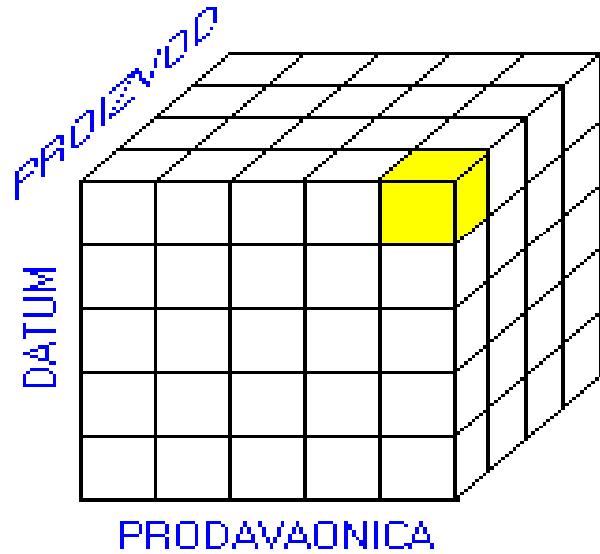
Napomena: u primjeru se uzima da svaki proizvod spada u samo jedan tip proizvoda te da svaki tip proizvoda spada u samo jednu kategoriju proizvoda)

- Pri dimenzijskom modeliranju potrebno je **identificirati funkcione zavisnosti!**



Modeli podataka za skladište podataka

Dimenzijski model



- ◆ **Ključevi dimenzijskih tablica** određuju **zrnatost** (granularnost, razinu detalja) kojom se opisuju mjere.
- ◆ Ako se prodaja opisuje pomoću dimenzija proizvod, prodavaonica i datum, **najdetaljnija informacija** koju možemo dobiti je:
koliko je određenog datuma u određenoj prodavaonici prodano određenih proizvoda.
U tom slučaju ne može se npr. dobiti informacija o broju kupljenih proizvoda u određenom satu.

Modeli podataka za skladište podataka

Dimenzijski model



- ◆ Kako su za **ključeve** svih dimenzijskih tablica uzeti temeljni atributi najdetaljnije hijerarhijske razine, ta će razina zrnatosti biti ostvarena i u činjeničnoj tablici. Upravo **zrnatost** u činjeničnoj tablici odgovara pojmu zrnatosti skladišta.
- ◆ Samo one kombinacije ključeva za koje postoje odgovarajući podaci pohranjuju se u činjeničnoj tablici te zato nema problema s pohranom praznih polja kao kod MOLAP-a.

Modeli podataka za skladište podataka

Dimenzijski model



- ◆ Zbrajanje vrijednosti mjera s obzirom na vrijednost nekog atributa u hijerarhiji na višoj razini od osnovne naziva se **agregacijom** po tom atributu.
- ◆ **Agregacijom prodaje po mjesecima zbrajaju se sve dnevne vrijednosti** prihoda i količine prodanih proizvoda koje pripadaju istom mjesecu u godini.
- ◆ Za svaki zapis u činjeničnoj tablici postoji samo jedan zapis u svakoj dimenziji. Za bilo koji zapis u bilo kojoj dimenzijskoj tablici može postojati više odgovarajućih zapisa u činjeničnoj tablici.
- ◆ Opisani međusobni odnos zapisa **omogućuje sumiranje (agregaciju) zapisa u činjeničnoj tablici.**

Modeli podataka za skladište podataka

Dimenzijski model

- ◆ **Dimenzijske tablice** sadrže i **opisne atribute** koji sadrže dodatne informacije o nekom drugom dimenzijskom atributu (npr. *NAZIV_PROIZVODA* i *VELICINA* opisuju proizvod koji je jednoznačno određen atributom *SIFRA_PROIZV*).
- ◆ Za razliku od ostalih dimenzijskih atributa, **opisni atributi se ne mogu koristiti za sumiranje podataka.**
- ◆ Opisni atributi **omogućuju kvalitetnije ograničenje (restrikciju) upita**. Funkcijski su ovisni o najdetaljnijem atributu dimenzijske tablice.

| PROIZVOD |
|-------------------|
| sifra_proizv |
| naziv_proiz |
| velicina |
| sifra_tip_proiz |
| naziv_tip_proiz |
| sifra_kateg_proiz |
| naziv_kateg_proiz |

Modeli podataka za skladište podataka

Dimenzijski model

- ◆ Svaki od glavnih atributa koji čine hijerarhiju temelj je jedne **razine hijerarhije** (engl. *hierarchy level*).
- ◆ U istoj razini hijerarhije nalaze se oni opisni atributi koji opisuju temeljni atribut te razine hijerarhije.
- ◆ Npr. atribut ***dan_u_tjed*** opisuje atribut ***kljuc_datum***, dok atribut ***naziv radn_dana_mj*** opisuje atribut ***mjesec***.



Modeli podataka za skladište podataka

Dimenzijski model



Zvezdasta shema

- ◆ **mjere (ili “činjenice”):**
 - brojčane
 - zbrojive (poželjno, jer najčešće trebamo izračunati zbroj detaljnih vrijednosti)
 - iz kontinuiranog skupa vrijednosti (vrlo širok skup vrijednosti; mjere mogu poprimiti drugačiju vrijednost svaki put kad se te vrijednosti pohranjuju)
- ◆ Mjere (engl. *measures*) su brojčani atributi s kontinuiranim (neprekidnim) skupom vrijednosti. Budući da skladište podataka sadrži ogroman broj zapisu, upravo će se zbrajanjem numeričkih podataka dobiti vrijednosti zanimljive za analizu.

Modeli podataka za skladište podataka

Dimenzijski model



- ◆ Mjere mogu biti potpuno **zbrojive, poluzbrojive i nezbrojive**.
- ◆ Potpuno **zbrojive** mjere mogu se zbrajati po svim dimenzijama koje opisuju činjenicu.
- ◆ **Poluzbrojive** mjere mogu se zbrajati samo po nekim dimenzijama. Mjera je **zbrojiva nad određenom dimenzijom** ako se njene vrijednosti mogu agregirati po pripadajućoj hijerarhiji koristeći operator zbrajanja.
- ◆ **Poželjno je da što veći broj mjera bude zbrojiv.**

Modeli podataka za skladište podataka

Dimenzijski model



- ◆ Za **nezbrojive** mjere ne može se izvršiti agregacija niti po jednoj vremenskoj dimenziji. Najčešće se radi o atributima koji bi se trebali naći u dimenzijama, ali zbog načela logičkog oblikovanja ostaju kao opis činjenice.
- ◆ Ako je činjenica nezbrojiva, poželjno je da se nad njom može provesti agregacija koristeći operatore kao što su srednja vrijednost (kao što je slučaj s mjerom cijena proizvoda), maksimalna vrijednost ili minimalna vrijednost.

Modeli podataka za skladište podataka

Dimenzijski model



Primjer za **poluzbrojivu** činjenicu:

Pretpostavimo da spremamo zapise o prodaji za proizvod A i proizvod B, te smo u prvom zapisu zabilježili da je bilo 24 kupca, a u drugom 16 kupaca.

Koliko je ljudi kupilo **bar jedan od ta dva proizvoda?**

Ne može precizno zaključiti,
taj je broj negdje između 24 i 40.

→ **broj kupaca** u činjeničnoj tablici koja sadrži podatke o *prodaji* **nije zbrojiv preko dimenzije Proizvod**.

- ◆ Svaku analizu koja koristi broj kupaca uvijek treba **ograničiti** na samo jedan proizvod. Drugim riječima, skup ključeva u nezbrojivim dimenzijama treba pri upitima ograničiti na samo jednu vrijednost.

Modeli podataka za skladište podataka

Dimenzijsko modeliranje



- ♦ **DIMENZIJSKO MODELIRANJE** je metoda logičkog oblikovanja kod koje se poslovni procesi opisuju **preko dimenzija koje su hijerarhijski organizirane**, čime se korisniku omogućava **višedimenzionalni pogled na podatke i brz pristup velikoj količini podataka**.

Modeli podataka za skladište podataka

Dimenzijski model



- ◆ Zvijezda spoj je osnovni tip sheme, no postoje i varijacije kao što je tzv. snježna pahuljica (**snowflake**).
- ◆ U **snowflake** shemi su jedna ili više dimenzija normalizirane uklanjanjem tranzitivnih funkcijskih ovisnosti.
- ◆ Mana: vrijeme potrebno za izvođenje upita raste
- ◆ Prednost: u nekim slučajevima je korisno povećati razinu normalizacije jer se time zauzima manje prostora na disku

Modeli podataka za skladište podataka

Primjer: analiza pristupa na www.hr



Primjer: analiza pristupa na www.hr - katalog hrvatskih Web stranica

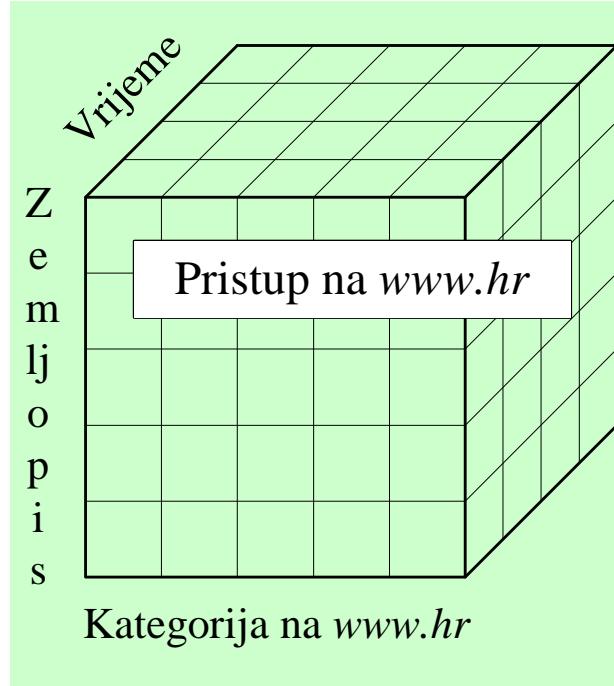
Npr. pitanje: *Koliko je bilo pristupa na web stranice u kategoriji Turizam i putovanja iz Sjeverne Amerike po mjesecima u 1998. godini?*

Dimenzije:

- ◆ **Zemljopisno područje** (Hrvatska ili kontinenti) gdje se nalazi računalo s kojeg se pristupa na web stranice kataloga
- ◆ **Kategorija** u kojoj se nalazi tražena stranica
- ◆ **Vrijeme** pristupa, tj. vremenska razdoblja unutar kojih je došlo do pristupa na www.hr stranice

Modeli podataka za skladište podataka

Primjer: analiza pristupa na www.hr



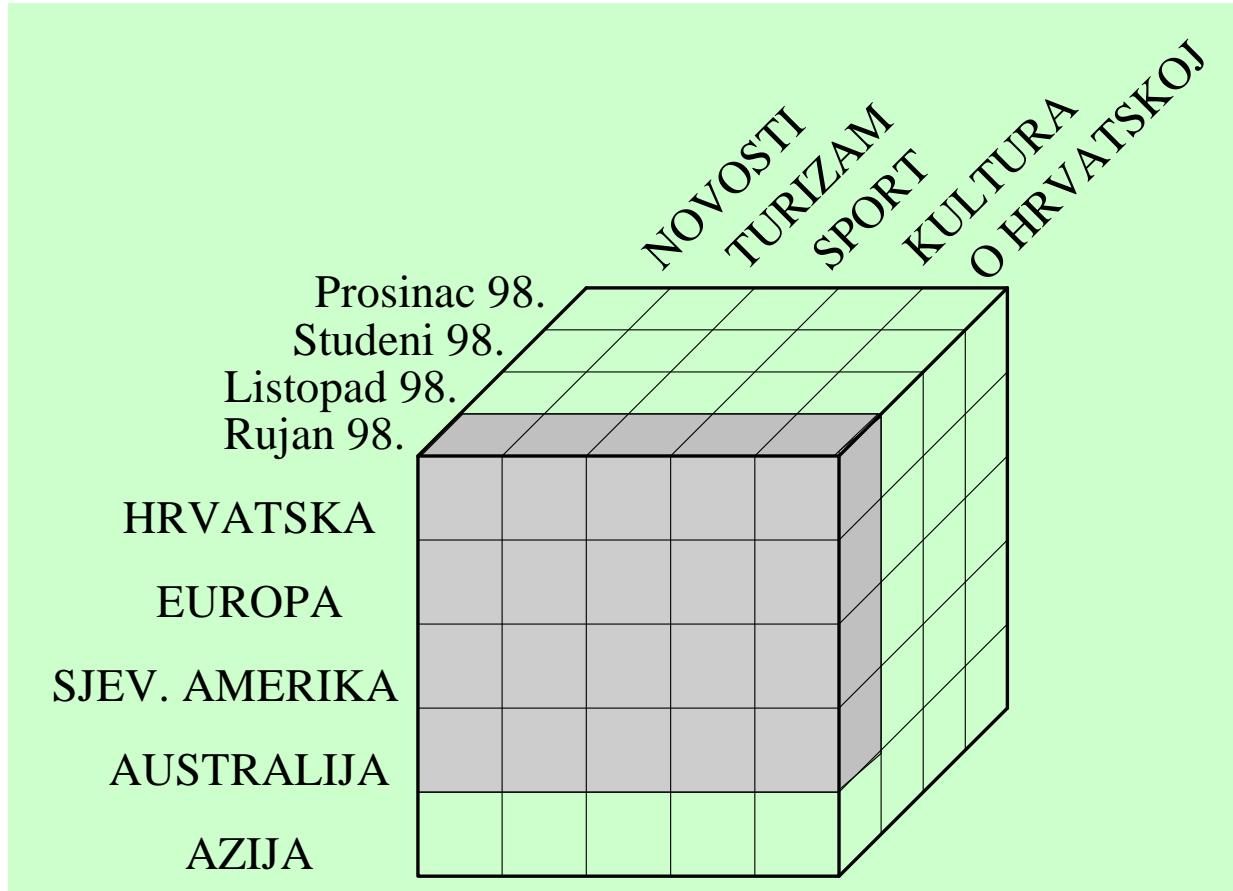
- analiza pristupa
na www.hr

Svaka kockica (tj. ćelija) unutar podatkovne kocke sadrži podatak o broju **pristupa** za određenu kombinaciju:

- ◆ **zemljopisnog područja** odakle se pristupa
- ◆ **kategorije** na koju se pristupa
- ◆ **vremenskog razdoblja** unutar kojeg se pristupa na **WWW.HR** stranice

Modeli podataka za skladište podataka

Primjer: analiza pristupa na www.hr



Operacije tipične za OLAP alate:

- promjena orientacije dimenzija (tzv. rotacija)
- postupno kretanje po podatkovnim razinama (od sumarnih ka detaljnim i obrnuto)

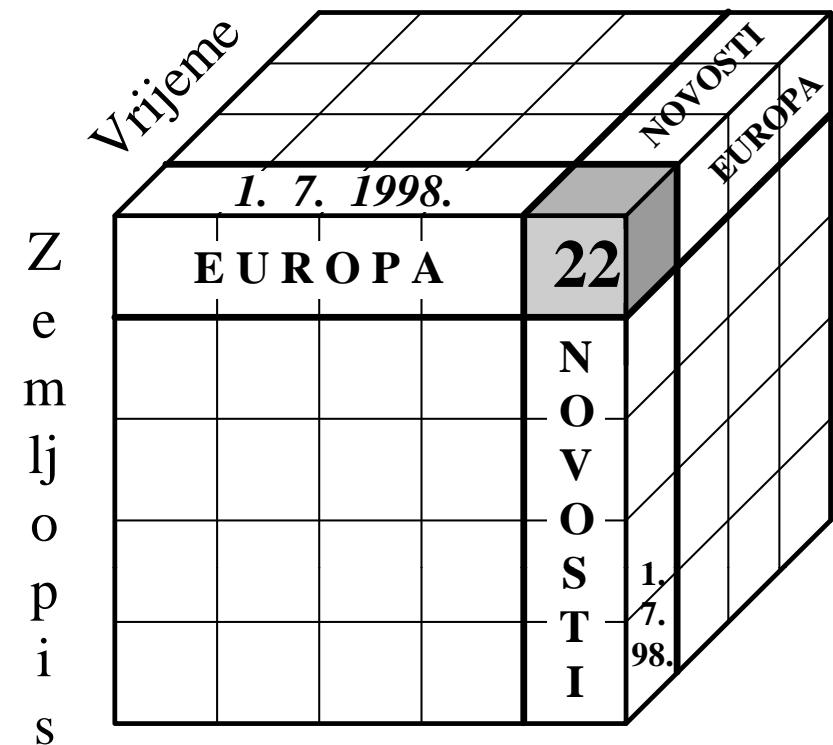
Modeli podataka za skladište podataka

Primjer: analiza pristupa na www.hr

| ključ_zemljopis | ključ_vrijeme | ključ_kategorija | broj_pristupa |
|-----------------|---------------|------------------|---------------|
| EUROPA | 01-07-1998 | NOVOSTI | 22 |
| : | : | : | : |

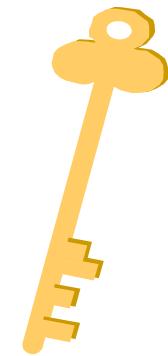
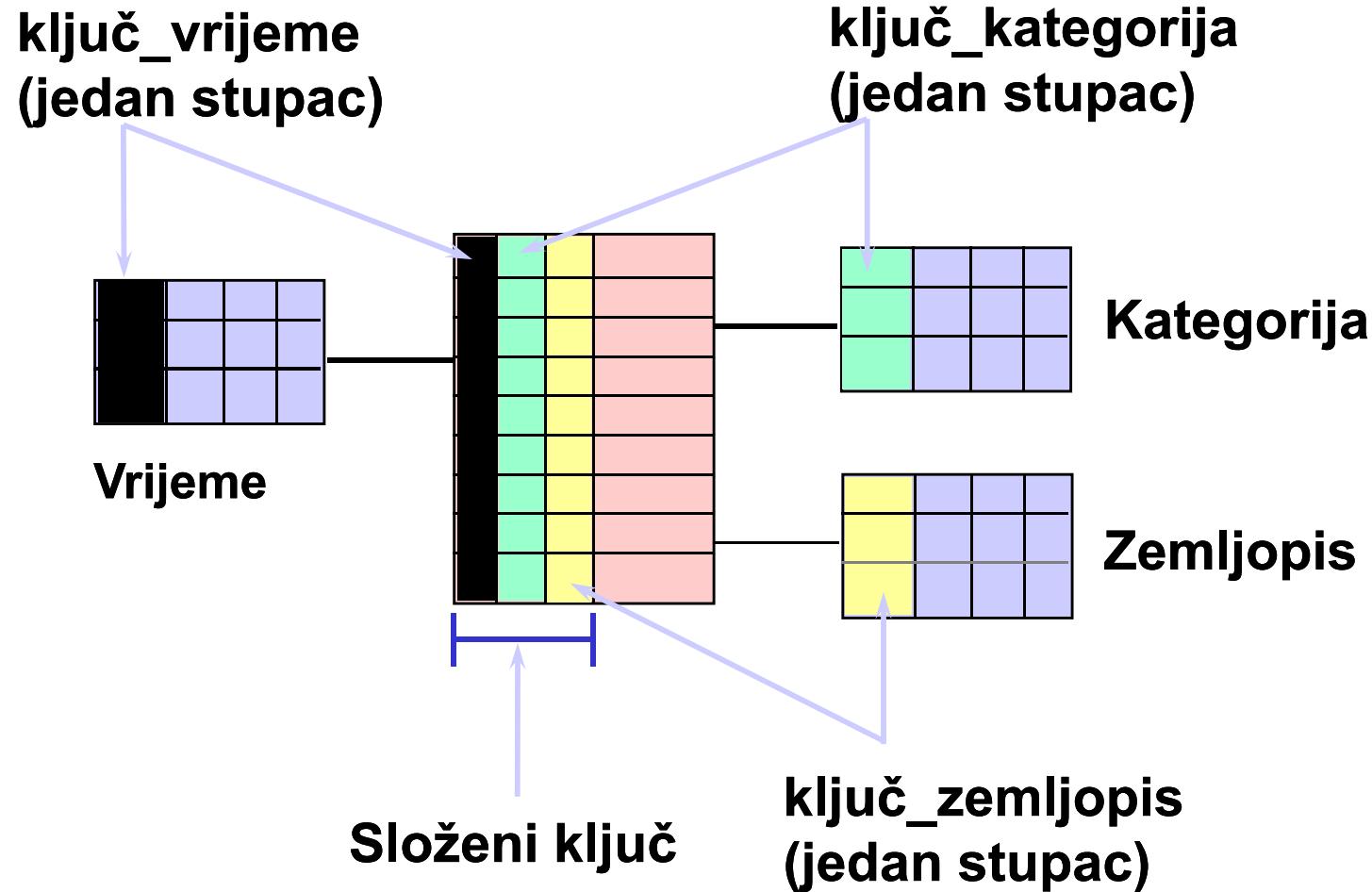
Činjenična tablica
(fact table)

Višedimenzionalni
konceptualni
pogled na podatke



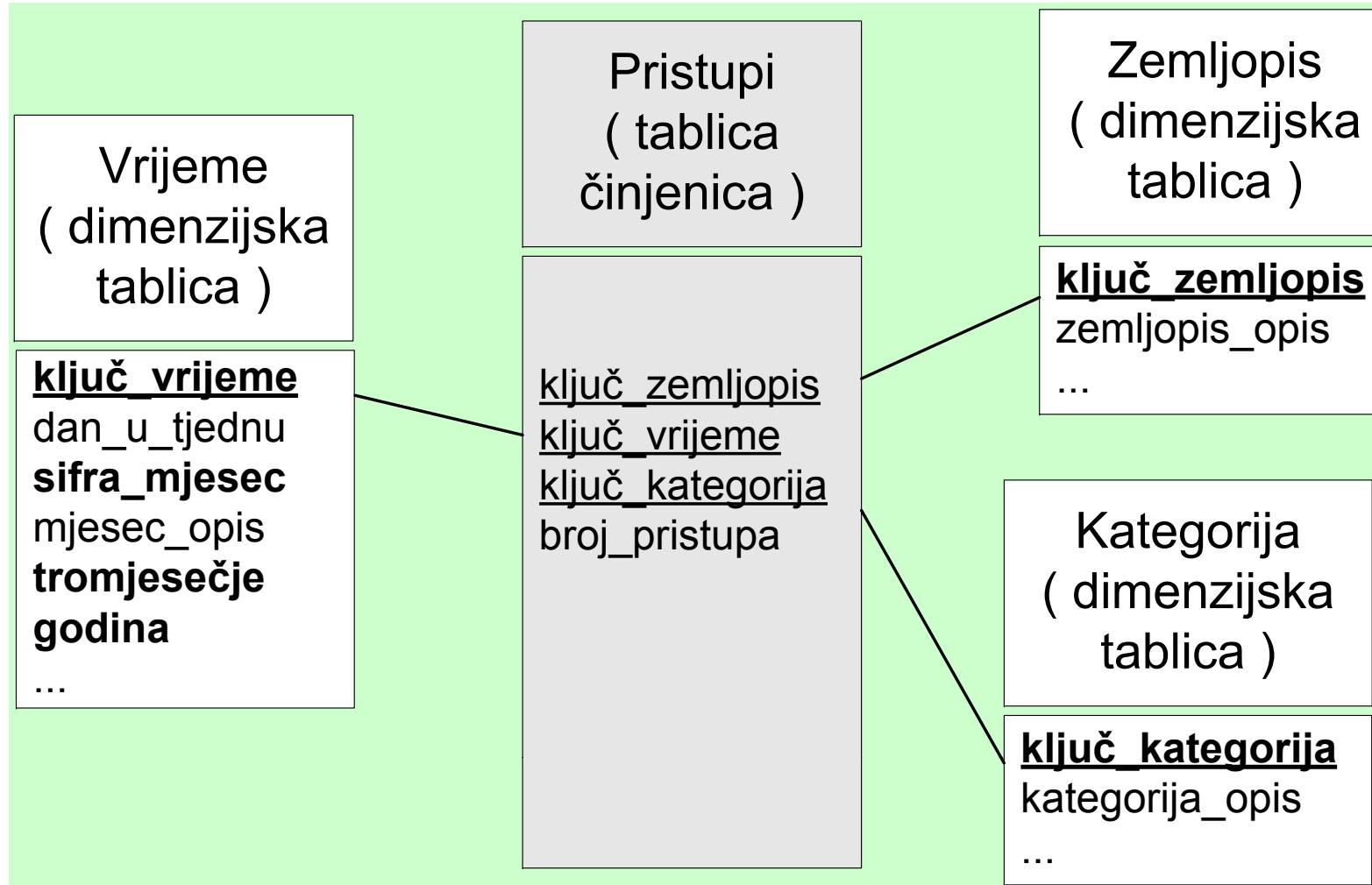
Modeli podataka za skladište podataka

Primjer: analiza pristupa na www.hr



Modeli podataka za skladište podataka

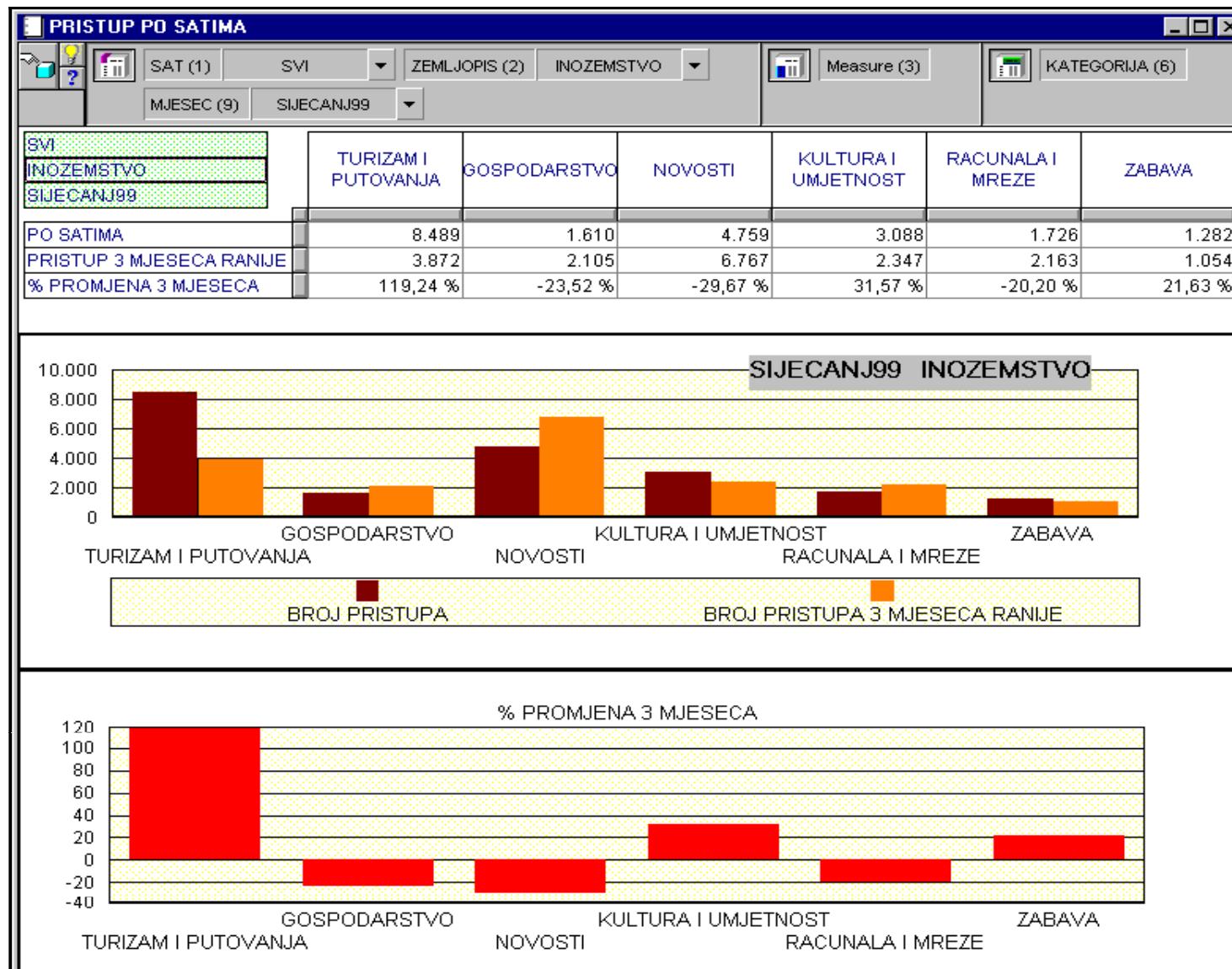
Primjer: analiza pristupa na www.hr



Zvijezda spoj koji opisuje pristupe na WWW.HR stranice

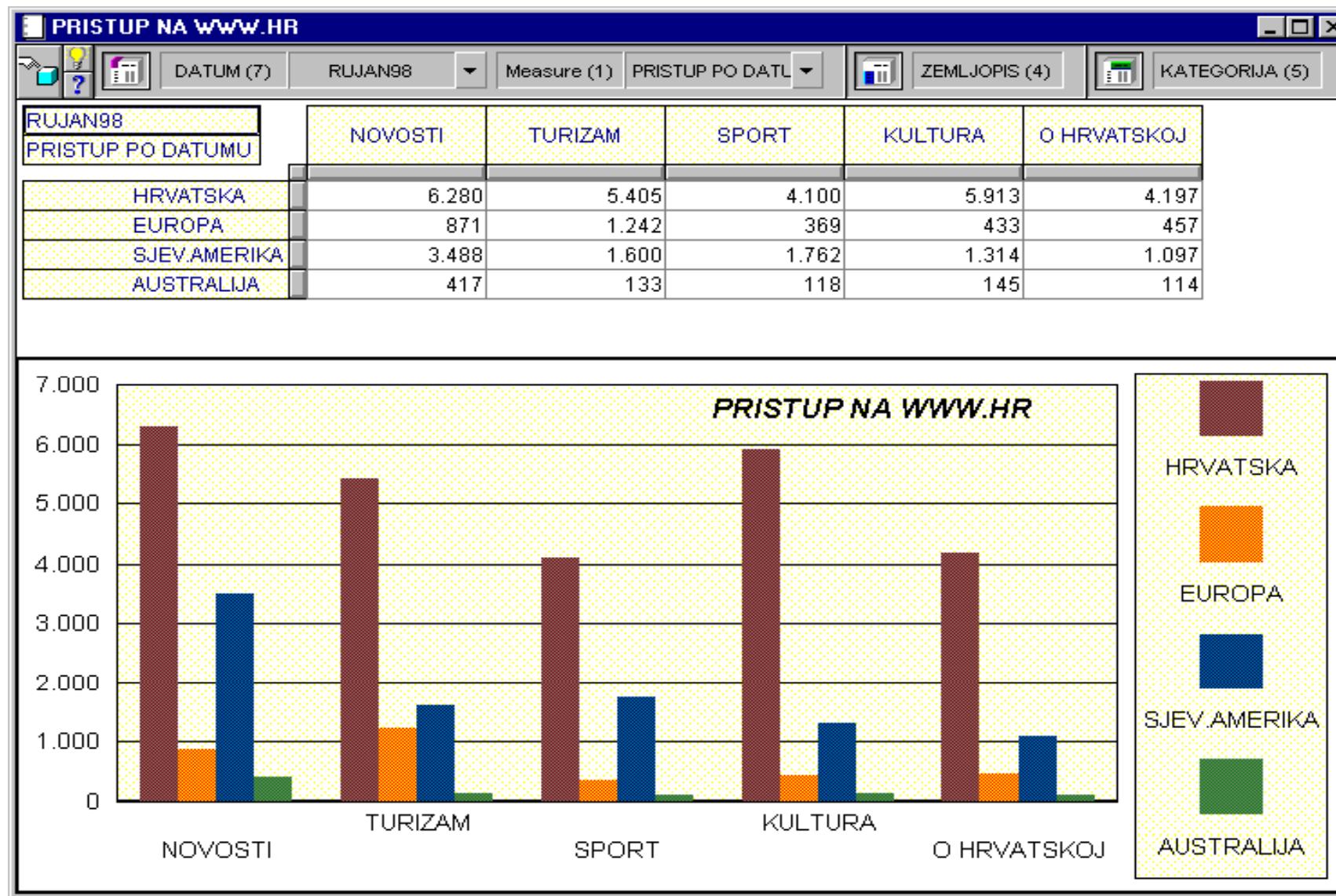
Modeli podataka za skladište podataka

Primjer: analiza pristupa na www.hr



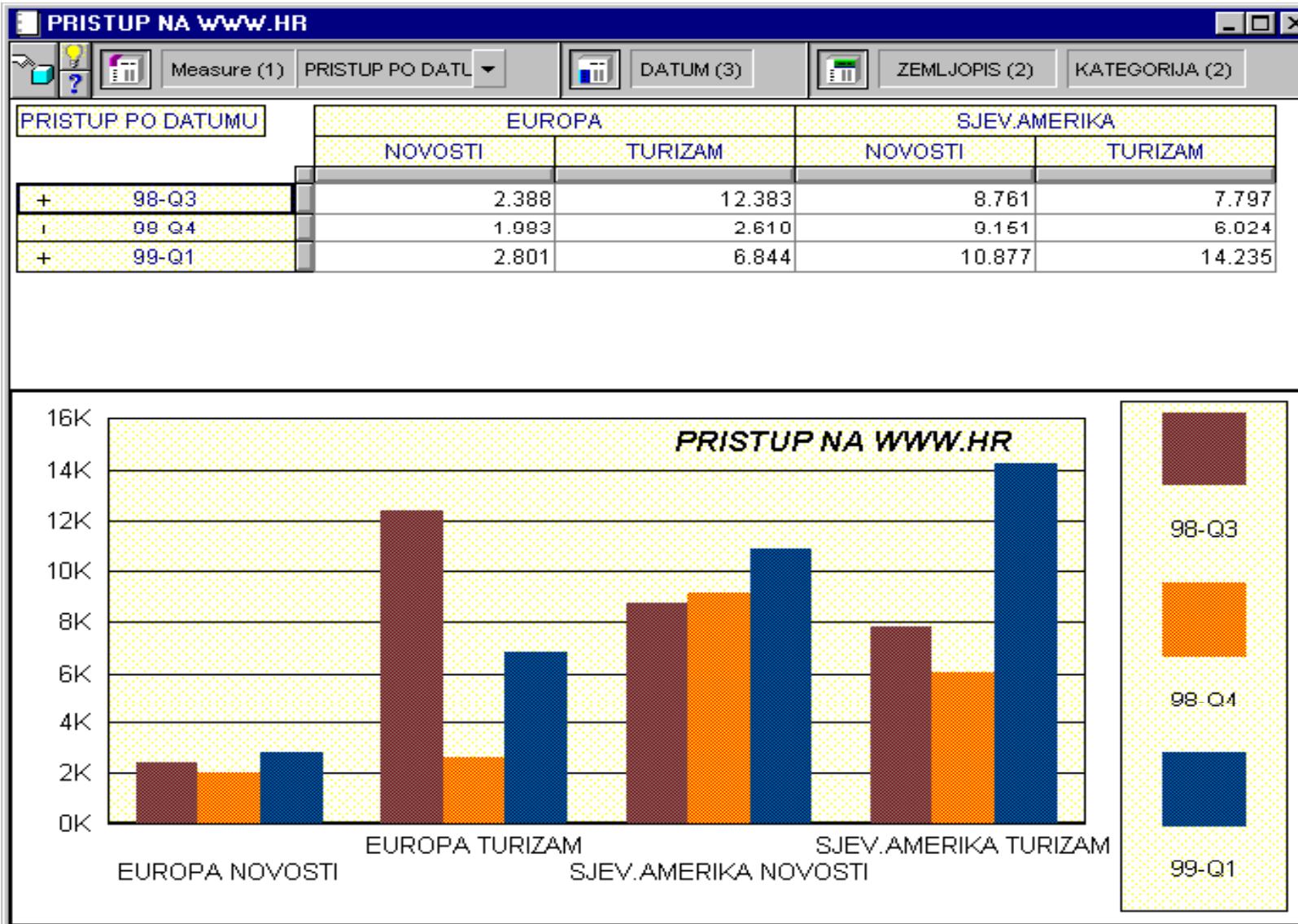
Modeli podataka za skladište podataka

Primjer: analiza pristupa na www.hr



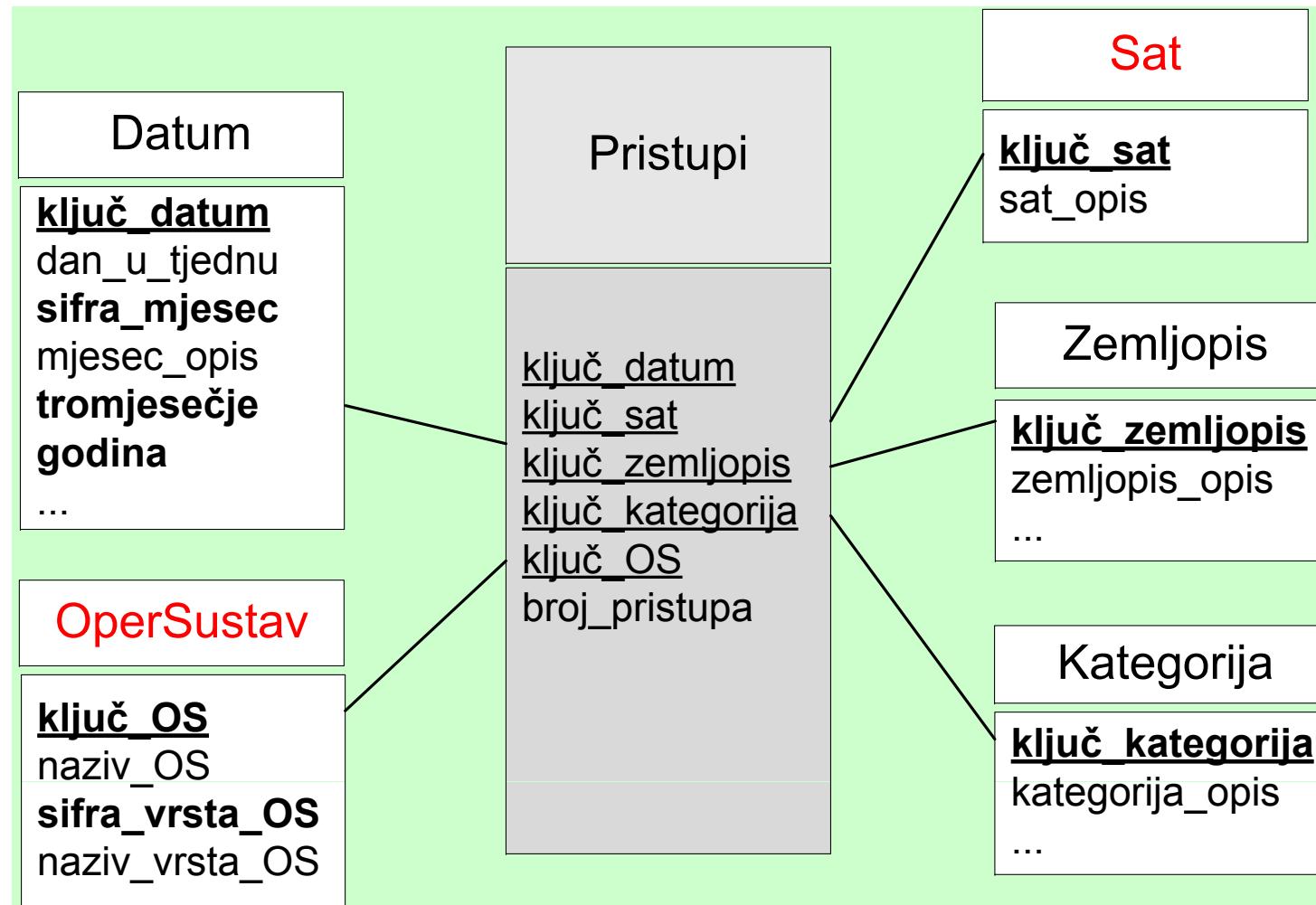
Modeli podataka za skladište podataka

Primjer: analiza pristupa na www.hr



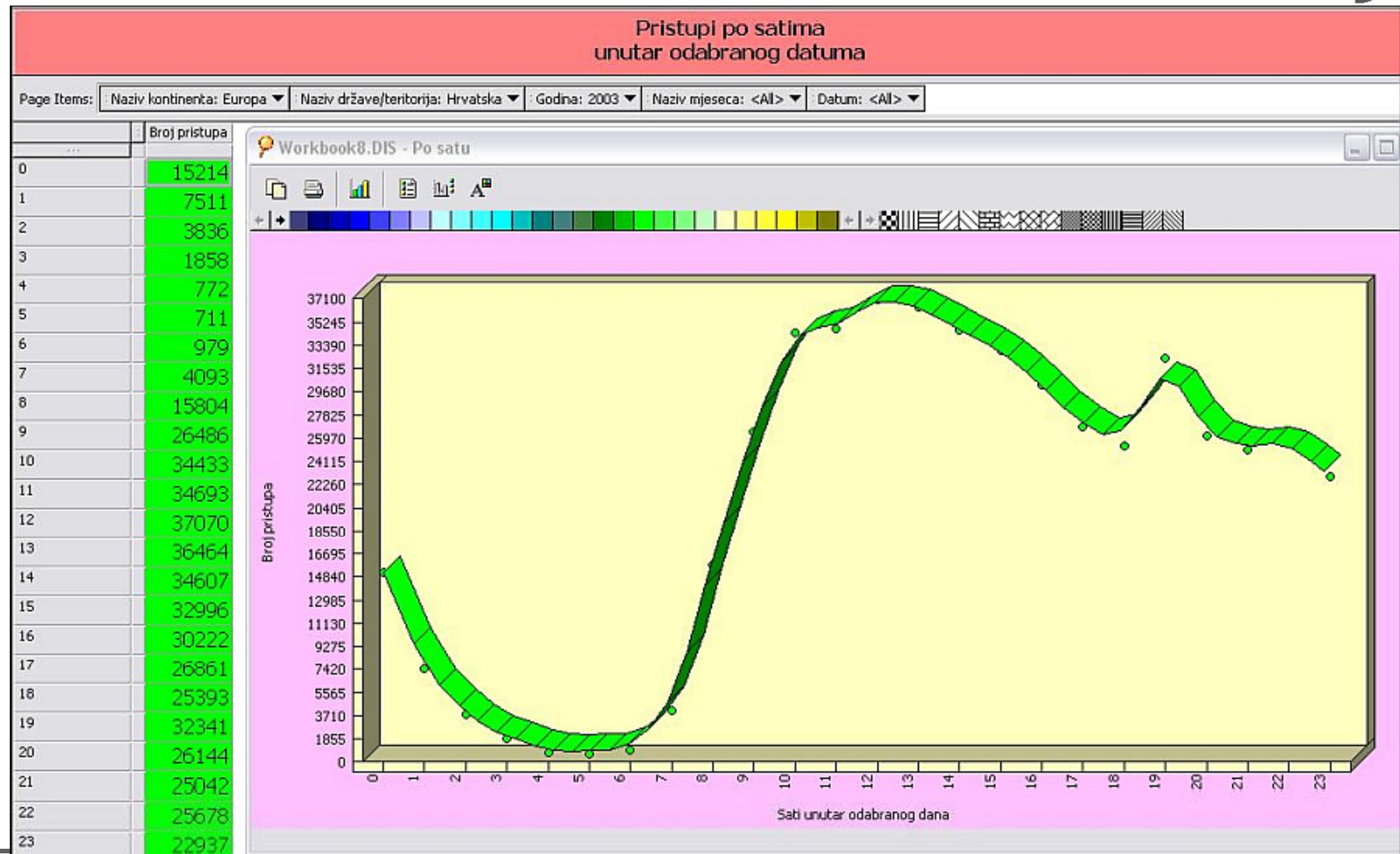
Modeli podataka za skladište podataka

Primjer: analiza pristupa na www.hr

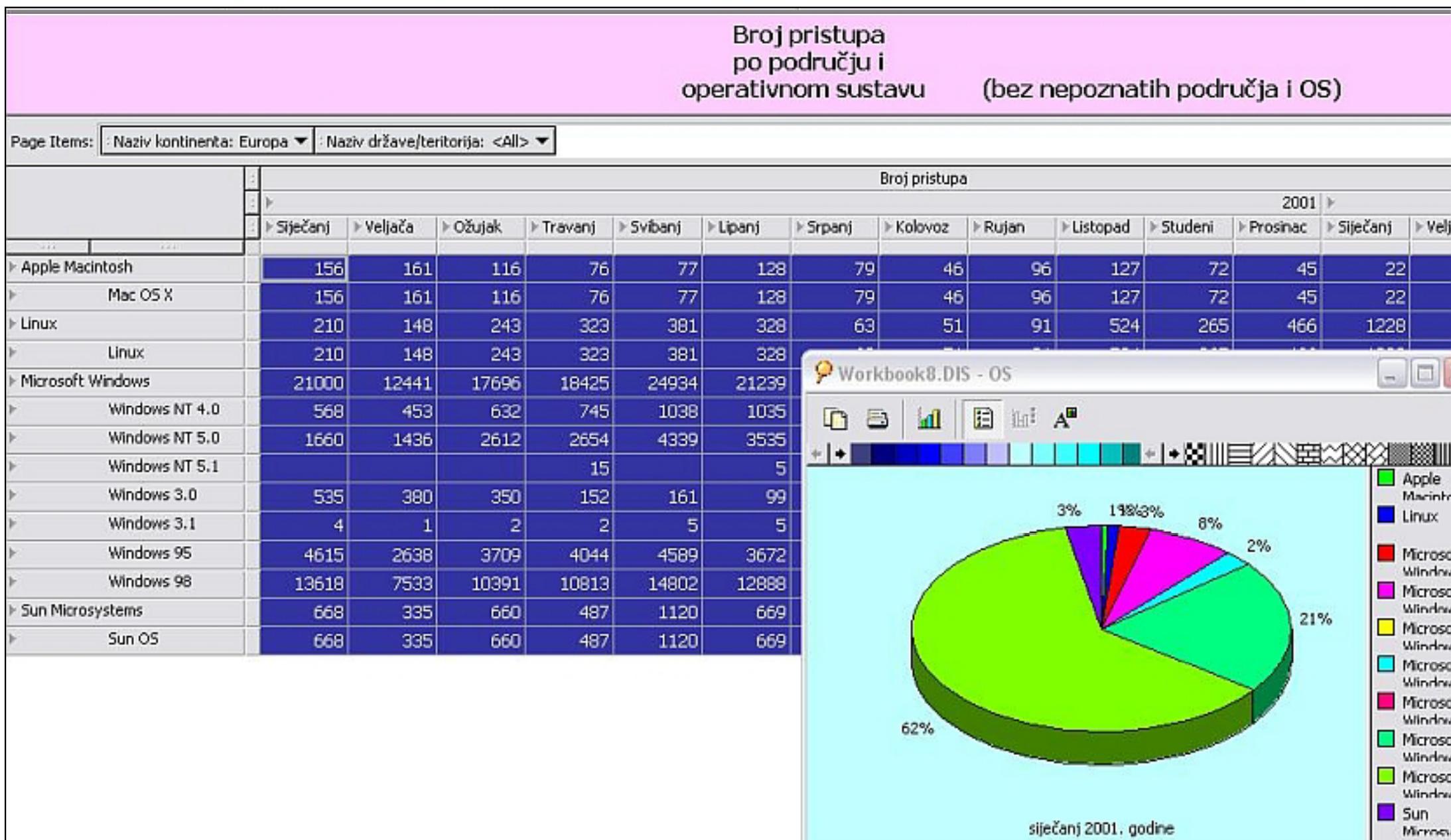


Zvijezda spoj koji opisuje pristupe na WWW.HR stranice

Modeli podataka za skladište podataka



Modeli podataka za skladište podataka



Modeli podataka za skladište podataka

Primjer: naplata parkiranja putem SMS-a

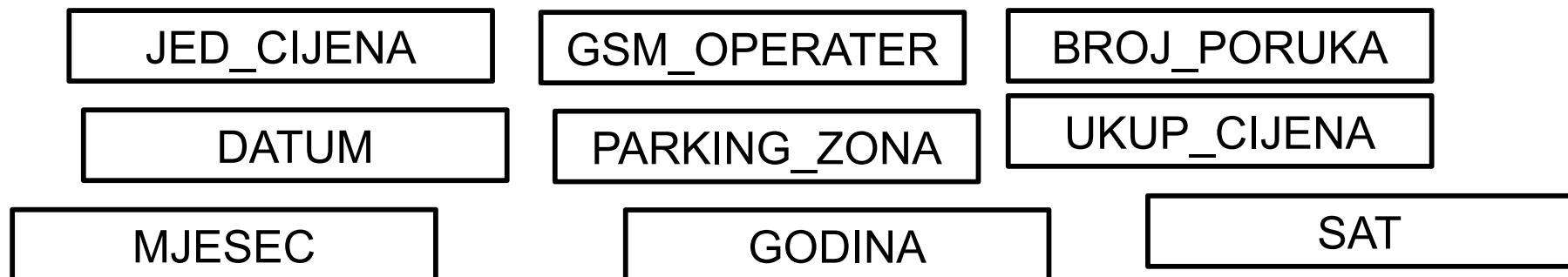
- ◆ Želimo napraviti **sustav za analizu naplate parkiranja putem SMS-a** u gradu Zagrebu,
pri čemu želimo omogućiti brzo dobivanje odgovora na ova i slična
pitanja:

- Koliko smo naplatili parkiranje u različitim parkirnim zonama po
danima od 1.2.2011. do 10.3.2011. godine?
- Koliko je u mjesecu ožujku 2011. primljeno SMS poruka za plaćanje
parkiranja od korisnika pojedinih GSM operatera?
- Koliko je u mjesecu ožujku 2010. primljeno SMS poruka za plaćanje
parkiranja po satima u različitim parkirnim zonama?
- **Oblikujte zvjezdastu shemu!**

Modeli podataka za skladište podataka

Primjer: naplata parkiranja putem SMS-a

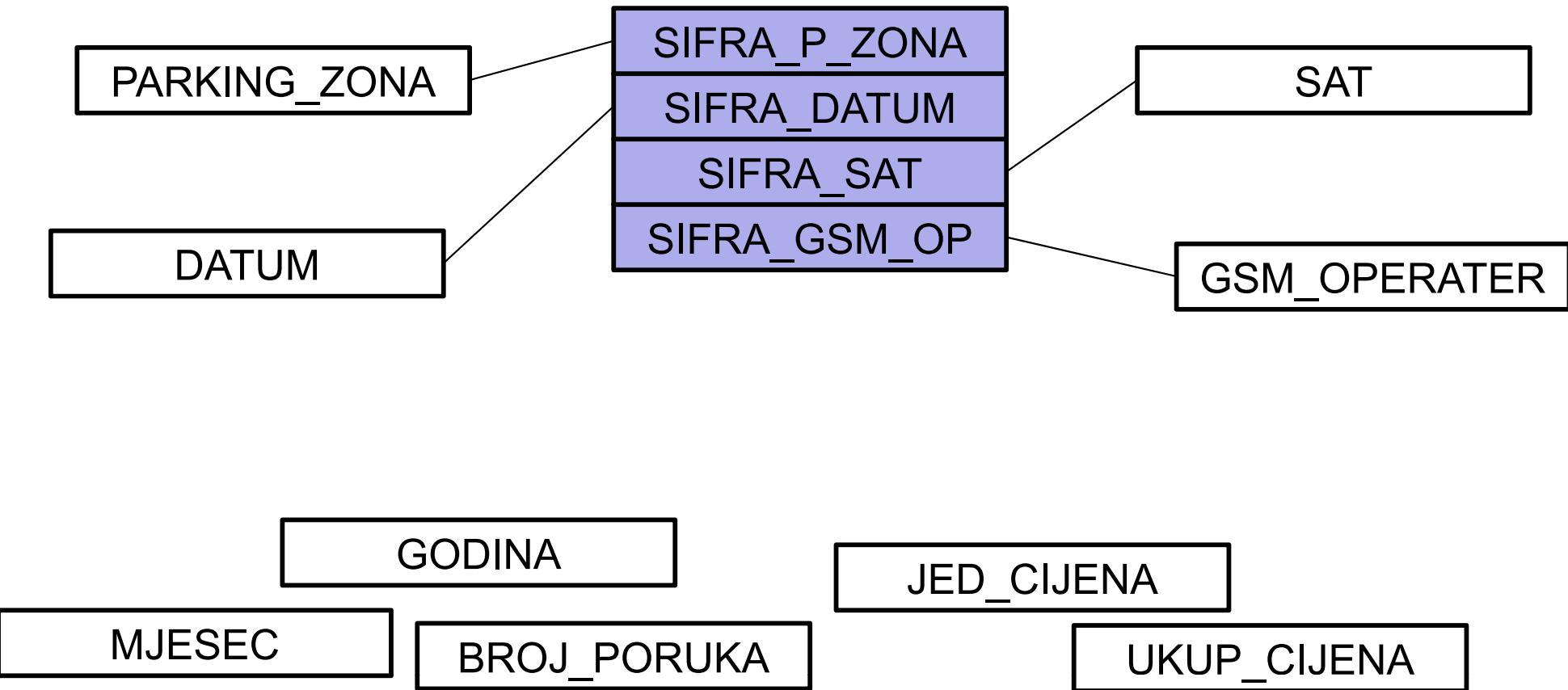
- ◆ Zajedno su prikazane **hijerarhijske razine** raznih dimenzija i **mjere**:



- ◆ Napomena: Svaka hijerarhijska razina će u pravilu sadržavati više atributa (ali te atribute u ovom primjeru ne prikazujemo).
Npr. razina PARKING_ZONA će imati bar dva atributa:
 - sifra_parking_zona
 - naziv_parking_zona
- **prvo odredite dimenzije**

Modeli podataka za skladište podataka

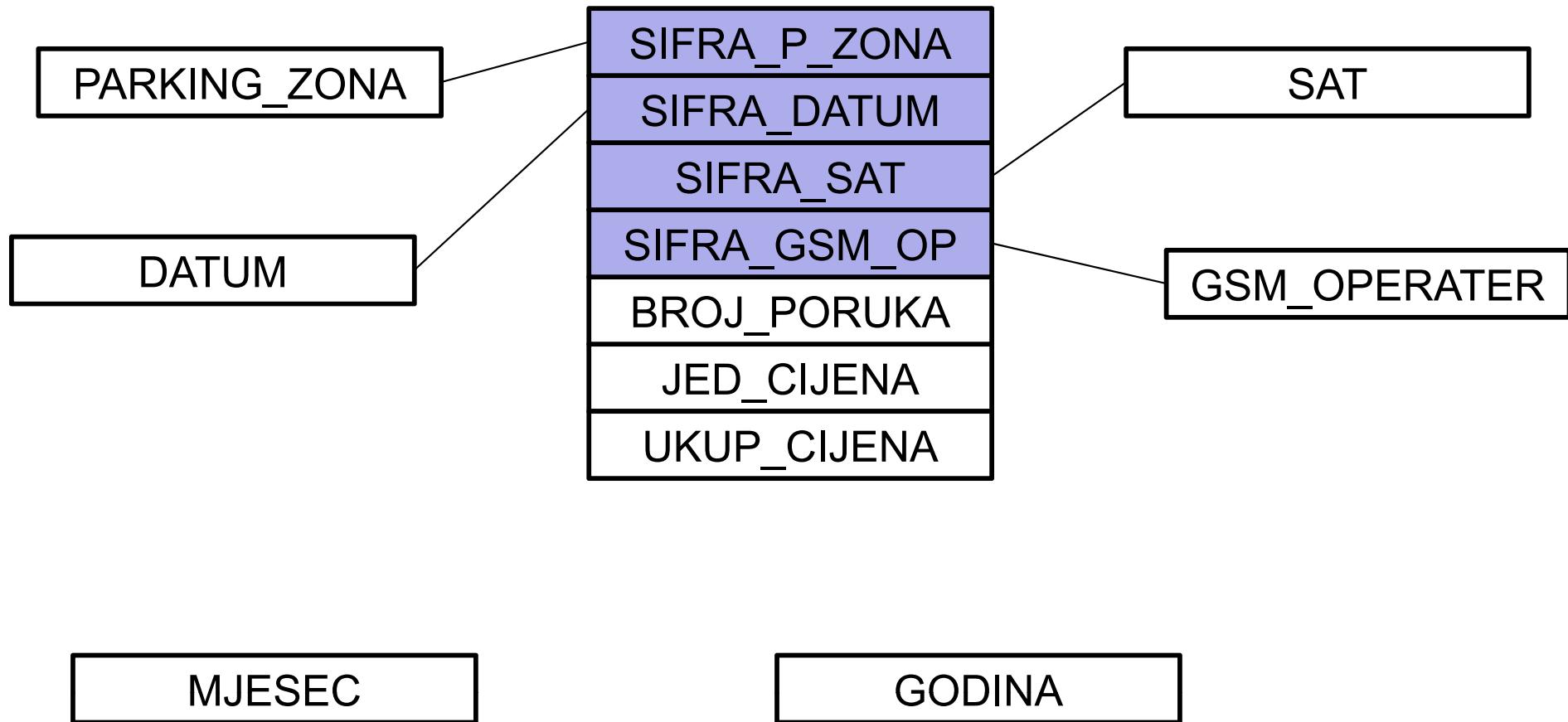
Primjer: naplata parkiranja putem SMS-a



- ♦ zatim definirajte mjere u činjeničnoj tablici i na kraju definirajte hijerarhijske razine u dimenzijama

Modeli podataka za skladište podataka

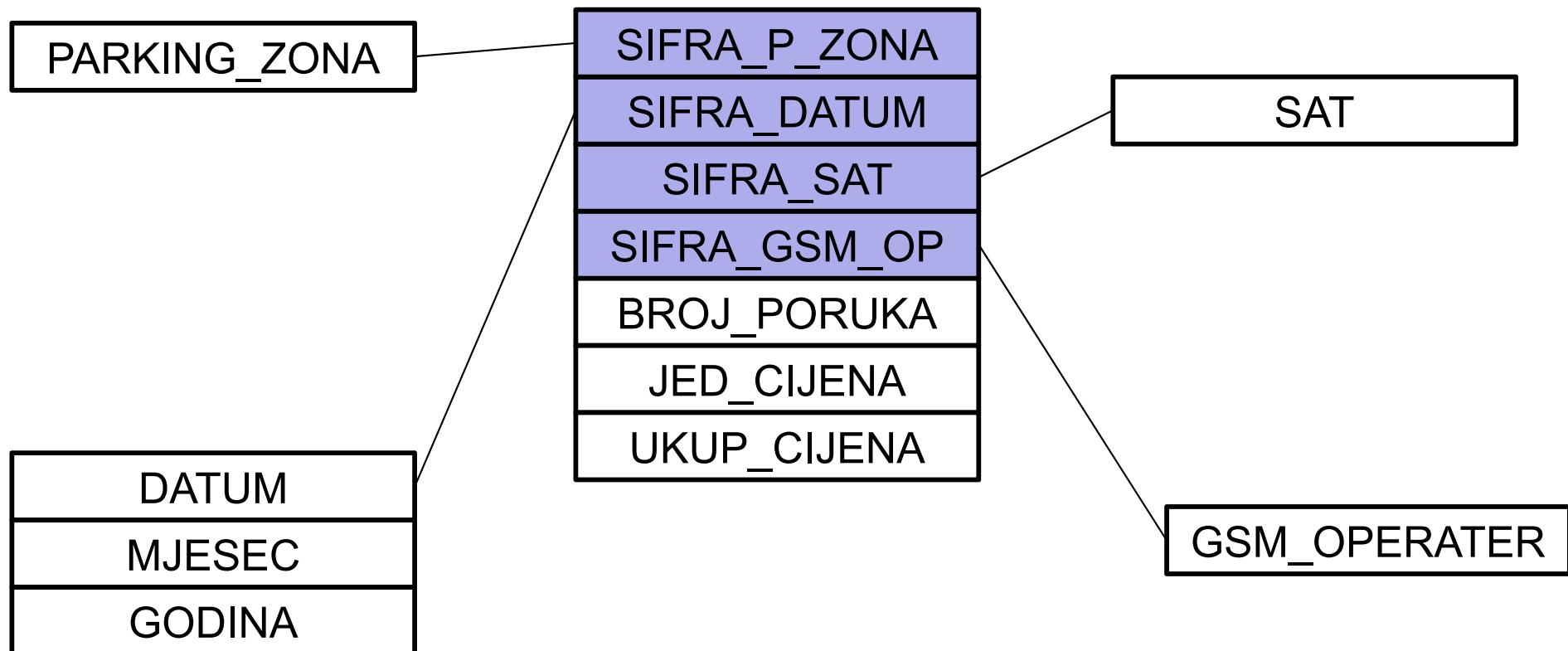
Primjer: naplata parkiranja putem SMS-a



- ♦ na kraju definirajte hijerarhijske razine

Modeli podataka za skladište podataka

Primjer: naplata parkiranja putem SMS-a



- ◆ napomena: nisu prikazani svi atributi dimenzijskih tablica, nego samo hijerarhijske razine

Modeli podataka za skladište podataka

Primjer: nabava proizvoda za trgovački lanac



- ◆ Želimo analizirati nabavu proizvoda različitih kategorija za naš trgovački lanac s obzirom na dobavljače i države iz kojih oni dolaze. Pri tome želimo omogućiti brzo dobivanje odgovora na ova i slična pitanja:
 - Koliko smo nabavili proizvoda A po mjesecima u zadnjih 6 mjeseci od dobavljača iz Austrije, a koliko od dobavljača iz Hrvatske?
 - Od kojeg dobavljača smo nabavili najveću količinu proizvoda B u proteklom tromjesečju?
 - Koliki je za 2013. i 2014. godinu bio ukupni trošak nabave (iznos u kunama) za proizvode iz kategorije „HRANA“?
 - **Oblikujte zvjezdastu shemu!**

Modeli podataka za skladište podataka

Primjer: nabava proizvoda za trgovački lanac

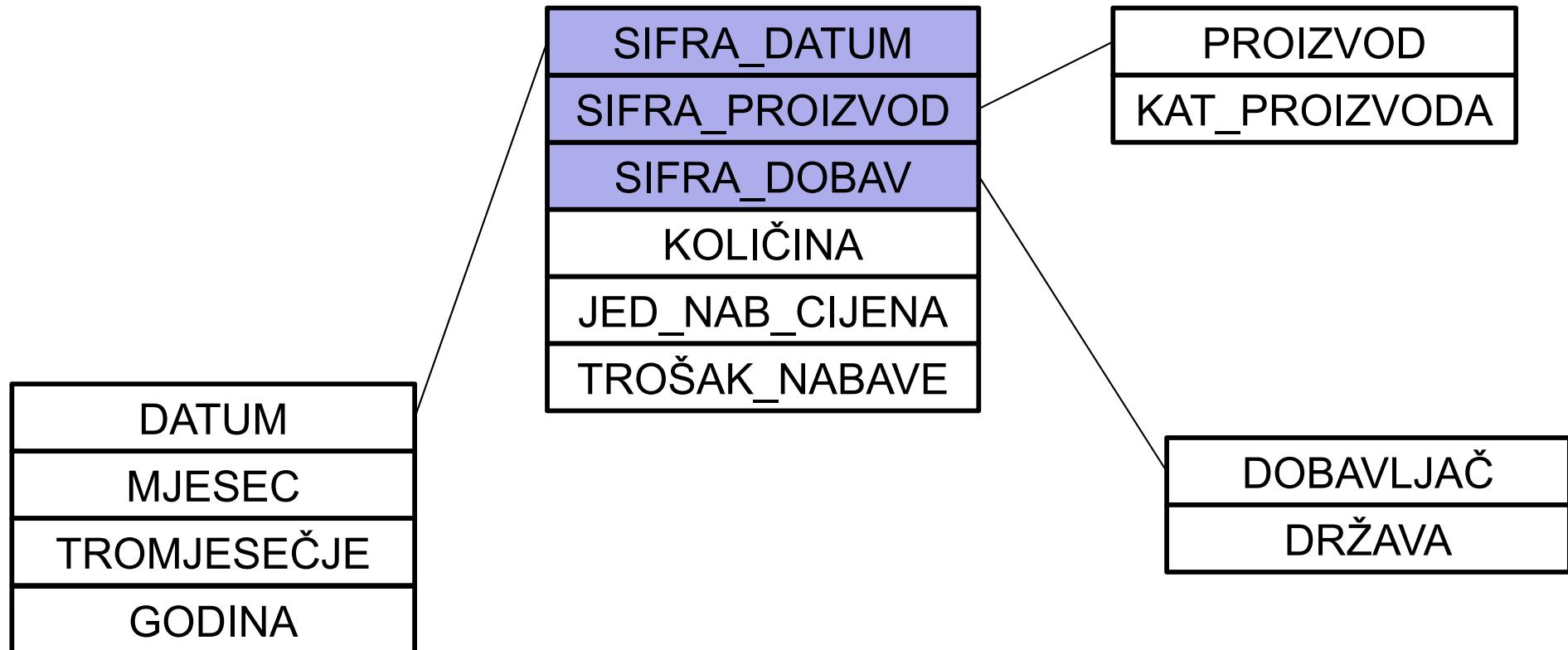
- ◆ Zajedno su prikazane **hijerarhijske razine** raznih dimenzija i **mjere**:



- ◆ Napomena: Svaka hijerarhijska razina će u pravilu sadržavati više atributa (ali te attribute u ovom primjeru ne prikazujemo).
- ◆ Sami zaključite što od navedenog su hijerarhijske razine u dimenzijama, a što mjere te koliko ukupno ima dimenzija.
- ◆ Trošak nabave računat ćemo kao umnožak količine nabavljenih proizvoda i njihove jedinične nabavne cijene.

Modeli podataka za skladište podataka

Primjer: nabava proizvoda za trgovački lanac



- napomena: nisu prikazani svi atributi dimenzijskih tablica, nego samo hijerarhijske razine

Modeli podataka za skladište podataka

Upiti

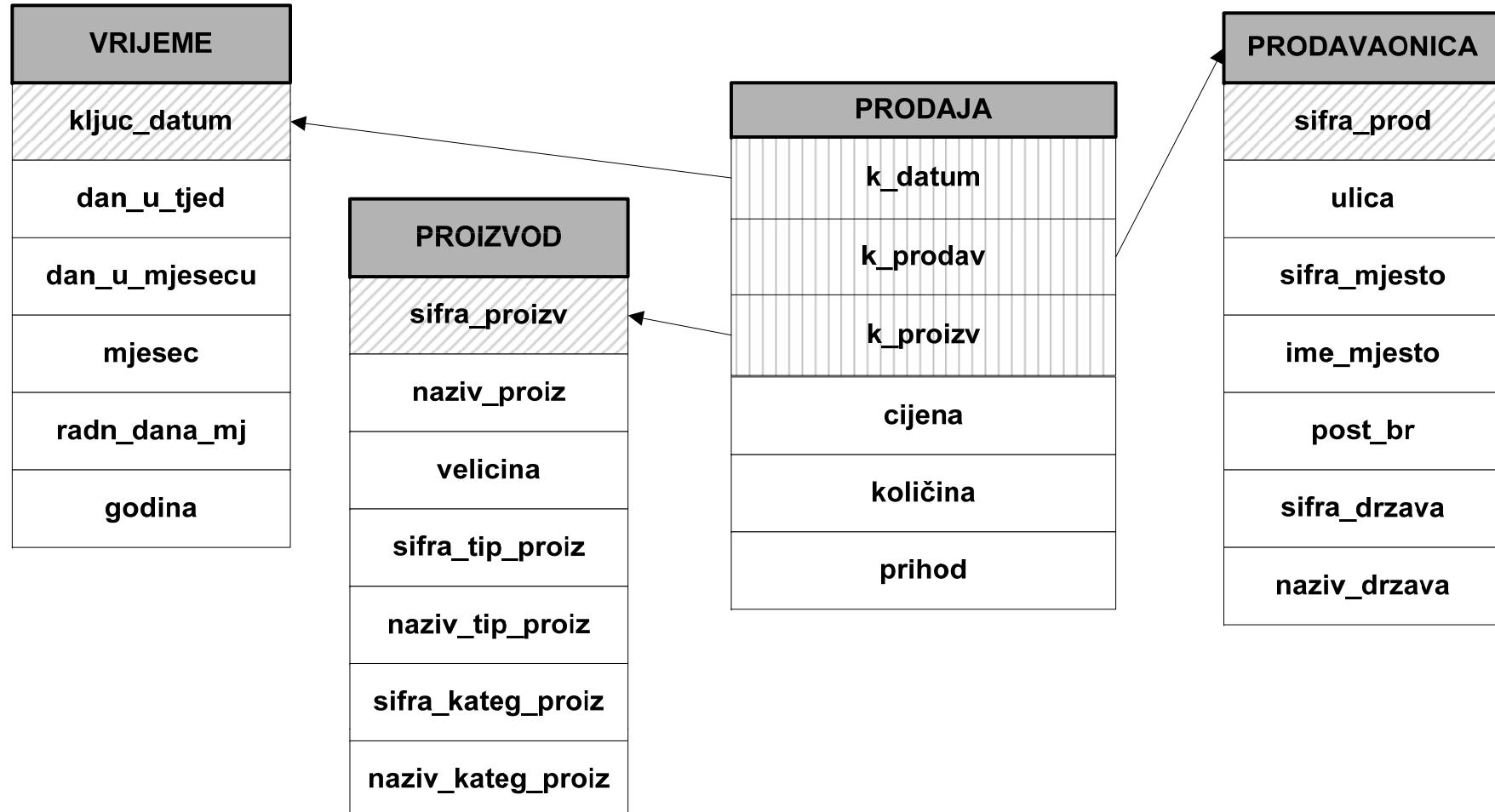


Izvedba upita:

- ◆ **U dimenzijskim tablicama** se nađu sve **vrijednosti ključa** koje zadovoljavaju postavljena ograničenja.
- ◆ Od njih se spoje sve moguće **kombinacije složenih ključeva** koje će se tražiti u činjeničnoj tablici.
- ◆ Svi nađeni podaci **u tablici činjenica** se zatim **grupiraju i sumiraju** prema specifikacijama korisnika.

Modeli podataka za skladište podataka

Upiti



- Postaviti upit koji za sve tipove proizvoda vraća ukupni prihod od prodaje u listopadu 2014. godine u Zagrebu.

Modeli podataka za skladište podataka

Upiti

SQL upit tipičan za skladište podataka:

```
SELECT DT2.sifra_tip_proiz, SUM(FT.prihod)
FROM prodaja FT, vrijeme DT1, proizvod DT2, prodavaonica DT3
WHERE FT.k_datum = DT1.kljuc_datum
      AND FT.k_proizv = DT2.sifra_proizv
      AND FT.k_prodav = DT3.sifra_prod
      AND DT1.mjesec = '10-2014'
      AND DT3.ime_mjesto= 'Zagreb'
GROUP BY DT2.sifra_tip_proiz;
```

- ♦ Upit za sve tipove proizvoda vraća ukupni prihod od prodaje u listopadu 2014. godine u Zagrebu.
- ♦ U prva tri reda dijela definirano je **spajanje** činjenične tablice s dimenzijskim tablicama preko ključeva.
- ♦ U dijelu SELECT koriste se agregacijska funkcija SUM za sumiranje prihoda za svaki tip proizvoda.

Modeli podataka za skladište podataka

Činjenične tablice bez mjera



- ◆ Postoje slučajevi kad činjenična nema niti jednu mjeru (tj. činjenicu) – tablice koje bilježe neki događaj.
- ◆ Primjer: činjenična tablica za pohađanje nastave na fakultetu:
- ◆ Dimenzije:
 - Datum
 - Student
 - Predmet
 - Profesor
 - Učionica
- ◆ Ključ činjenične tablice biti će složen od 5 atributa, po jedan za svaku dimenzijsku tablicu.
- ◆ Nema mjera (tj. činjenica) !

Modeli podataka za skladište podataka

Činjenične tablice bez mjera



Moguća pitanja:

- *Koja su predavanja najposjećenija?*
 - *Koji nastavnici imaju najviše studenata?*
 - *Koje se učionice najmanje koriste?*
 - ...
-
- ◆ Informacije o pohadanju nastave dobiju se **prebrojavanjem** odgovarajućih redaka u činjeničnoj tablici (koristi se agregatna funkcija COUNT) .

Modeli podataka za skladište podataka

Pretvorba ER dijagrama u zvjezdaste sheme



- ◆ Postupak pretvorbe normaliziranog modela podataka dobivenog ER modeliranjem u dimenzijski model (tj. skup zvjezdastih shema):

1. Razdvajanje ER dijagrama na pojedine poslovne procese

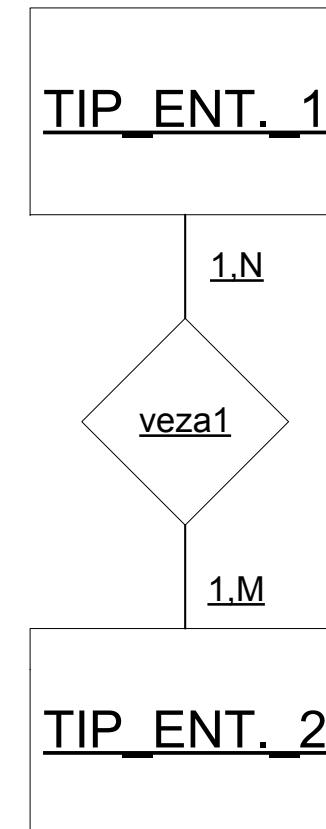
ER dijagram, koji često sadrži sve poslovne procese nekog poduzeća, prevodi se u više dimenzijskih dijagrama. Zato je prvi korak u pretvorbi ER dijagrama u skup dimenzijskih dijagrama **razdvajanje ER dijagrama na pojedine poslovne procese i modeliranje svakog od njih posebno**.

Modeli podataka za skladište podataka

Pretvorba ER dijagrama u zvjezdaste sheme

2. Odabir onih veza tipa više-prema-više koji sadrže brojčane i zbrojive atribute

- ◆ Od veza tipa više-prema-više koje sadrže brojčane i zbrojive atribute u ER dijagramu nastat će **činjenične tablice**.
- ◆ Veze tipa više-prema-više između dva, tri ili više entiteta.
- ◆ Može se raditi i o izvedenom entitetu nastalom na mjestu veze tipa više-prema-više.



Modeli podataka za skladište podataka

Pretvorba ER dijagrama u zvjezdaste sheme



- ◆ Prema pravilima pretvorbe ER modela u relacijski model podataka, veze tipa **više-prema-više** iz ER modela uvijek **rezultiraju** relacijom tj. **tablicom sa složenim ključem** u relacijskom modelu podataka.
- ◆ Takve veze mogu se u ER dijagramu prikazati kao binarne, ternarne ili veze višeg stupnja u Chenovom prikazu ili se radi o asocijativnim entitetima u Martinovom prikazu.

Modeli podataka za skladište podataka

Pretvorba ER dijagrama u zvjezdaste sheme



- ◆ Ako nema neključnih atributa u n-arnoj vezi tipa više-prema-više, odnosno odgovarajućem asocijativnom entitetu, tada će i dobivena relacija imati samo attribute koji čine složeni ključ.
- ◆ Činjenična tablica, međutim, u pravilu ima neključne attribute koji su brojčani i po mogućnosti zbrojivi po svim dimenzijama (osim ako nam je baš potrebna određena činjenična tablica bez mjera).
- ◆ Pri pretvaranju ER dijagrama u dimenzijski model **traže se veze tipa više-prema-više koje sadrže brojčane i zbrojive neključne attribute**, te se od takvih veza dobijaju činjenične tablice.

Modeli podataka za skladište podataka

Pretvorba ER dijagrama u zvjezdaste sheme



3. Denormalizacija svih ostalih tablica

- ♦ **Denormaliziraju se sve ostale tablice u dimenzijske tablice** s jednostavnim ključevima koji izravno povezuju te tablice s činjeničnim tablicama.

Na taj način, pretvorbom n-arne veze tipa više-prema-više iz konceptualnog ER modela u implementacijski relacijski model dobija se **shema zvijezda**.

Od n-arne veze nastaje zvjezdasta shema s jednom činjeničnom tablicom i n dimenzijskih tablica.

Modeli podataka za skladište podataka

Različiti pristupi oblikovanju skladišta podataka



- ◆ Dva osnovna pristupa:

1. Inmon:

U skladištu podataka prevladava normalizirani model podataka. Za pojedine teme, tj. grupe pitanja koje će krajnji korisnici često postavljati, kreiraju se manji dimenzijski modeli.

2. Kimball:

Čitav model skladišta podataka se radi u dimenzijskom modelu. Pojedini dimenzijski modeli (zvijezde) povezani su zajedno inzistiranjem na uporabi zajedničkih tj. usklađenih dimenzijskih tablica.

Modeli podataka za skladište podataka

Različiti pristupi oblikovanju skladišta podataka



Pristup 1 – Inmon:

- ◆ U skladištu podataka prevladava normalizirani model podataka (obično u 3. normalnoj formi) koji se u većini slučajeva dobija postupkom ER oblikovanja.
- ◆ Kreira se i više manjih dimenzijskih modela za pojedine teme, tj. grupe pitanja koje će krajnji korisnici često postavljati, obično preko OLAP alata. Ti će dimenzijski modeli sadržavati sumarne podatke.
- ◆ Da se ne bi mijenjao normalizirani model, nekad se koriste pogledi (tj. samo logičke, a ne i fizičke definicije relacija), kojima se može pri postavljanju upita “sakriti” stvarna složenost modela.

Modeli podataka za skladište podataka

Različiti pristupi oblikovanju skladišta podataka



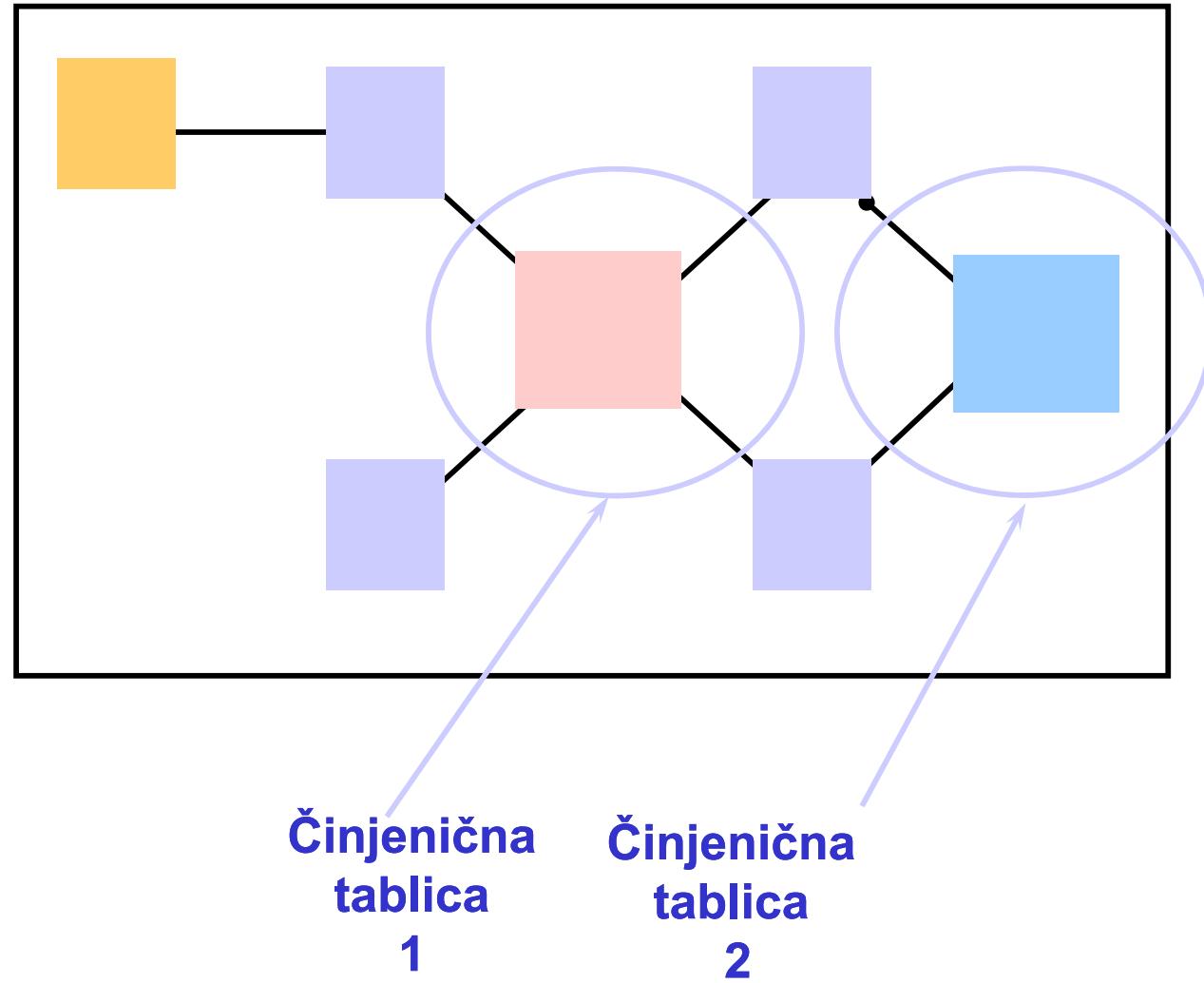
Pristup 2 – Kimball:

- ◆ Skup zvjezdastih shema, od kojih svaka opisuje jedan poslovni proces, povezuje se u model podataka čitavog skladišta podataka.
- ◆ Pojedine zvjezdaste sheme povezane su zajedno inzistiranjem na uporabi **zajedničkih**, odnosno **usklađenih dimenzijskih tablica**. Ako se dvije dimenzijske tablice koje bi trebale imati isto značenje nalaze u dvije zvijezde, to moraju biti **potpuno iste dimenzijske tablice ili jedna mora biti podskup druge tablice** (tj. projekcija). Usklađena dimenzijska tablica ima isto značenje bez obzira na koju je činjeničnu tablicu spojena.

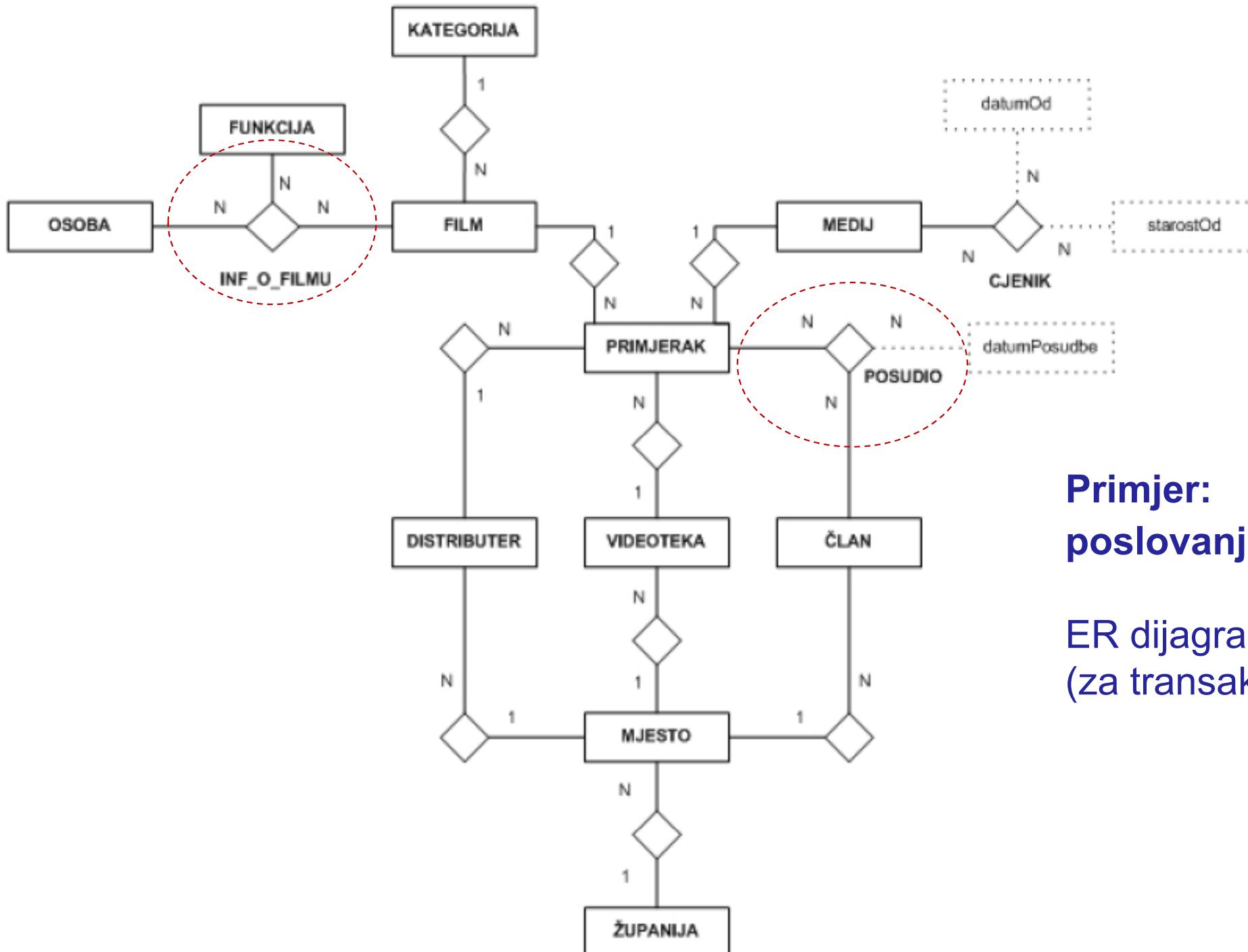
Modeli podataka za skladište podataka

Različiti pristupi oblikovanju skladišta podataka

- ◆ Kimball
- ◆ zajedničke (usklađene) dimenzijske tablice



Modeli podataka za skladište podataka



**Primjer:
poslovanje videoteke**

**ER dijagram
(za transakcijsku bazu)**

Modeli podataka za skladište podataka

Primjer: poslovanje videoteke



Pitanja vezana uz poslovanje videoteka (**posuđivanje** filmova):

- ◆ Koji se filmovi najviše posuđuju?
- ◆ Koje kategorije filmova su najpopularnije?
- ◆ Koji članovi ne vraćaju filme na vrijeme ili ne plaćaju zakasninu?
- ◆ Koji medij za pohranjivanje filmova je najrašireniji (po broju posuđenih filmova na tom mediju)?
- ◆ Koje videoteke najslabije posluju (najmanje zarađuju)?

Napomena: u primjeru se uzima da svaki film pripada samo jednoj kategoriji

Koje su dimenzije za činjeničnu tablicu POSUDBA?

Modeli podataka za skladište podataka

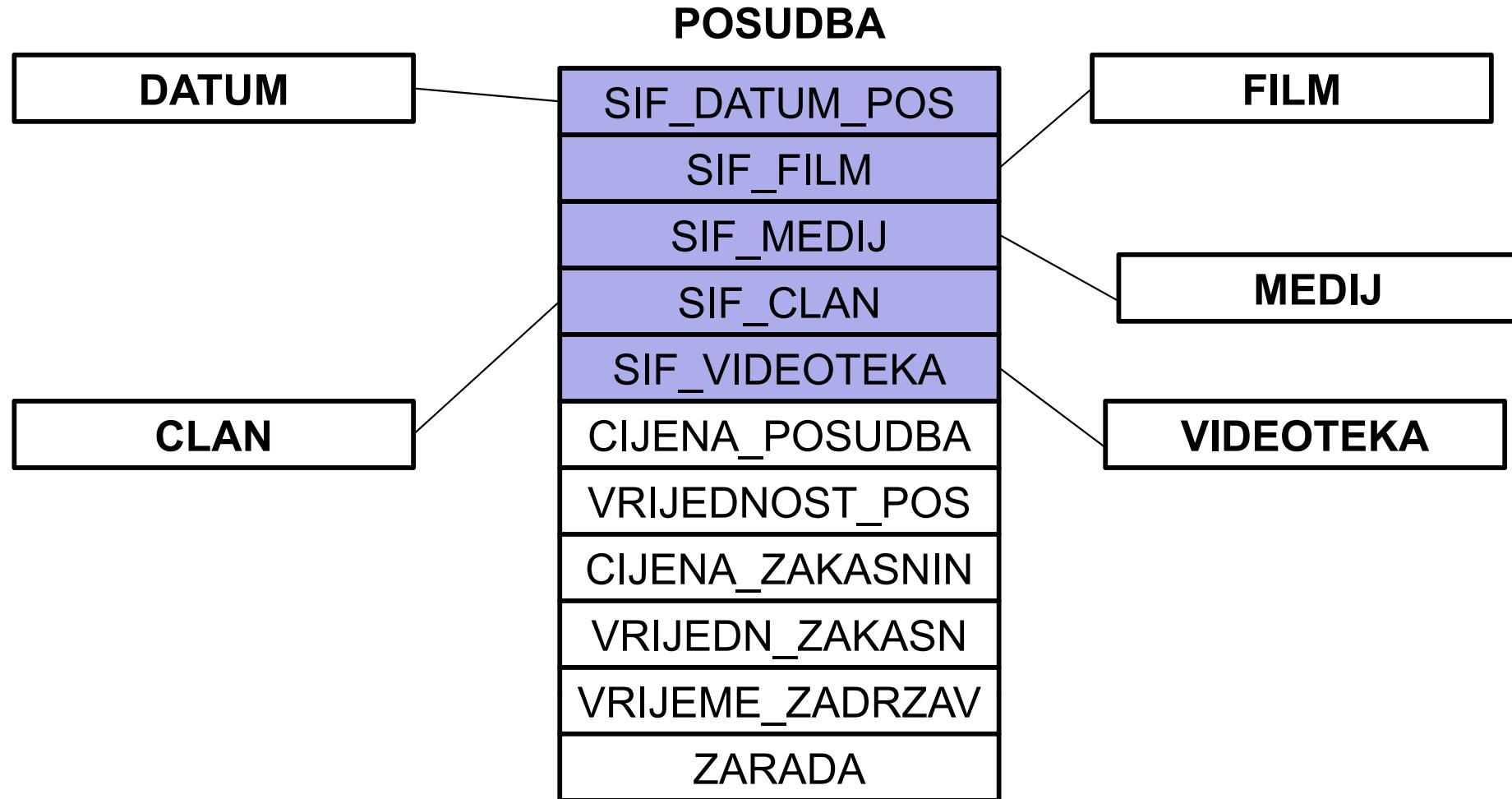
Primjer: poslovanje videoteke



- ◆ Potrebno je znati:
 - koje **filmove** je određeni **član** određene **videoteke** posudio,
 - na kojem **mediju** su filmovi pohranjeni (VHS, DVD ili BD – *Blue-ray Disc*),
 - kada su posuđeni i kada su vraćeni (**datumi posudbi i vraćanja**)
- ◆ Mogu se izračunati **mjere**:
 - jedinična cijena posudbe,
 - stvarna vrijednost posudbe,
 - jedinična cijena zakasnine,
 - vrijeme zadržavanja filma,
 - vrijednost zakasnine,
 - zarada (vrijednost posudbe + vrijednost zakasnine)

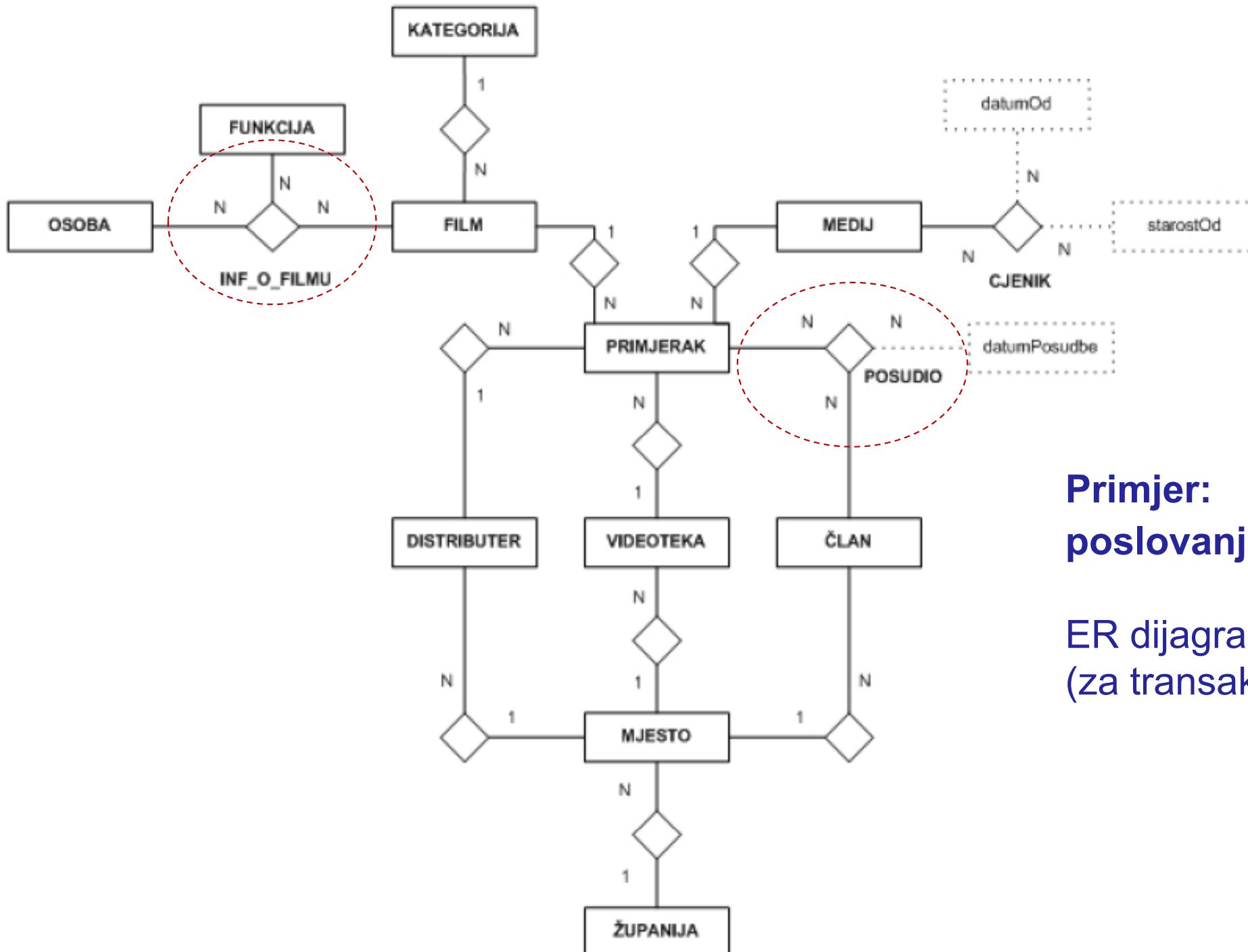
Modeli podataka za skladište podataka

Primjer: poslovanje videoteke



- ◆ hijerarhijske razine? (pratimo veze –prema-jedan u ER)

Modeli podataka za skladište podataka

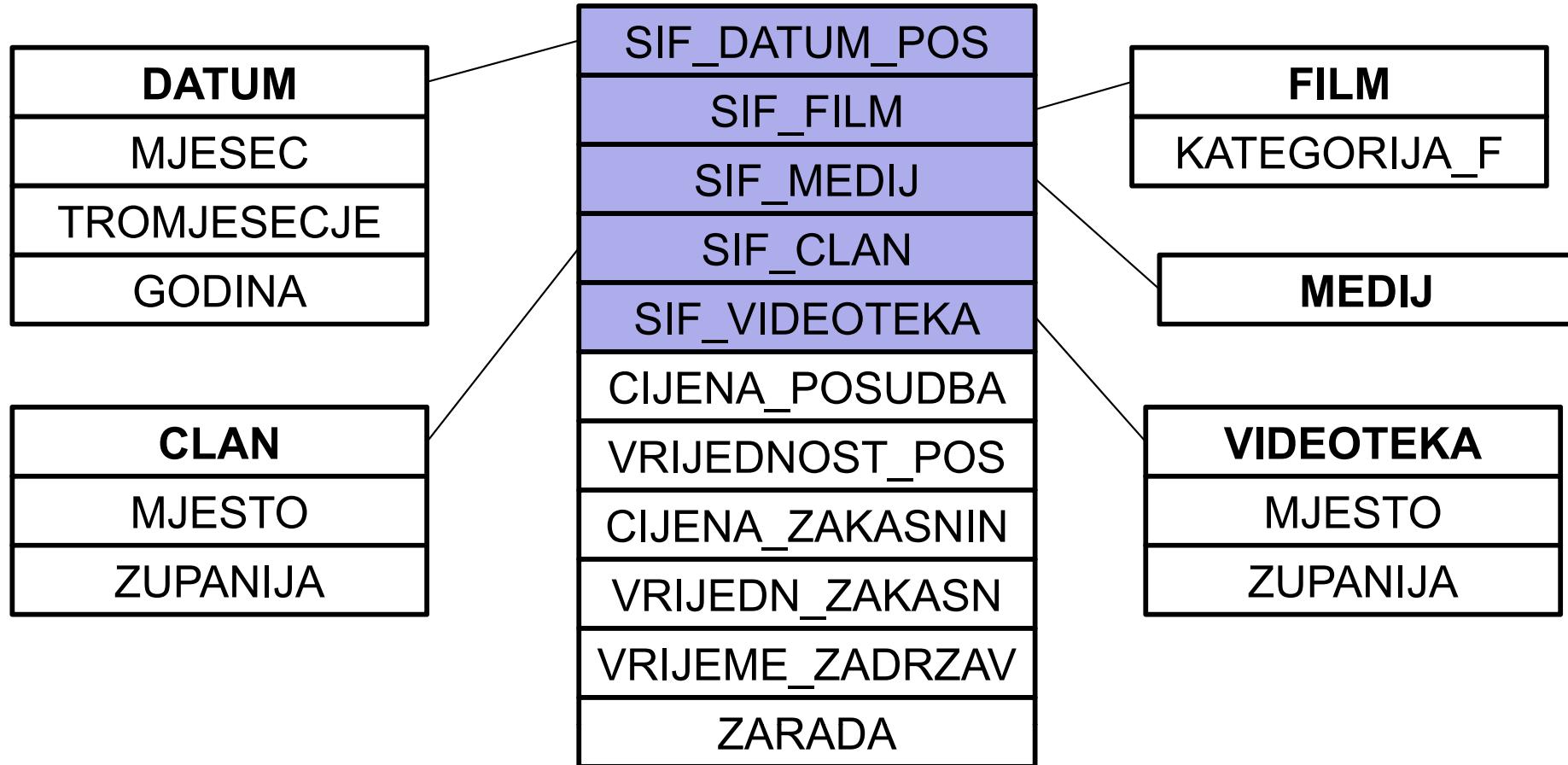


**Primjer:
poslovanje videoteke**

**ER dijagram
(za transakcijsku bazu)**

Modeli podataka za skladište podataka

Primjer: poslovanje videoteke



- Koliko iznose godišnje zarade po gradovima u kojima se nalaze videoteke?

Modeli podataka za skladište podataka

Primjer: poslovanje videoteke

- Koliko iznose godišnje zarade po gradovima u kojima se nalaze videoteke?
- potrebno je spojiti činjeničnu tablicu POSUDBA i tablice DIM_VREMENSKA (zbog atributa GODINA) i DIM_VIDEOTEKA (zbog atributa NAZIV_MJESTA) te za svaku godinu i pojedino mjesto zbrojiti vrijednosti obavljenih posudbi i zakasnina

| dDatum | Measures | | | | |
|--------|---------------|--------|-------|-----------|--------|
| | UKUPNA_ZARADA | | | | |
| | dVideoteka | | | | |
| dDatum | Rijeka | Osljek | Split | Dubrovnik | Zagreb |
| 2003 | 426 | 718 | 465 | 544 | 1.520 |
| 2004 | 1.086 | 2.088 | 1.340 | 81 | 3.561 |
| 2005 | 1.575 | 2.557 | 1.766 | 642 | 3.961 |
| 2006 | 1.609 | 2.990 | 2.189 | 1.710 | 5.258 |
| 2007 | 4.254 | 6.348 | 5.209 | 1.841 | 12.988 |
| 2008 | | | | 522 | 105 |

Modeli podataka za skladište podataka

Primjer: poslovanje videoteka



```
SELECT dim_vremenska.GODINA, dim_videoteka.NAZIV_MJESTA,  
SUM(ZARADA)  
  
FROM posudba, dim_vremenska, dim_videoteka  
  
WHERE posudba.DATUM = _____ .DATUM  
and posudba.SIF_VIDEOTEKA = _____ .SIF_VIDEOTEKA  
  
GROUP BY dim_vremenska._____, dim_videoteka._____  
  
ORDER BY dim_vremenska.GODINA;
```

Modeli podataka za skladište podataka

Primjer: poslovanje videoteka



```
SELECT dim_vremenska.GODINA, dim_videoteka.NAZIV_MJESTA,  
SUM(ZARADA)  
  
FROM posudba, dim_vremenska, dim_videoteka  
  
WHERE posudba.DATUM = dim_vremenska.DATUM  
and posudba.SIF_VIDEOTEKA = dim_videoteka.SIF_VIDEOTEKA  
  
GROUP BY dim_vremenska.GODINA, dim_videoteka.NAZIV_MJESTA  
  
ORDER BY dim_vremenska.GODINA;
```

Modeli podataka za skladište podataka

Primjer: poslovanje videoteke



Pitanja vezana uz poslovanje videoteka (**zaprimanje** filmova):

- ◆ Koji distributeri nude filme po najnižim cijenama?
- ◆ Kolike su ukupne zaprimljene vrijednosti za određene videoteke?
- ◆ Koji medij za pohranjivanje filmova je najrašireniji (po broju zaprimljenih filmova)?

Modeli podataka za skladište podataka

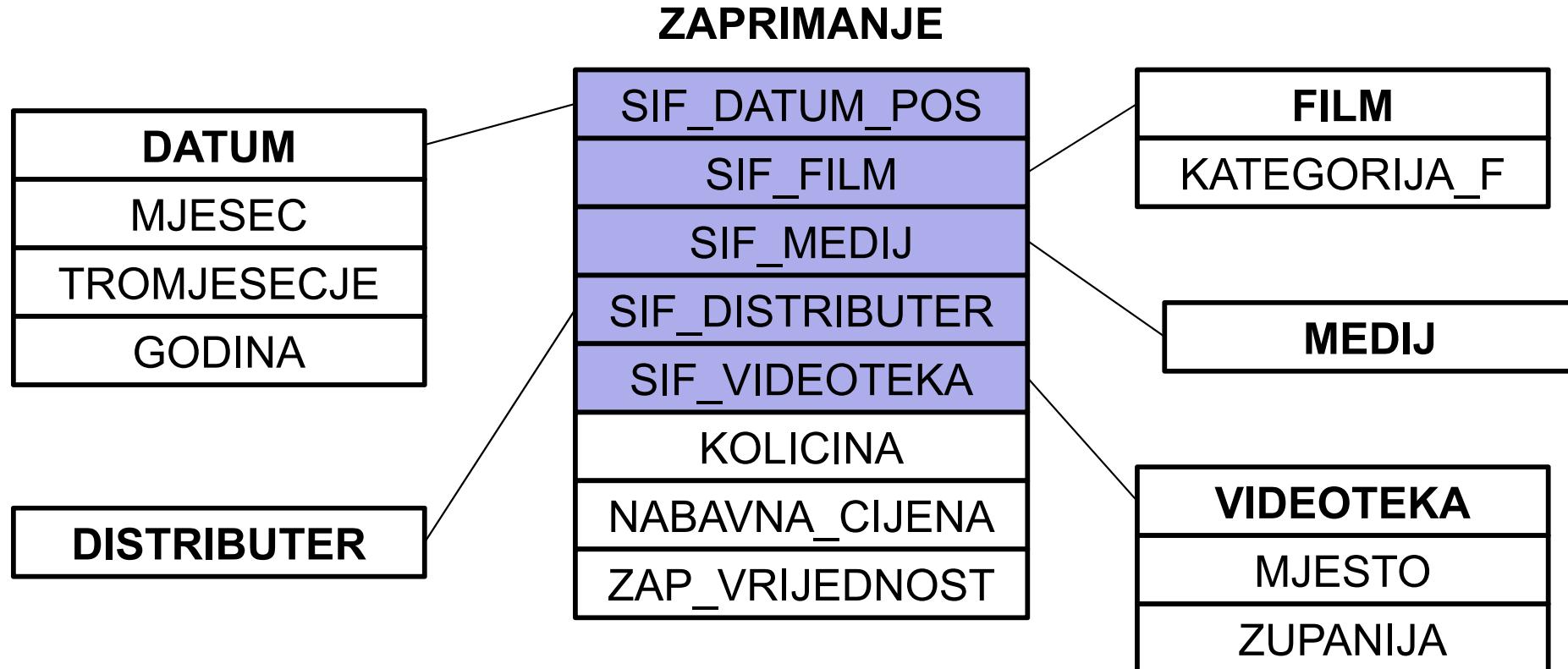
Primjer: poslovanje videoteke



- ◆ Potrebno je znati:
 - koji su **filmovi** na kojem **mediju** zaprimljeni u koje **videoteke** od kojeg **distributera** kojeg **datuma**.
 - Koje su dimenzije za činjeničnu tablicu **ZAPRIMANJE**?
- ◆ Mogu se izračunati vrijednosti za mjere:
 - količina zaprimljenih primjeraka,
 - jedinična nabavna cijena primjerka
 - ukupna zaprimljena vrijednost (umnožak količine i nabavne cijene)

Modeli podataka za skladište podataka

Primjer: poslovanje videoteke

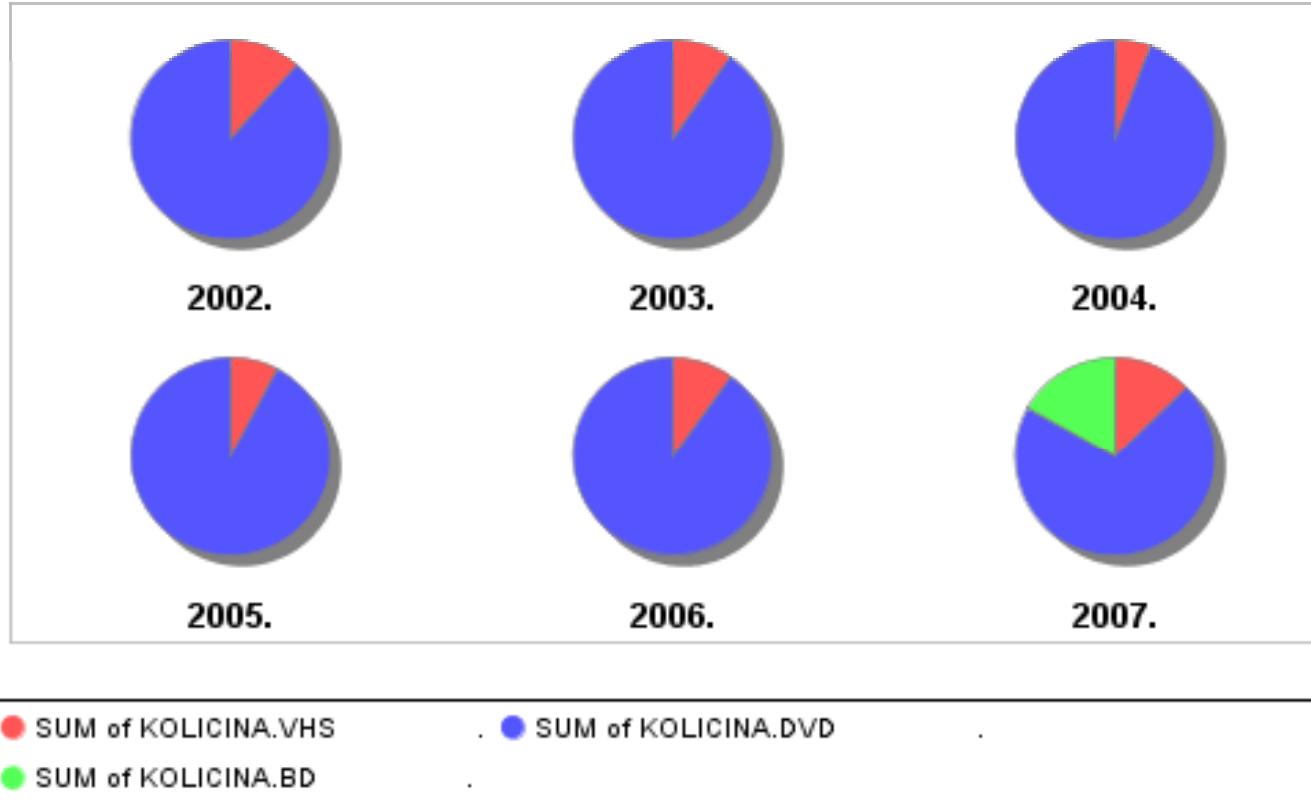


- ◆ Broj zaprimljenih primjeraka na različitim medijima (VHS, DVD, BD) po godinama?

Modeli podataka za skladište podataka

Primjer: poslovanje videoteke

- ◆ Broj zaprimljenih primjeraka na različitim medijima po godinama?



| Measures | SUM of KOLICINA | | | |
|----------|-----------------|-------|-----|----|
| | dMedij | | | |
| | dDatum | VHS | DVD | BD |
| +2002 | 264 | 1.984 | | |
| +2003 | 18 | 168 | | |
| +2004 | 18 | 296 | | |
| +2005 | 24 | 280 | | |
| +2006 | 36 | 324 | | |
| +2007 | 60 | 332 | 80 | |

Modeli podataka za skladište podataka

Primjer: poslovanje videoteka



```
SELECT dim_vremenska.GODINA, dim_mediј.NAZIV_MEDIJ,  
sum(KOLICINA)  
  
FROM zaprimanje, dim_vremenska, dim_mediј  
  
WHERE zaprimanje.DATUM = _____ .DATUM  
and zaprimanje.SIF_MEDIJ = _____ .SIF_MEDIJ  
  
GROUP BY dim_vremenska._____, dim_mediј.NAZIV_MEDIJ  
  
ORDER BY dim_vremenska.GODINA;
```

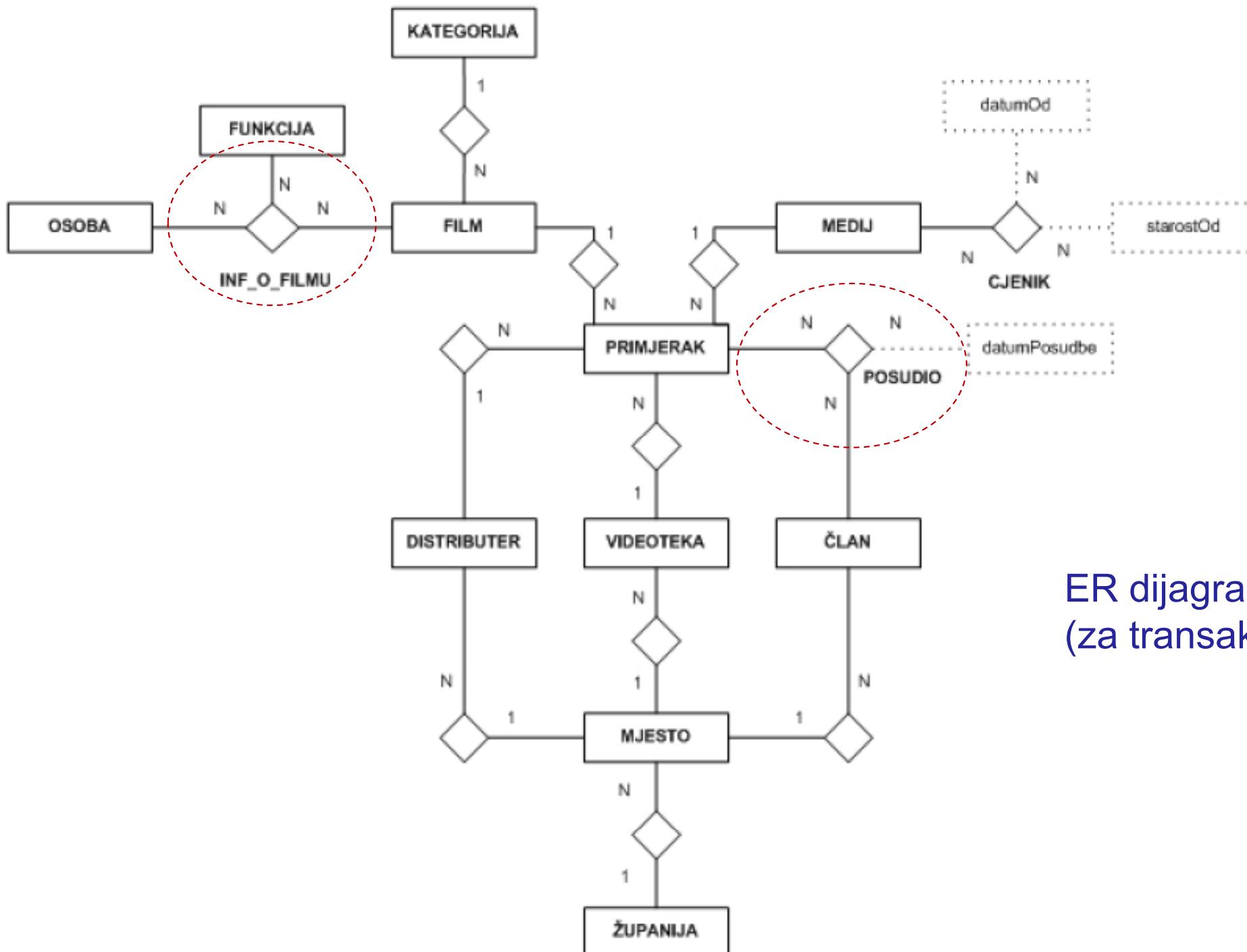
Modeli podataka za skladište podataka

Primjer: poslovanje videoteka



```
SELECT dim_vremenska.GODINA, dim_mediј.NAZIV_MEDIJ,  
sum(KOLICINA)  
  
FROM zaprimanje, dim_vremenska, dim_mediј  
  
WHERE zaprimanje.DATUM = dim_vremenska.DATUM  
and zaprimanje.SIF_MEDIJ = dim_mediј.SIF_MEDIJ  
  
GROUP BY dim_vremenska.GODINA, dim_mediј.NAZIV_MEDIJ  
  
ORDER BY dim_vremenska.GODINA;
```

Modeli podataka za skladište podataka

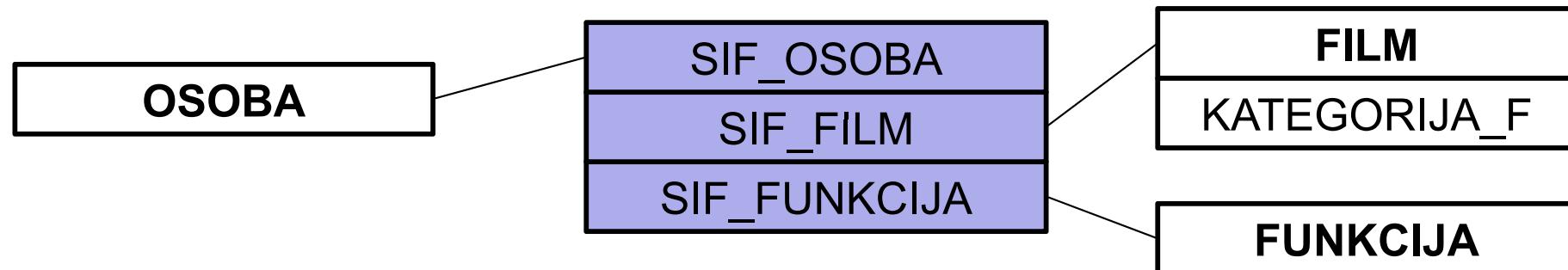


ER dijagram
(za transakcijsku bazu)

Modeli podataka za skladište podataka

Primjer: poslovanje videoteke

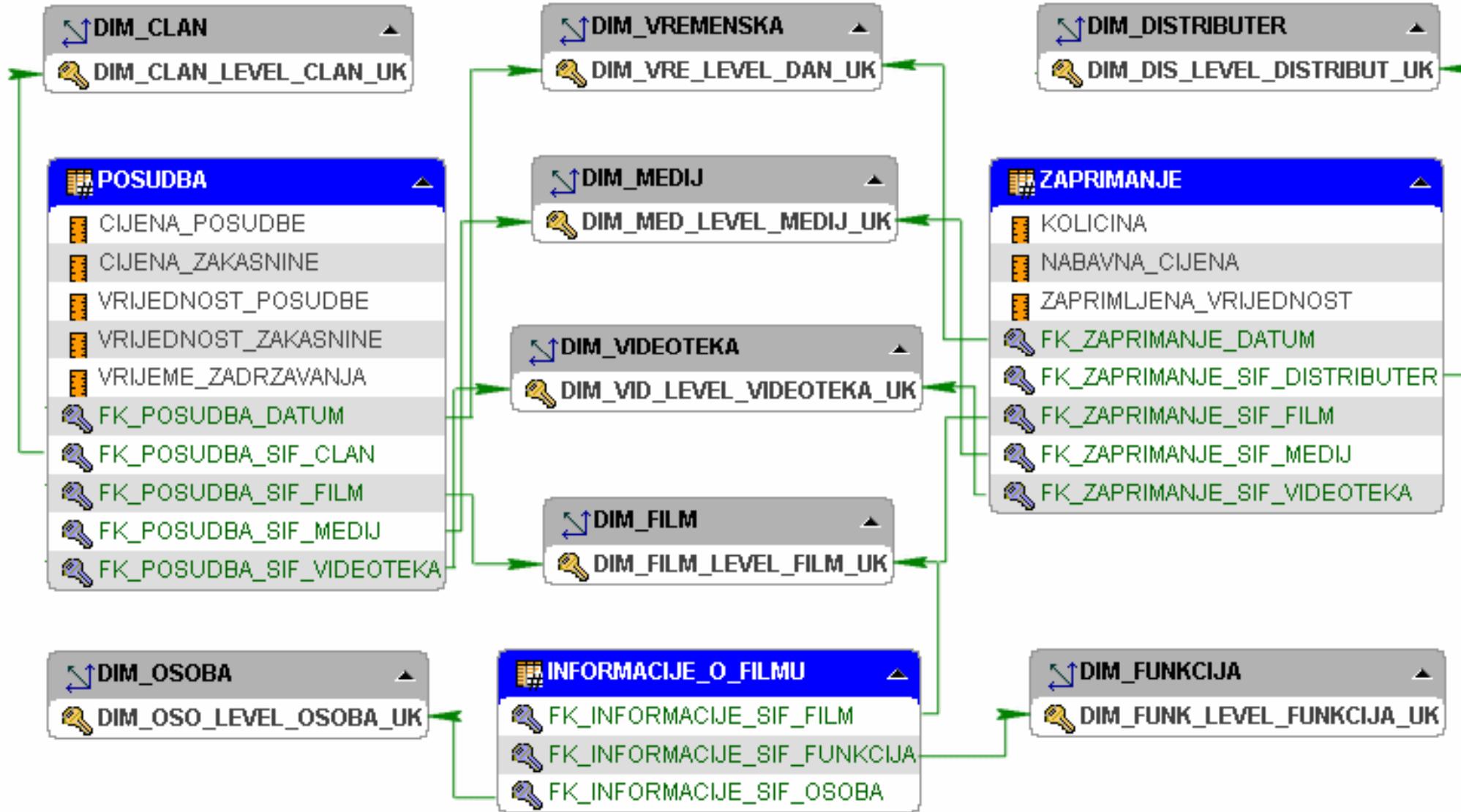
- ◆ Zvijezdom INFORMACIJE_O_FILMU ne prati se nikakav proces u poslovanju videoteke, već se samo pruža informacija o filmovima koje lanac videoteka sadrži (činjenična tablica bez mjera):



- ◆ Činjenična tablica INFORMACIJE_O_FILMU sadrži podatke o funkcijama (glumac, redatelj ili scenarist) pojedine (poznate) osobe u nekom filmu.
- ◆ Moguća pitanja:
 - Koji filmovi određenog redatelja se nalaze u videoteci?
 - Koje kategorije (žanrove) filmova određeni glumci najčešće snimaju?

Modeli podataka za skladište podataka

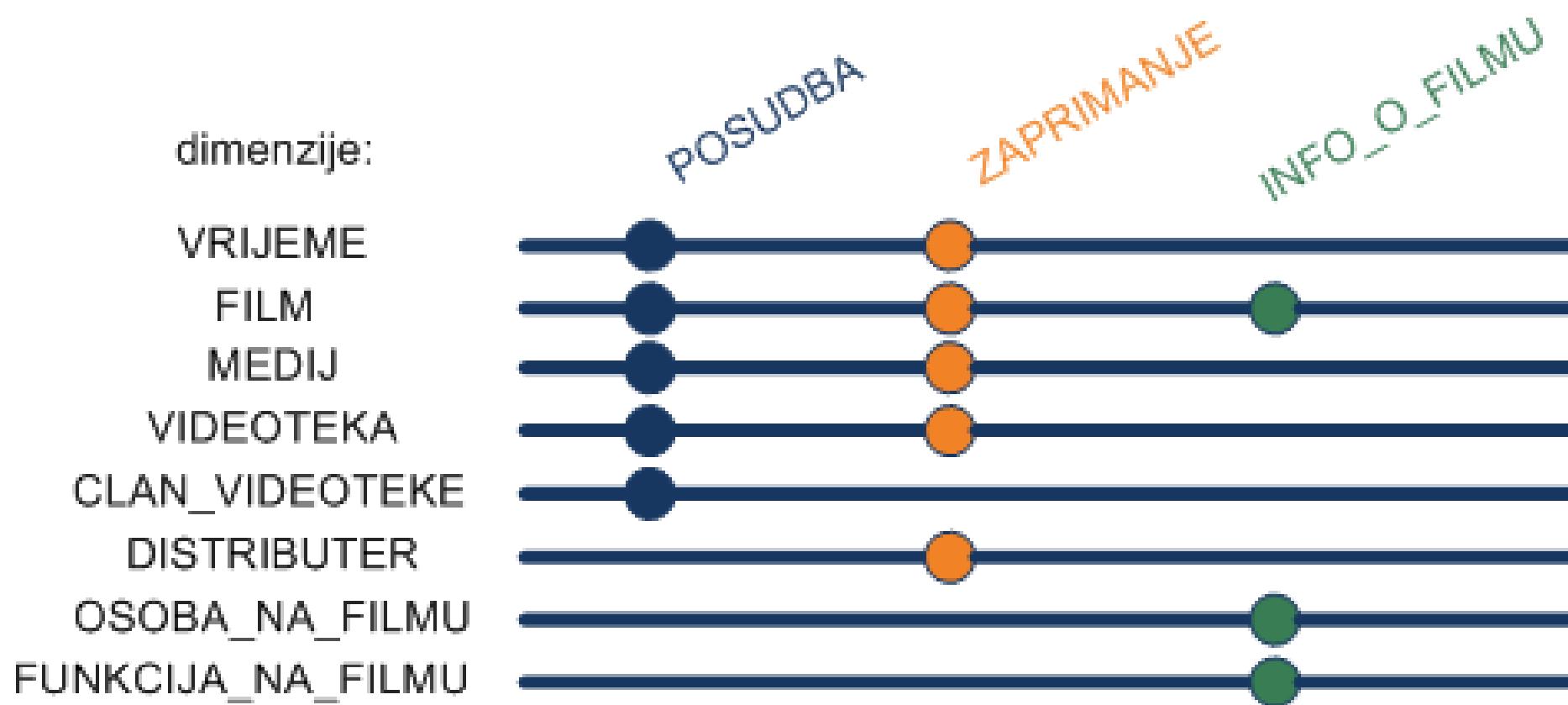
Primjer: poslovanje videoteka



Modeli podataka za skladište podataka

Primjer: poslovanje videoteke

- ◆ ***data warehouse bus architecture*** (Kimball)
- ◆ zajedničke (usklađene) dimenzije



UPRAVLJANJE PODACIMA

SKLADIŠTENJE PODATAKA
(3.dio)

Dimenzijski model podataka

- Sporopromjenjive dimenzije
- Ista dimenzija u različitim ulogama
- Pohranjivanje sumarnih (agregiranih) podataka

Izgradnja skladišta podataka

Modeli podataka za skladište podataka

Sporopromjenjive dimenzije



Vrijednosti u dimenzijskim tablicama su prilično statične, no i one će se u određenim slučajevima mijenjati tijekom vremena.

Primjeri:

- ◆ Kupac X je promijenio adresu.
- ◆ Promijenio se opis proizvoda A.
- ◆ Promijenila se zemljopisna podjela na regije.

Skladište podataka treba ispravno pratiti promjene, a ujedno znati kakvo je stanje bilo u određenom trenutku.

Dva su osnovna načina obrade promjena u dimenzijskim tablicama.

Modeli podataka za skladište podataka

Sporopromjenjive dimenzije



1. način

- ◆ **Stara vrijednost u dimenzijskoj tablici zamijeni se novom.**
- ◆ **Gubi se povijest promjena** dimenzijskih vrijednosti.
- ◆ Ovaj način se koristi za **ispravljanje pogrešaka** pri punjenju dimenzijske tablice podacima.
- ◆ Može se koristiti i **za dimenzije čija nas povijest ne zanima**.

Modeli podataka za skladište podataka

Sporopromjenjive dimenzije



2. način

- ◆ Dodaje se novi redak (s novom vrijednošću primarnog ključa) u dimenzijsku tablicu.
- ◆ I stari i novi retci uz primarni ključ sadrže i **stalni identifikator** ("prirodni ključ"), za npr. identifikator određenog kupca koji je promijenio adresu ili identifikator određenog proizvoda kojem je promijenjen opis.
- ◆ Dimenzijska tablica može sadržavati sljedeće stupce: **zastavicu za aktualni redak, vrijedi_od, vrijedi_do**.

Modeli podataka za skladište podataka

Sporopromjenjive dimenzije

| Sifra (PK) | Mat_br | Naziv | Ulica | Kbr | Mjesto | Post_br | Vrijedi_od | Vrijedi_do |
|---------------|--------|-------|--------------|-----|--------|---------|--------------------------|--------------------------|
| 12311 | 123 | AB | Teslina | 3 | Zagreb | 10000 | 14.7. 2008. | 23. 10. 2014. |
| 12312 | 123 | AB | Ilica | 8 | Zagreb | 10000 | 24. 10. 2014. | |

Pri promjeni vrijednosti dimenzijskog atributa pronađemo u dimenzijskoj tablici trenutno važeći redak. Lociramo ga u ovom primjeru pomoću atributa *Mat_br* i zatim tražimo redak čiji atribut *Vrijedi_do* ima vrijednost NULL .

Ažuriramo vrijednost *Vrijedi_do* i dodamo novi redak s novim primarnim ključem i novom vrijednošću atributa čiju promjenu pratimo.

Modeli podataka za skladište podataka

Sporopromjenjive dimenzije

| Sifra (PK) | Mat_br | Naziv | Ulica | Kbr | Mjesto | Post_br | Vrijedi_od | Aktualno |
|---------------|--------|-------|--------------|----------|--------|---------|------------------|----------|
| 12311 | 123 | AB | Teslina | 3 | Zagreb | 10000 | 14.7. 2008. | N |
| 12312 | 123 | AB | Ilica | 8 | Zagreb | 10000 | 24. 10. 2014. | D |

Umjesto stupca *Vrijedi_do* može se koristiti zastavica *Aktualno* s vrijednostima za DA i NE.

Može se pamtit i sva tri zapisa: *Vrijedi_od*, *Vrijedi_do* i *Aktualno*. Na taj način ubrzava se dohvaćanje podataka o vremenskom trajanju pojedine vrijednosti, odnosno nije potrebno dodatno pretraživanje po dimenziji da bismo pronašli prethodno važeći zapis u svrhu otkrivanja datuma do kojeg je određena vrijednost atributa važila.

Kreiranjem indeksa nad stupcem *Aktualno*, ubrzava se lociranje trenutno važećeg zapisa.

Modeli podataka za skladište podataka

Sporopromjenjive dimenzije

| Sifra (PK) | Mat_br | Naziv | Ulica | Kbr | Mjesto | Post_br | Vrijedi od | Aktualno |
|---------------|--------|-------|--------------|----------|--------|---------|------------------|----------|
| 12311 | 123 | AB | Teslina | 3 | Zagreb | 10000 | 14.7. 2008. | N |
| 12312 | 123 | AB | Ilica | 8 | Zagreb | 10000 | 24. 10. 2014. | D |

U činjeničnoj tablici mogu se naći retci s vrijednošću odgovarajućeg atributa jednakoj 12311 za svaku kupnju prije 24.10.2014. i retci s vrijednošću 12312 za kupnju od 24.10.2014. do danas.

Ako u upitu tražimo da je naziv kupca jednak 'AB', obje vrijednosti šifre tog kupca (12311 i 12312) će se tražiti u činjeničnoj tablici te će se dobiti odgovarajući skup rezultata.

Modeli podataka za skladište podataka

Ista dimenzija u različitim ulogama



- ◆ Primjer: analiza slanja SMS poruka
- ◆ Telekomunikacijski operater A želi analizirati koliko njegovi korisnici **šalju SMS poruka korisnicima različitih operatera** u različitim vremenskim periodima
- ◆ također želi znati **u mreži kojeg operatera se pošiljatelj nalazio** u trenutku slanja SMS poruke (korisnik je mogao npr. slati SMS poruku iz inozemstva), odnosno **u mreži kojeg operatera je primatelj primio poruku**
- ◆ Dimenzija TK_OPERATOR pojavljuje se u tri uloge:
 - operater iz čije mreže je poslan sms
 - operater primatelja
 - operater u čijoj mreži je primljen sms

TK_OPERATOR

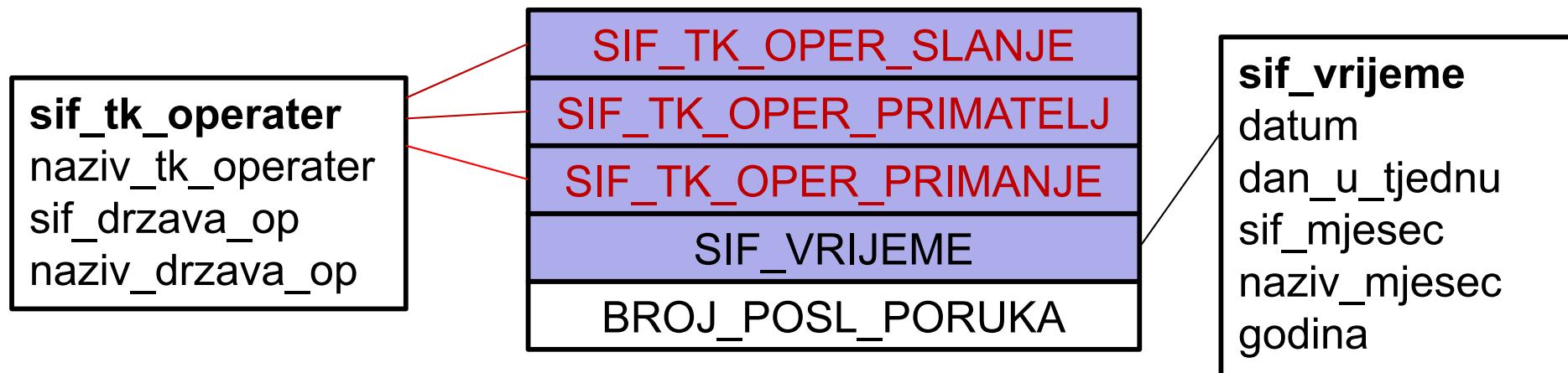
| |
|--------------------------|
| sif_tk_operater |
| naziv_tk_operater |
| sif_drzava_op |
| naziv_drzava_op |

Prepostavka: svaki operater djeluje u samo jednoj državi

Modeli podataka za skladište podataka

Ista dimenzija u različitim ulogama

- ◆ Činjenična tablica u kojoj jedna dimenzija (TK_OPERATER) ima 3 uloge:



- ◆ Za svaku od uloga dimenzije kreira se zasebna **virtualna relacija** (eng. *view*) npr.

```
CREATE VIEW tk_oper_primatelj AS  
SELECT  
    sif_tk_operater AS sif_tk_oper_primatelj,  
    naziv_tk_operater AS naziv_tk_oper_primatelj,  
    . . .  
FROM tk_operater;
```

Modeli podataka za skladište podataka

Ista dimenzija u različitim ulogama



- ◆ Za dimenziju koja se pojavljuje u više uloga napravi se jedna dimenzijska tablica te se za svaku ulogu napravi virtualna relacija (view) preko koje se pristupa podacima iz dimenzijske tablice.
- ◆ Ista dimenzijska tablica je tako više puta spojena na činjeničnu tablicu – za svaku ulogu stavi se u činjeničnu tablicu jedan atribut koji je strani ključ na dimenzijsku tablicu i dio je složenog ključa činjenične tablice.
- ◆ Preporučuje se u svakoj virtualnoj relaciji **preimenovati nazive atributa**.
- ◆ Dimenzija koja se najčešće koristi u više uloga je vremenska dimenzija.

Modeli podataka za skladište podataka

Pohranjivanje sumarnih (agregiranih) podataka



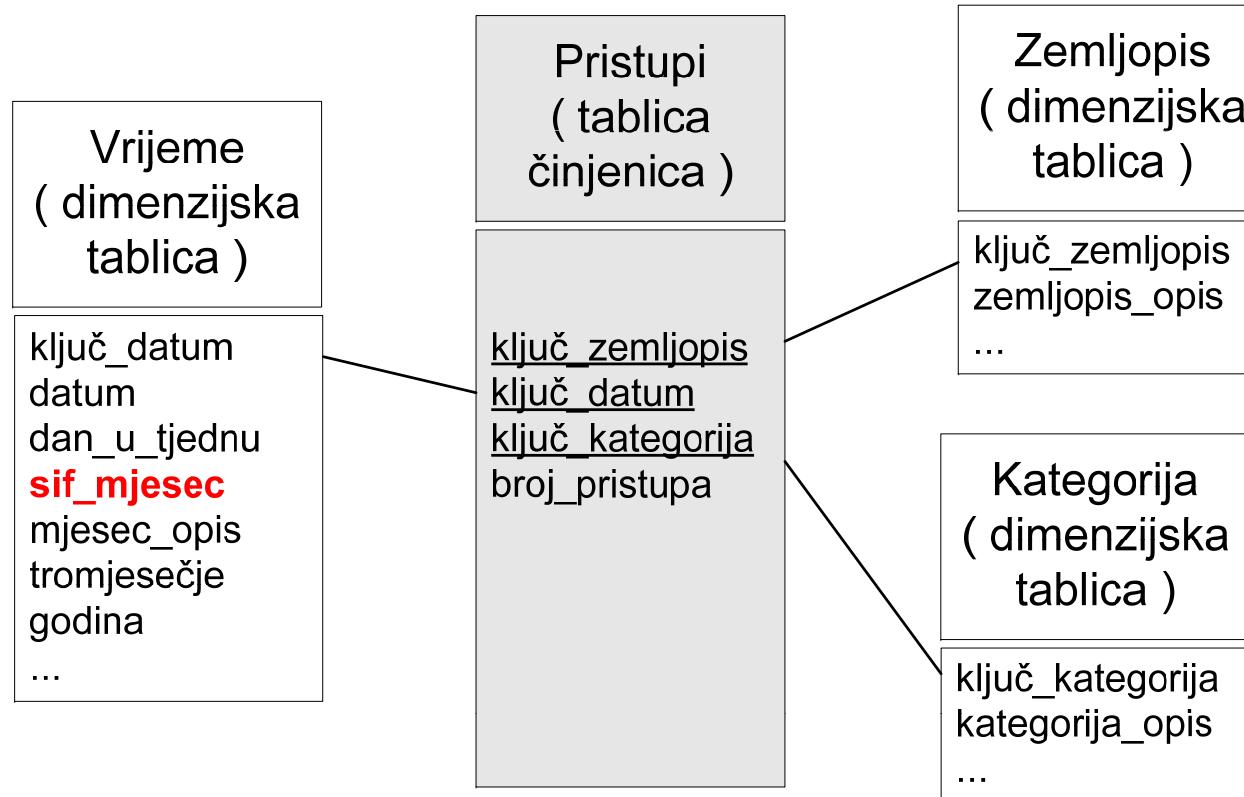
- ◆ Krajnji korisnik u svojim upitima češće traži **sumarne podatke** nego podatke na osnovnoj razini zrnatosti.
- ◆ Najjednostavniji oblik zvijezda spoja ima samo jednu tablicu činjenica i ta tablica spremi podatke na samo jednoj (osnovnoj) razini zrnatosti, a ne sadrži nikakve sumarne (agregirane) podatke.
- ◆ Osnovna razina zrnatosti, na kojoj su prilično detaljni podaci, može imati vrlo velik broj zapisa, pa izvlačenje podataka s te razine zrnatosti ne daje optimalnu kvalitetu rada. - Zbrajanje velikog broja podataka iz tablice činjenica osnovne razine će dugo trajati.
- ◆ U skladištu podataka se stoga podaci unaprijed spremaju na **više razina zrnatosti**.

- ◆ **Agregirane činjenice (mjere)**, tj. činjenice koje su unaprijed sumirane i pohranjene, najčešće se dobivaju tako da se činjenice s osnovne razine zrnatosti zbrajaju, no činjenice se mogu i prebrojavati ili se može računati njihova minimalna, maksimalna ili srednja vrijednost i sl.
- ◆ Agregirane činjenice se najčešće spremaju u **posebne činjenične tablice**, odvojeno od podataka osnovne razine. Svaka agregacijska razina ima svoju činjeničnu tablicu.
- ◆ Pojedine **agregacijske tablice** se mogu staviti *off-line* i zatim opet *on-line*, a da to nema utjecaja na ostale podatke.

Modeli podataka za skladište podataka

Pohranjivanje sumarnih (agregiranih) podataka

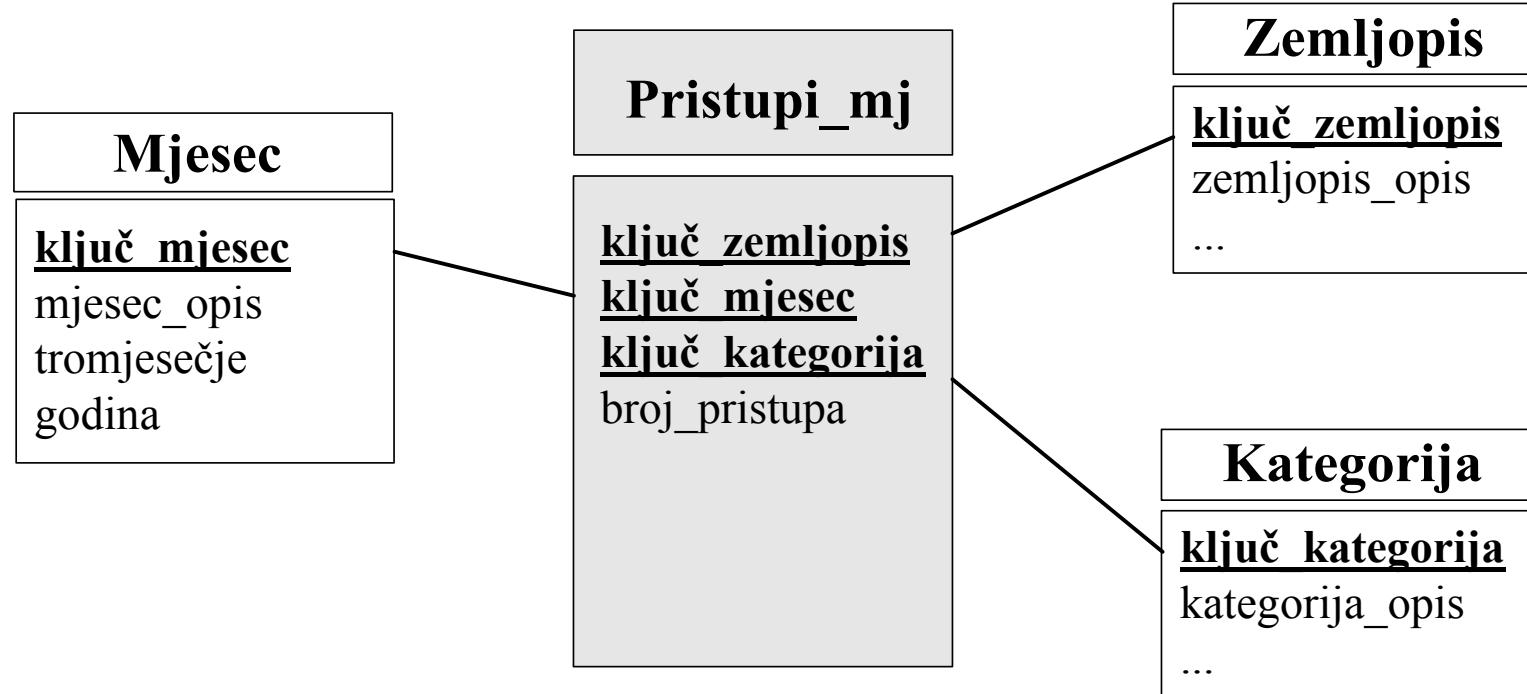
- ◆ Zvijezda spoj koji opisuje pristupe na www.hr stranice



- ◆ Primjer: želimo u posebnu agregacijsku tablicu pohraniti broj pristupa na *www.hr* po mjesecima

Modeli podataka za skladište podataka

Pohranjivanje sumarnih (agregiranih) podataka



- Umjesto dimenzijske tablice *Vrijeme* u shemu je uključena dimenzijska tablica *Mjesec* koja sadrži dio atributa tablice *Vrijeme* (uključeni samo atributi koji opisuju mjesec, tromjesečja i godine).
- Za dimenzijsku tablicu *Mjesec* treba odrediti primarni ključ.

Pohranjivanje sumarnih (agregiranih) podataka

- ◆ **Smanjene (izvedene) dimenzijske tablice** su važne za brzinu pretraživanja pri upitima. Kad se upitom traže podaci za određeni dan u tjednu, pri izvedbi upita se vidi da u tablici *Mjesec* nema atributa *dan_u_tjednu* i tada se ide na tablicu *Vrijeme* koja taj atribut sadrži.
- ◆ U ovakvom pristupu, krajnji korisnik ne vidi agregacijske činjenične tablice, već samo tablicu činjenica na osnovnoj razini.
- ◆ Sve SQL naredbe se odnose samo na osnovnu tablicu činjenica i njoj pridruženim dimenzijskim tablicama.

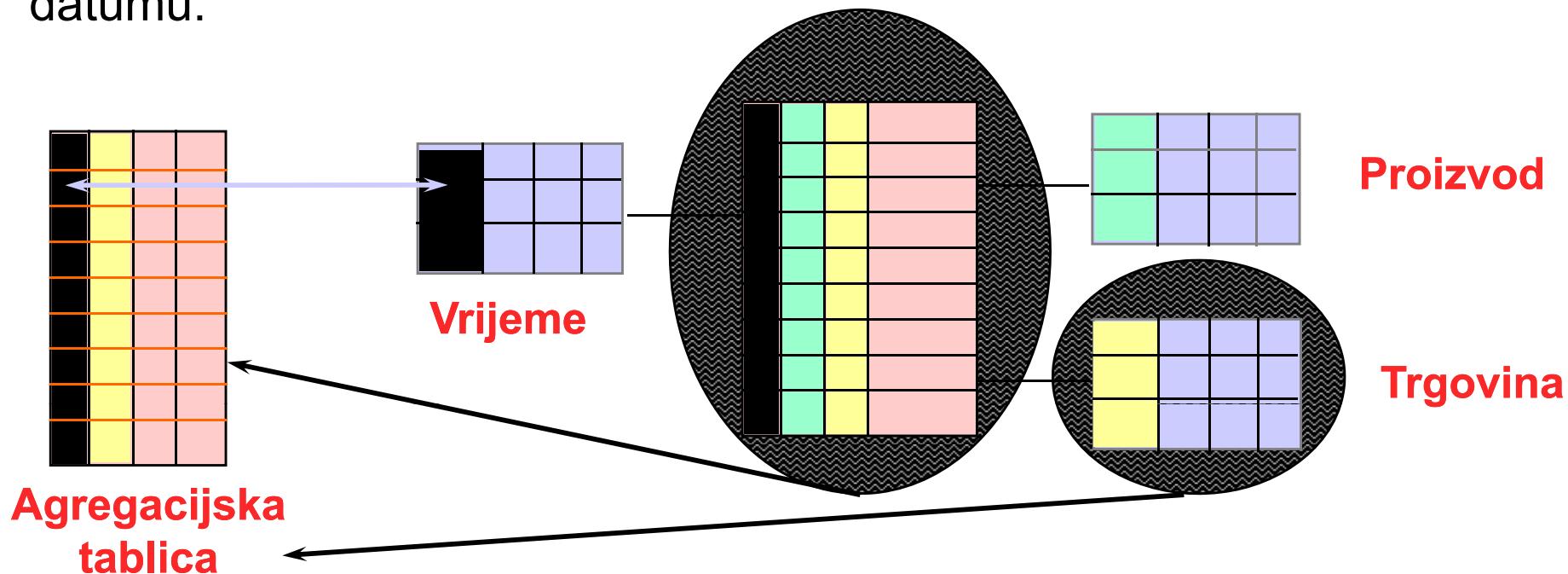
Upiti se izvode na sljedeći način:

1. Za svaku SQL naredbu **nađe se najmanja činjenična tablica** (tj. tablica s najmanjim brojem redaka) koja još nije ispitana, u skupu shema na koji se upit odnosi.
2. Ako se svi atributi koji su navedeni u SQL upitu mogu naći u toj činjeničnoj tablici i pridruženim joj smanjenim dimenzijskim tablicama, tada se u originalnim SQL naredbama **zamijene** odredišna **imena tablica stvarnim imenima** tablica.
 - Ako se bar jedan atribut iz SQL upita ne može naći u ispitivanim tablicama, traži se sljedeća veća činjenična tablica.
 - Ako upit nije riješen ranije, na kraju će se doći do osnovne činjenične tablice koja sigurno daje odgovor na upit.
3. **Pokreću se promijenjene SQL naredbe.**

Modeli podataka za skladište podataka

Pohranjivanje sumarnih (agregiranih) podataka

- ◆ Agregacijska tablica može imati manje ključnih atributa od osnovne činjenične tablice tj. može biti povezana s manje dimenzijskih tablica.
- ◆ Npr. ako sumiramo zaradu od prodaje svih proizvoda po trgovini i datumu:



- ◆ Primjeri agregacijskih tablica

Prodaja po:

- kategoriji proizvoda, danu i trgovini
- proizvodu, mjesecu i gradu
- tromjesečju i regiji
- ...

- ◆ Agregiranjem se uklanja potreba za ponavljanjem računanja koji bi se inače provodili svaki put kad se traži sumarna informacija i time se **znatno ubrzava izvođenje upita**.

- ◆ Međutim, agregiranje ima i svoje **nedostatke**:
 - zauzima se prilično **prostora na disku**,
 - kreiranjem novih tablica otežava se **upravljanje** skladištem,
 - ako se agregiranje provodi svaki put kad se dodaju novi podaci u činjeničnu tablicu, bit će potrebno dosta **vremena za učitavanje** podataka u skladište.

Treba obratiti pozornost na to koliko ima agregacijskih tablica i koliko je ukupno prostora na disku zauzeto agregatima.

Pohranjivanje sumarnih (agregiranih) podataka

Kad se određuje koje će se agregacijske tablice napraviti, potrebno je razmotriti:

- za koje hijerarhijske razine po dimenzijama korisnik **najviše postavlja upite**,
- gdje se podaci koncentriraju, tj. za koje hijerarhijske elemente u dimenzijskoj tablici je prosječni broj redaka relativno velik, odnosno za koje hijerarhijske razine se **sporo dobija odgovor**. Ako neki element hijerarhije u dimenzijskoj tablici ima daleko više redaka nego drugi elementi u hijerarhiji, agregacija na razini tog elementa bi drastično popravila izvedbu.

Pohranjivanje sumarnih (agregiranih) podataka

- ◆ Agregacijske tablice počnu se planirati već u ranim fazama oblikovanja **na osnovu korisničkih zahtjeva za upitima**. Ti se zahtjevi dokumentiraju, implementiraju i nadgledaju.
- ◆ Kad se skladište podataka da na korištenje, ispituje se na koji način su se **najčešće obavljala grupiranja podataka** (tj. koji su atributi najčešće bili u GROUP BY dijelu SQL naredbi). Dobivene informacije koriste se za kreiranje novih agregacijskih tablica (ili ukidanje postojećih).
- ◆ Dobar odabir agregacijskih tablica je jedan od ključnih faktora koji skladište podataka čine učinkovitim - korisnici mogu **brzo dobiti odgovore na svoje upite**.

Planiranje projekta skladištenja podataka

- ◆ Projekt skladištenja podataka je zahtjevniji od većine informatičkih projekata jer **nema konačan i statican skup zahtjeva** koji zadaje jedna osoba ili jedan odjel.
- ◆ Zahtjeve za projekt skladištenja podataka trebalo bi odrediti cjelokupno poslovno osoblje iz svih odjela neke tvrtke ili organizacije.
- ◆ Pri planiranju projekta potrebno je odgovoriti na četiri ključna pitanja:
 1. Što će biti rezultat projekta?
 2. Kad će to biti dovršeno?
 3. Koliko će to koštati?
 4. Tko će to napraviti?

Izgradnja skladišta podataka

Planiranje projekta skladištenja podataka



- ◆ Mogući rizici:
 - nedostatak potpore, odlučnosti i uključenosti od strane rukovodećeg osoblja
 - prekid financiranja ili smanjeno financiranje
 - nerealni rokovi, nerealan opseg rada, nerealna očekivanja, nerealan budžet
 - članovi tima nedovoljno obučeni, nedostatak stručnjaka za neka područja
 - stalne izmjene poslovnih prioriteta
 - neučinkovito upravljanje projektom

Izgradnja skladišta podataka

Definiranje projektnih zahtjeva

Razmotriti:

- ◆ Funkcionalnost
 - Koje informacije poslovni ljudi u organizaciji trebaju?
 - Na koja pitanja ne mogu trenutno dobiti odgovore?
 - Kakve izvještaje žele?
 - Koji izvještaji su najvažniji? Koji su manje važni?
 - Koje tipove upita će poslovni analitičari postavljati?
- ◆ Podaci
 - Koji podaci su poslovnim ljudima potrebni? Odakle ih trenutno dobivaju?
 - Koliko su podaci danas čisti? Koliko čisti bi trebali biti?
 - Koji podaci su najvažniji?
 - Mogu li se podaci sumirati? Ako mogu, po kojim dimenzijama?
 - Koliko detaljni podaci trebaju biti?
 - Kakva su očekivanja što se tiče raspoloživosti podataka?

Izgradnja skladišta podataka

Definiranje projektnih zahtjeva



- ◆ Povijest
 - Koliko godina stare povijesne podatke trebamo čuvati?
 - Možemo li početi skupljati podatke od ovog trenutka ili moramo učitavati i stare arhivirane podatke?
- ◆ Sigurnost
 - Kakva treba biti razina sigurnosti što se tiče podataka? Kako se provodi sigurnost u operacijskim izvorima podataka?
 - Trebaju li svi podaci jednaku razinu sigurnosti?
 - Tko smije pristupiti kojim podacima?
- ◆ Rad sustava
 - Koje najdulje vrijeme odgovora na upit će biti prihvatljivo?
 - Mogu li se izvješća kreirati tijekom noći da bi se smanjilo zagušenje sustava?
 - Koliko često i koliko dugo će poslovni analitičari pristupati skladištu podataka tijekom dana?

Izgradnja skladišta podataka

Analiza izvornih podataka

Razmotriti:

- ◆ Izvori podataka
 - Znamo li gdje su pohranjeni izvorišni podaci? U kojim sustavima? U kojim datotekama? U kojim bazama podataka?
 - Postoje li višestruki izvori za iste podatke?
 - Tko su vlasnici podataka?
 - Postoji li raspoloživa dokumentacija za izvore podataka? Je li dokumentacija potpuna i jesu li u njoj praćene sve izmjene u sustavu?
- ◆ Kvaliteta podataka
 - Znamo li koliko su podaci čisti?
 - Jesu li podaci dovoljno čisti za sve vrste korisnika skladišta podataka?
- ◆ Čišćenje podataka
 - Jesu li pogreške u podacima već dokumentirane?
 - Znamo li koji su podaci više, a koji manje važni?

Izgradnja skladišta podataka

Izrada prototipa

- Izrada **prototipa** skladišta podataka podrazumijeva brzu izgradnju (npr. u roku 2-3 mjeseca) manjeg skladišta podataka koje će obuhvaćati samo dijelove planiranog modela podataka i tek dio ukupne planirane funkcionalnosti sustava.

- Na taj način krajnji korisnici mogu u ranoj fazi izgradnje skladišta podataka:
 - vidjeti mogućnosti i ograničenja tehnologije,
 - analizirati funkcionalnost sustava,
 - pronaći nedostatke i neusklađenosti u postojećim zahtjevima,
 - proširiti ili mijenjati zahtjeve,
 - preciznije i realnije odrediti krajnje ciljeve projekta.

Izgradnja skladišta podataka

Izrada prototipa

- Kad krajnji korisnici na početku projekta razmišljaju o željenim rezultatima projekta, obično se ne mogu sjetiti svih detalja i nisu svjesni svih međuvisnosti poslovnih procesa i pripadajućih podataka.
- **Eksperimentiranje** s različitim načinima oblikovanja skladišta podataka, različitim sustavima za upravljanje bazom podataka, različitim metodama vizualizacije, razvojnim alatima ili tehnikama programiranja jeftinije je **u fazi izgradnje prototipa**.
- Rezultati koje dobijemo analizom prototipa pomažu pri odabiru najboljih metoda, sustava i alata koje zatim koristimo za izgradnju cijelog skladišta podataka.
- Ipak, na prototipu ne možemo testirati performanse jer prototip skladišta sadrži malu količinu podataka.

Izgradnja skladišta podataka

Izrada prototipa



Pri izgradnji prototipa potrebno je:

- ◆ **odabrati samo podskup funkcionalnosti sustava i podskup modela podataka**

Na taj način se prototip može relativno brzo isporučiti korisnicima, oni mogu shvatiti njegove mogućnosti i ograničenja, što im pomaže da razumiju način korištenja i mogućnosti budućeg cjelovitog skladišta podataka.

- ◆ **razumjeti zahtjeve nad bazom podataka**

Prototip će administratoru baze podataka pomoći da shvati na koji način krajnji korisnici žele pristupati bazi, koju razinu detalja traže, koji podaci su najzanimljiviji... Administrator će moći u određenoj mjeri predvidjeti brzinu izvedbe određenih upita, kao i veličinu skladišta podataka. Sve te informacije pomoći će administratoru da poduzme određene akcije u cilju optimizacije sustava.

Izgradnja skladišta podataka

Izrada prototipa



Pri izgradnji prototipa potrebno je (nastavak):

- ◆ **odabrati prave podatke**

Potrebno je odabrati one podatke koji su bitni za analizu i na kojima ćemo moći testirati željene funkcionalnosti sustava. Ti podaci moraju biti čisti, a količina podataka mora biti mala da ne bismo gubili vrijeme na čišćenje i dugotrajno učitavanje podataka.

- ◆ **testirati korištenje alata, uključiti krajnje korisnike**

Potrebno je krajnje korisnike obučiti za kreiranje izvještaja i za rad s OLAP alatom te zatim analizirati rad krajnjih korisnika sa sustavom, pratiti njihove reakcije. Testirati također kako se sustav ponaša kad broj korisnika raste.

Izgradnja skladišta podataka

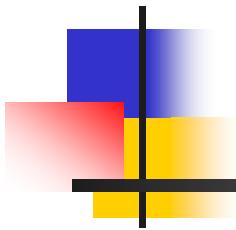
Zaključak



- ◆ dugoročno planiranje razvoja čitavog sustava
 - ◆ ispravna metodologija
 - ◆ implementacija - dio po dio, u malim koracima i na iterativan način, sa što učestalijim povratnim informacijama od krajnjih korisnika
(što im se sviđa? što im se ne sviđa? je li korištenje jednostavno? što nedostaje?)
- ⇒ dijelovi sustava mogu se relativno brzo dati na korištenje, a sustav čini kompaktnu cjelinu i odgovara potrebama krajnjih korisnika

Koje od navedenih izjava su NETOČNE?

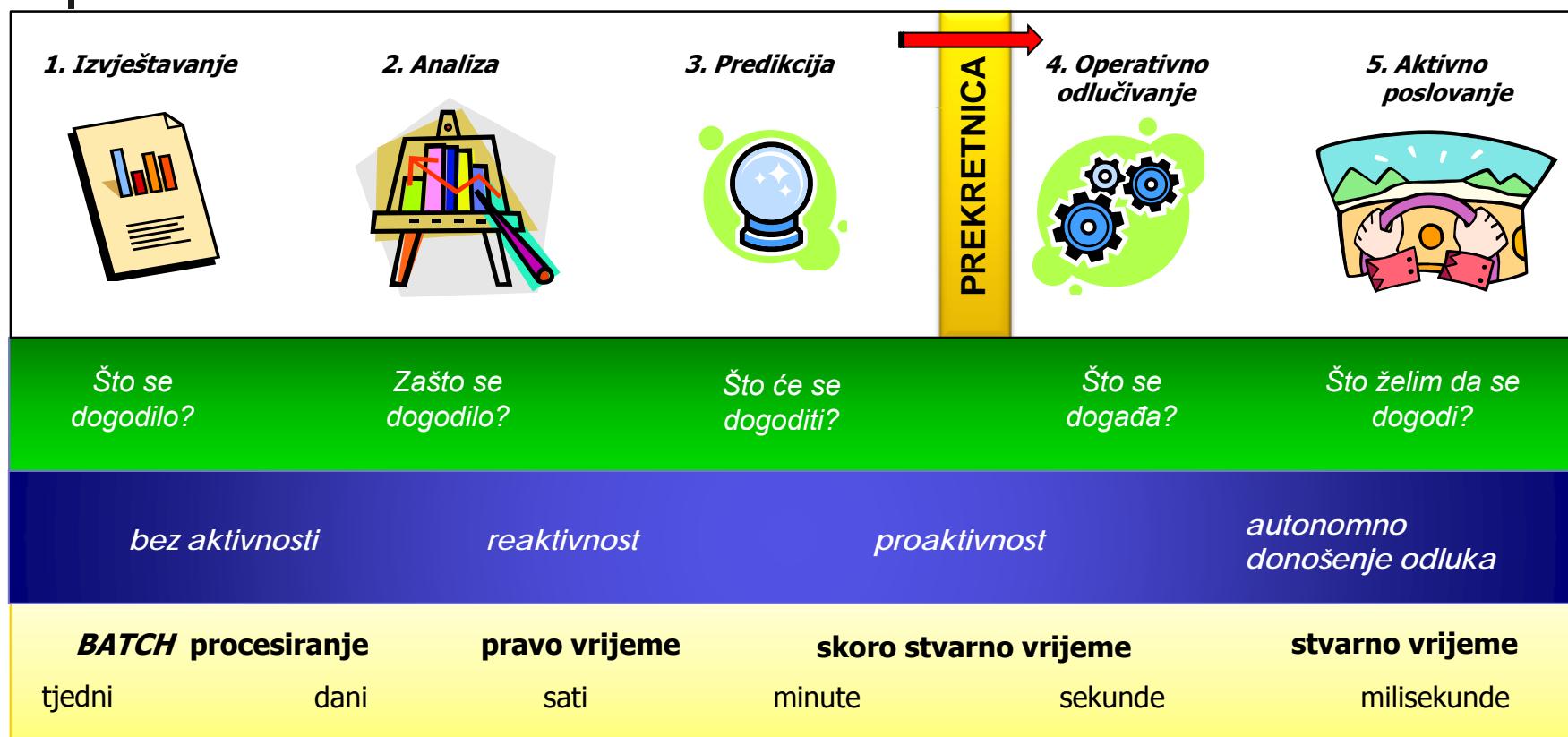
- A. Zvezdasta shema se sastoji od jedne činjenične tablice i više dimenzijskih tablica.
- B. Dimenzijske tablice trebaju zadovoljavati 3. normalnu formu.
- C. Među glavnim atributima hijerarhijskih razina u dimenzijama postoji funkcionalna zavisnost, odnosno veza tipa N:1.
- D. Hjerarhijske razine u dimenzijskim tablicama omogućavaju dobivanje detaljnijeg ili sumarnijeg višedimenzijskog pogleda na podatke.
- E. Dimenzijski atributi su u pravilu brojčani, iz kontinuiranog skupa vrijednosti i poželjno je da su zbrojivi.
- F. Agregacijske tablice se rade za one hijerarhijske razine u dimenzijama za koje se najviše postavljaju upiti, a sporo se dobijaju odgovori.

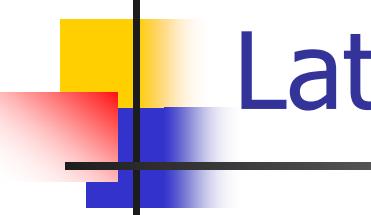


Stvarnovremensko skladištenje podataka

Upravljanje podacima
2015./2016.

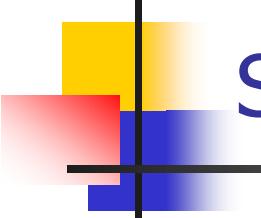
Poslovno odlučivanje i skladište podataka - evolucija





Latencija

"Vremenski interval između pojave poslovnog događaja i trenutka kada poslovni događaj postaje adekvatno evidentiran u skladištu podataka te kao takav dostupan za potrebe poslovnih analiza."



Stvarnovremensko skladištenje podataka

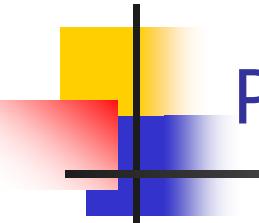
Zašto stvarnovremensko skladištenje podataka
(eng. *Real Time Data Warehousing*)?

- zato što to poslovni korisnici zahtjevaju
 - globalno 24/7 poslovanje naspram lokalnog "od 9 do 5"
 - potrebne su prave informacije u pravom trenutku
 - pristup informacijama što jednostavniji (sve informacije na jednom mjestu)
- zato što to tehnologija omogućuje
 - porast hardverskih i softverskih mogućnosti
 - ono što je bilo nepojmljivo prije 15 godina danas je normalno (primjer: FULL HD video preko Interneta)



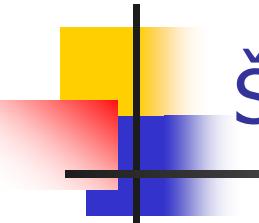
Tradicionalni DW vs RT DW

- **tradicionalni** skladišni sustav učitava podatke u skladište tijekom "mirovanja" poslovanja (*downtime*) kada neće opterećivati postojeće informatičke resurse – tzv. *batch loading*
- skladište predstavlja sliku povijesnih podataka
- skladište podataka služi kao potpora za strateško, dugoročno odlučivanje
- postoji stroga i vidljiva razlika između "operativne" i "analitičke" informatičke podrške poduzeća
- **stvarnovremenski** skladišni sustav konstantno učitava podatke u skladište – tzv. *stream loading*
- skladište predstavlja sliku povijesnih podataka ali i prikaz trenutnih poslovnih događaja
- skladište podataka je potpora za strateško (dugoročno) ali i taktičko (operativno) odlučivanje
- operativni i analitički sustavi predstavljaju zatvorenu petlju



Primjeri potrebe za RT-DW

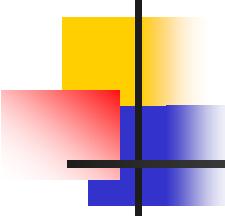
- globalno lansiranje novog uređaja (*gadget-a*)
 - zbog vremenske razlike prodaja prvo počinje u Aziji, potom u Europi te na kraju u Americi
 - voditelje poslovnica na zapadu zanimaju podaci o prodaji kako bi **prilagodili uvjete prodaje** na svojem tržištu
- aviokompanije - praćenje ponašanja putnika
 - trenutna situacija u svijetu zahtjeva brzu i učinkovitu **identifikaciju ilegalnih aktivnosti**
- kartične kuće
 - potreba analiza transakcija u stvarnom vremenu kako bi se **spriječila zlouporaba** (uočavanje zlouporabe prije nego počinitelj napusti trgovinu)
- financijsko i burzovno poslovanje
 - potrebna brza **reakcija na promjene** na globalnom tržištu



Što je zapravo "stvarno vrijeme"?

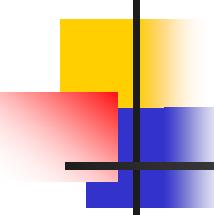
- četiri definicije ovisne o kontekstu:
 - na vrijeme (*on time*)
 - pravo vrijeme (*right time*)
 - gotovo stvarno vrijeme (*near real time*)
 - pravo stvarno vrijeme (*true real time*)

- *Kimball* (webinar o skladištenju):
"Stvarno vrijeme je sve što je prebrzo za vaš trenutni ETL sustav"



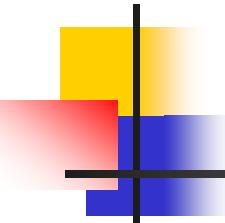
Načini prikupljanja i pohranjivanja podataka u stvarnom vremenu

- **povećavanje frekvencije učitavanja ETL sustava**
 - npr. učitavanje dva puta dnevno
- **direktno učitavanje (*Direct Trickle Feed*)**
 - prikupljati nove podatke u stvarnom vremenu , transformirati ih i unositi u skladište podataka
 - problem skalabilnosti –skladište je optimizirano za upite, ne za česti unos i ažuriranje podataka - preopterećenje sustava
- **učitavanje i preklapanje (*Trickle & Flip*)**
 - podaci se stalno učitavaju u tablice koje se nalaze izvan skladišta
 - u određenim vremenskim intervalima podaci se ažuriraju u skladištu
- **stvarnovremenska particija**
 - zasebna stvarnovrem. baza koja je optimizirana za česta učitavanja i stvarnovrem. režim rada - može biti i izvan SUBP-a gdje je skladište
 - nema degradacije performansi i problema skalabilnosti, ali ima više posla oko održavanja i prilagođavanja upita



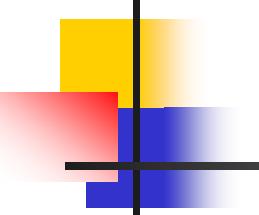
Izazovi i problemi implementacije stvarnovremenskog skladištenja

- **Kako riješiti probleme skalabilnosti?**
- konstantno učitavanje i ažuriranje podataka predstavlja dodatni teret na (ograničene) dostupne resurse tako da uvođenje stvarnovremenskog skladištenja može onesposobiti skladišni sustav
- moguća rješenja:
 - ograničiti uporabu stvarnovremenskih podataka
 - investirati u dodatne resurse
 - izolirati stvarnovremenske podatke
 - uvesti dodatne mehanizme/infrastrukturu dizajniranu upravo za ovu svrhu



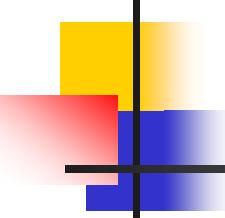
Izazovi i problemi implementacije stvarnovremenskog skladištenja

- **Kako registrirati pojavu novih podataka?**
- tradicionalni ETL sustavi prikupljaju podatke "odjednom" i to u "točno dogovoreno vrijeme", a stvarnovremenski zahtijevaju mehanizme prepoznavanja pojave novih podataka
- moguća rješenja:
 - n-minutni ciklusi provjere pojave novih podataka
 - mehanizmi objave novih podataka (*monitoring & triggering*)
 - može se koristiti tzv. prag objave (*alert threshold*) - pojave interesantnog poslovnog događaja se treba dobro definirati npr. obavijest da su zalihe pale ispod 5% treba poslati samo jednom, a ne na kraju svakog ciklusa



Postojeće implementacije stvarnovremenskog skladištenja

- veliki proizvođači softvera rano su prepoznali potrebu za stvarnovremenskim skladištenjem
 - najnovije verzije softvera za implementaciju skladišnih sustava uzimaju u obzir zahtjeve za stvarnovremenskim skladištenjem te najčešće nude podršku za to
 - pojavljuju se tvrtke sa **specijaliziranim rješenjima** strogo orijentiranim prema stvarnovremenskom skladištenju (npr. Netezza)
- problem prilagodbe i opravdanosti uvođenja stvarnovremenskog skladištenja
 - zamjena i/ili prilagodba postojećeg skladišnog sustava može predstavljati **veliku i riskantnu investiciju** i stoga **mora** biti opravdana
 - potrebna je pažljiva procjena poslovnih potreba te ocjena pogodnosti alternativnih rješenja



Zaključak

- stvarnovremensko skladištenje je danas tehnološki izvedivo i opravdano zahtjevima modernog elektroničkog poslovanja
- ključno je uskladiti poslovne zahtjeve, potrebe i tehnološke mogućnosti izvedbe
- poduzeća moraju donijeti (tešku) odluku:
 - treba li nam stvarnovremensko skladištenje?
 - da li je opravdana investicija u novi sustav?
 - može li se realizirati alternativno i prihvatljivije rješenje pomoću dostupnih otvorenih tehnologija?



Diplomski studij

Informacijska i
komunikacijska tehnologija:

Telekomunikacije i informatika



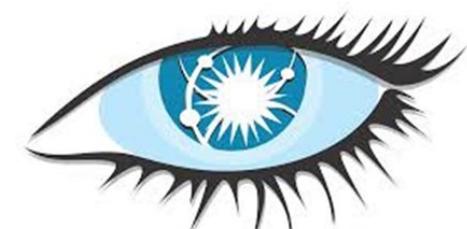
Upravljanje podacima

APACHE CASSANDRA DATABASE

dr. sc. Marko Banek

Ericsson Nikola Tesla d.d.

Lipanj 2016.



cassandra

PREGLED



- › Big Data kao izazov; skalabilnost; CAP teorem
- › osnovni koncepti u Cassandri: particioniranje; razine konzistentnosti; pisanje; čitanje
- › modeliranje podataka u Cassandri



BIG DATA KAO
IZAZOV

BIG DATA | 3V

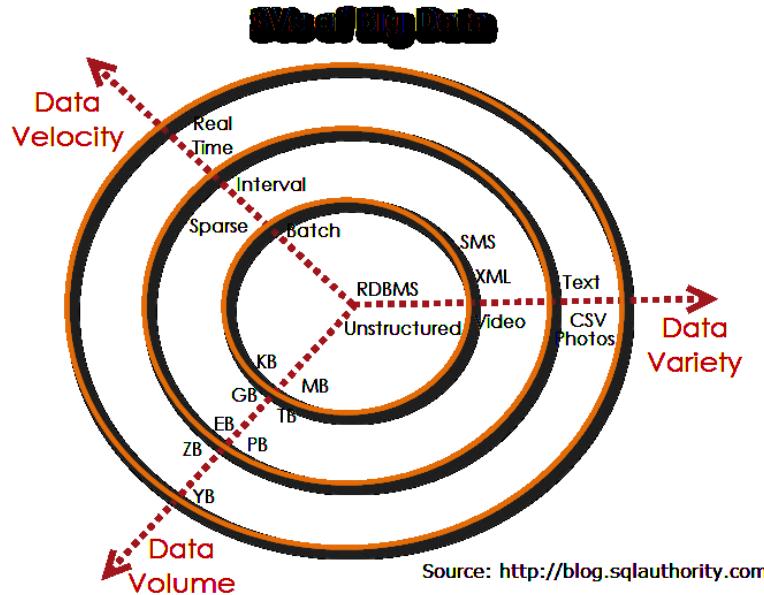


Gartner (2001, 2012):

„Big data is

- › high VOLUME,
- › high VELOCITY, and/or
- › high VARIETY

information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization”



IZAZOV BIG DATA U BAZAMA PODATAKA



- › raspoloživost u više geografskih regija
 - › nepostojanje točke u kojoj cijeli sustav pada (*no single point of failure*)
 - › jednostavna skalabilnost
 - › vrlo brzi i pouzdani odgovori
- } specifična arhitektura (NoSQL)
- } raspodijeljeni sustav

SKALABILNOST



› vertikalna

- dodavanje veće snage JEDNOM postojećem stroju
 - › veći disk(ovi) tj. više GB raspoloživih za pohranu
 - › snažniji CPU tj. veći broj jezgara
 - › više radne memorije
- s obzirom na veličinu(VOLUME) i zahtjeve na performanse tj. brzinu (VELOCITY), mogućnosti vertikalnog skaliranja su ograničene

› horizontalna

- dodavanje novih strojeva strojeva u skup resursa kojima raspolažemo
- **RASPODIJELJENI SUSTAV**

SKALABILNOST



› 2 osnovne mjere performansi baze podataka

- kašnjenje (eng. *latency*) – vrijeme potrebno za obradu zahtjeva nad bazom podataka (mjera: jedinice vremena tj. µs, ms, s)
- propusnost (eng. *throughput*) – broj uspješno obavljenih operacija čitanja/pisanja (mjera: #operacija/s)
 - › često se govori o broju operacija u sekundi po jednoj procesorskoj jezgri

SKALABILNOST



- › procesi u raspodijeljenom sustavu
 - koordinacija (identificiranje čvor(ov)a koji će obraditi podatke)
 - izvođenje operacija čitanja/pisanja na čvorovima
 - mrežni promet
- › (skoro-)linearna skalabilnost
 - dodavanje K% više čvorova povećava propusnost za K%
 - kašnjenje ostaje isto (tj. ne povećava se)
 - **ZAKLJUČAK:** uz linearu skalabilnost, dodavanje novih čvorova neće uzrokovati da se više vremena i procesnih resursa troši na koordinaciju

SKALABILNOST



Cassandra (crveno)

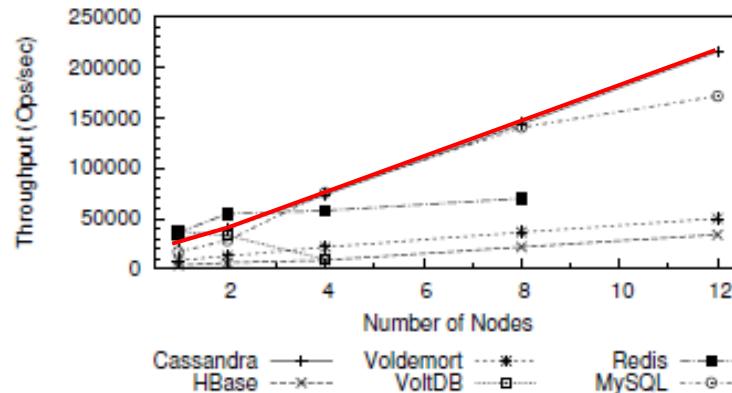


Figure 6: Throughput for Workload RW

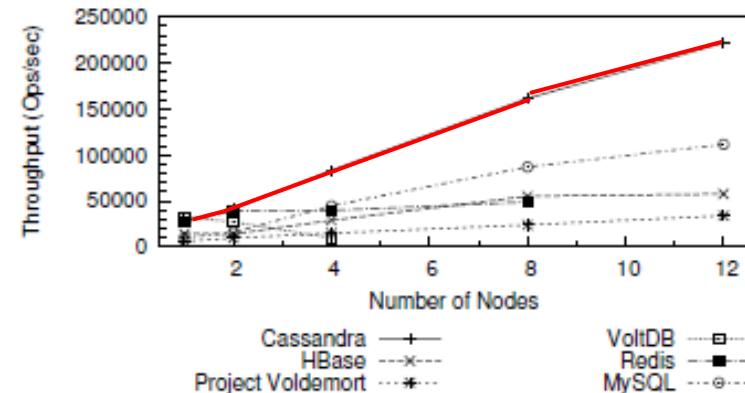


Figure 9: Throughput for Workload W

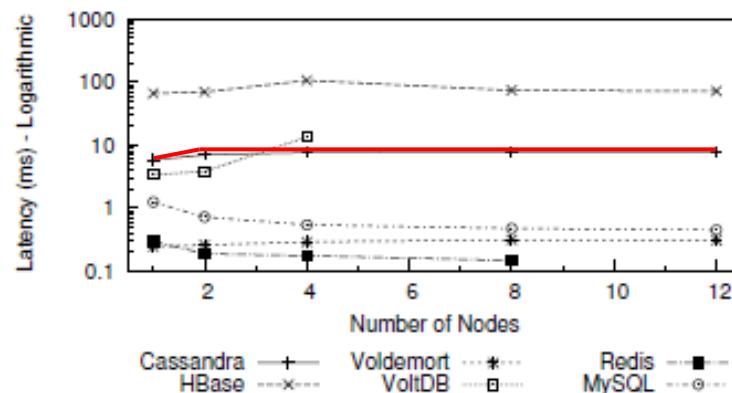


Figure 7: Read latency for Workload RW

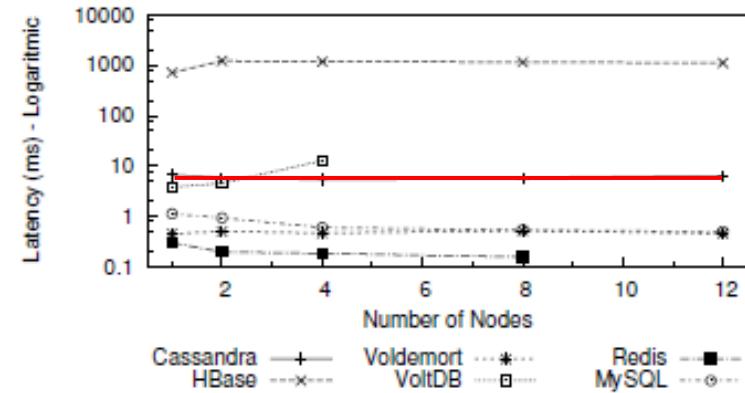
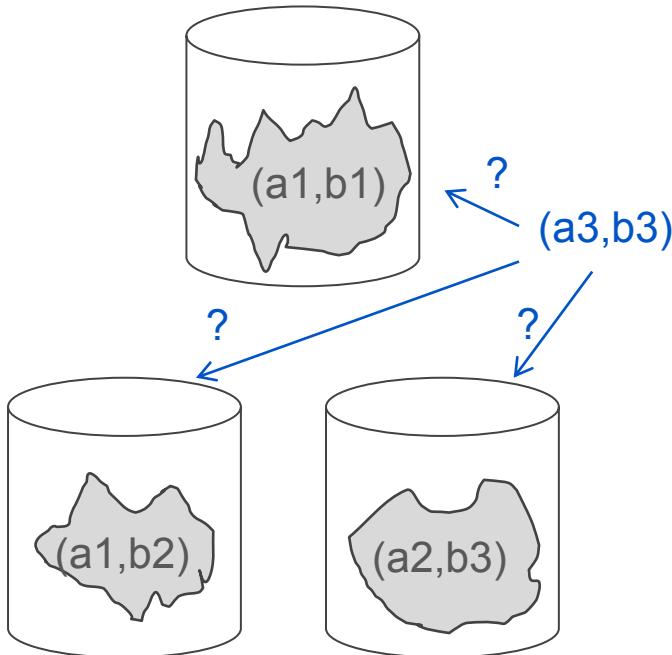


Figure 10: Read latency for Workload W

ORGANIZIRANJE PARTICIJA U RASPODIJELJENIM BAZAMA PODATAKA



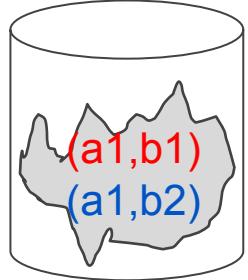
```
CREATE TABLE X (  
    a INTEGER,  
    b INTEGER,  
    c INTEGER,  
    PRIMARY KEY (a, b))
```

(standardna naredba SQL CREATE TABLE)

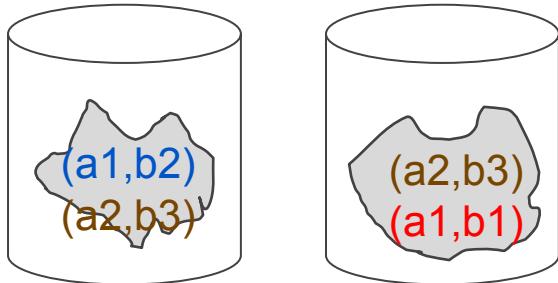
uz pretpostavku faktora replikacije 1:

- › svaki čvor pohranjuje dio ukupnog skupa podataka
- › treba uspostaviti uslugu particioniranja/koordiniranja kako bi se upiti usmjerili na čvorove „vlasnike“ podataka (one na kojima su podaci)
- › kada čvor padne, dio podataka postaje nedostupan

ORGANIZIRANJE PARTICIJA U RASPODIJELJENIM BAZAMA PODATAKA



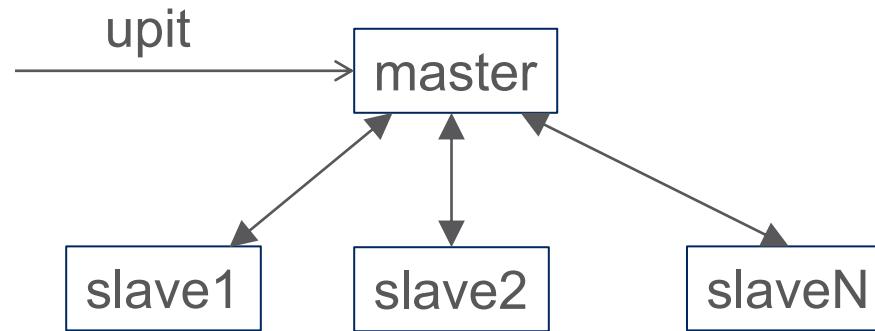
- › replikacijski faktor (RF) >1
- › sadržaj svakog čvora ima kopiju i na drugim čvorovima
- › osim usluge particioniranja/koordiniranja potrebno uspostaviti mehanizme održavanja **KONZISTENTNOSTI**
- › standardni RF za Cassandru u produkciji: 3



RASPODIJELJENE BAZE PODATAKA

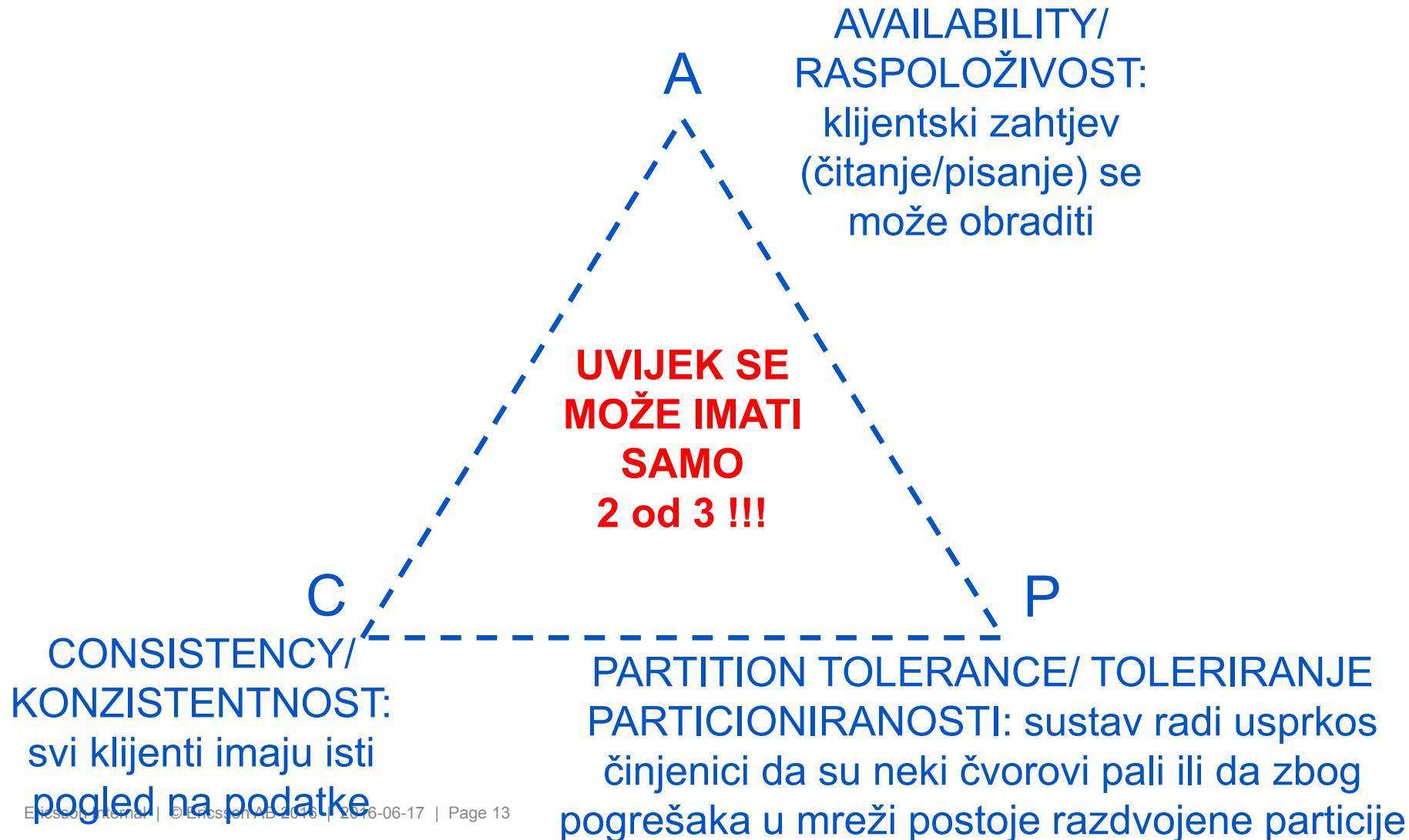


- › klasične relacijske baze zasnovane na jednom poslužitelju
- › proširenja za raspodijeljene okoline (PostgreSQL, MySQL...) zasnovane na mehanizmu *master-slave*



- › pad poslužitelja:
 - *slave*: ostatak sustava i dalje obrađuje upite
 - *master*: cijeli sustav prestaje s obradom upita (100% podataka palo)

CAP TEOREM





CAP TEOREM

› CA

- kod pojave particija sustav ne može ispravno raditi
- neraspodijeljeni sustav je CA
- *master/slave* konfiguracija u slučaju da padne *master*

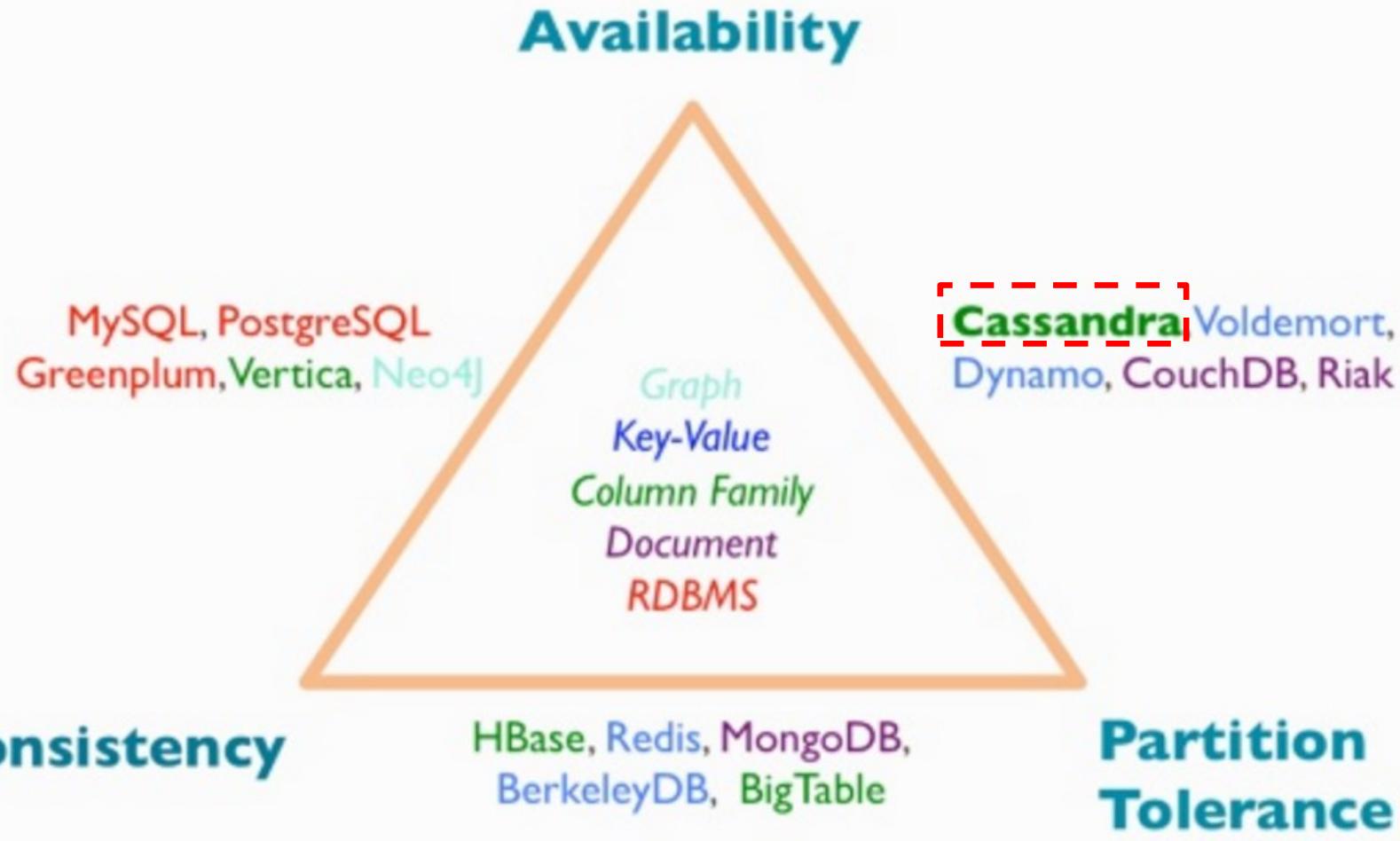
› CP

- u slučaju pada čvor(ov)a/mreže pojedini podaci su nedostupni; ostatak (većina) podataka ostaje raspoloživ i konzistentan
- konfiguracija *master/slave* u slučaju pada čvora *slave* ili mrežnog prekida

› AP

- raspoloživost se nastoji držati na 100% (dok broj padova čvorova/mrežnih prekida nije prevelik)
- na račun moguće nekonzistentnosti podataka
- ako se čvorovi/mreža mogu ponovno podignuti nekonzistentnosti će biti uklonjene nakon određenog vremena

BAZE PODATAKA I CAP TEOREM



<http://www.slideshare.net/ErickRamirez45/meetup-core-conceptserickramirez20150729-51122058>

ČETIRI OSNOVNA TIPOA NOSQL BAZA PODATAKA



- › graf (*graph*): svaki podatkovni element povezan je s n drugih u grafu/mreži
- › ključ-vrijednost (*key-value*): ključevi se mapiraju na skup bilo kakvih vrijednosti bilo kojeg tipa
- › dokument (*document*): skupovi dokumenata (JSON) nad kojima se mogu vršiti upiti (nad dijelom dokumenta ili dokumentom u cijelosti)
- › skup stupaca (*column family*): ključevi se mapiraju na skup n stupaca zadanog tipa
 - Cassandra

RASPROSTRANJENOST CASSANDRE



Alcatel-Lucent

BlackBerry



CALL-DUTY

CANONICAL



coursera

CREDIT SUISSE



Disney

DISQUS

ebay



Expedia

FedEx



GitHub



IBM



JBoss[®]
by Red Hat

KASPERSKY[®]

Microsoft



NETFLIX



paddypower

PayPal



Sony
Entertainment
Network

Spotify



Telefonica

The New York Times



Walmart[®]

NEKE OSNOVNE ČINJENICE



- › Cassandra je implementirana u Javi
- › ključni problemi s podešavanjem (*tuning*) Cassandre vezani su uz *Java Virtual Machine* (Java Heap) i *Garbage Collector*
- › za rad s podacima (DDL+DML) koristi se CQL (Cassandra Query Language), jezik sličan SQL-u
 - SELECT-FROM-WHERE-ORDER BY
 - INSERT, UPDATE, DELETE
 - CREATE TABLE, ALTER TABLE
 - tipovi podataka (INT, VARCHAR, BLOB...)



OSNOVNI POJMOVI U CASSANDRI

RJEŠAVANJE PROBLEMA „VOLUME & VELOCITY”



› FIZIČKA RAZINA

- rotacijski diskovi: glava diska pomiče se s jednog dijela diska na drugi kod svake operacije pisanja/čitanja

› LOGIČKA RAZINA

- prije pisanja treba najprije pročitati vrijednosti primarnog ključa
- normalizacija zahtijeva spajanje prilikom čitanja; u RSUBP-ovima spajanja su spora, ali normalizacija je nužna kako bi se izbjeglo udvostručavanje podataka (anomalija unosa/izmjene/brisanja)
- ACID transakcije – nekompatibilne s AP stranom CAP trokuta

RJEŠAVANJE PROBLEMA „VOLUME & VELOCITY”



› operacija pisanja u Cassandri:

- RSUBP: disk+*commitlog*=redundantno i sigurno
- *commitlog*: samo dodavanje na kraj (*append-only*), čime se pomicanje glave diska svodi na minimum
- replikacijski faktor>1: više čvorova → više *commitlog*ova ILI diskova=redundantno i sigurno
- tablica spremljena u memoriji (*in-memory*) umjesto na disku kako bi se upiti čitanja obradili što brže
- nakon zapisivanja *in-memory*+*commitlog* → potvrди (*acknowledge*) operaciju pisanja
- tablica iz memorije zapisuje se (*flush*) na disk asinkrono (kasnije, mnogo vremena nakon što je operacija pisanja potvrđena)
- pisanje je u pravilu brže od čitanja!

RJEŠAVANJE PROBLEMA „VOLUME & VELOCITY”



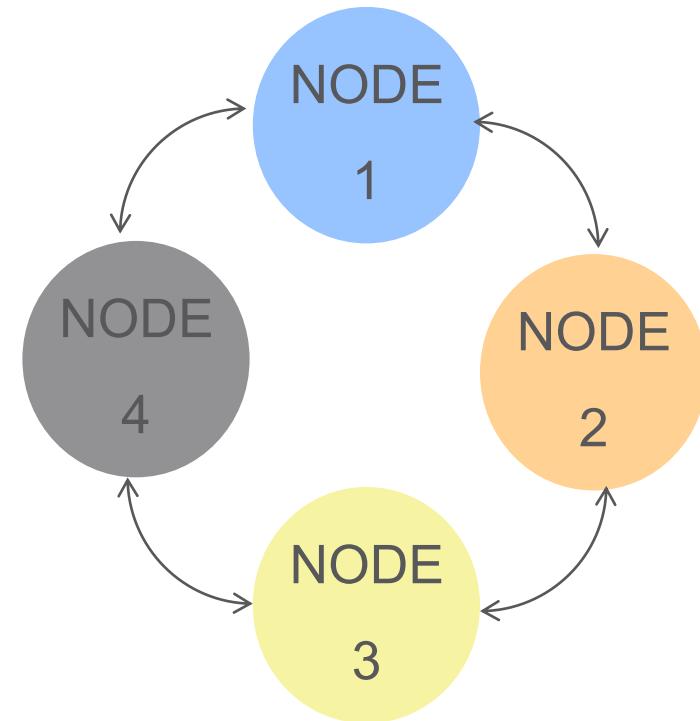
› Cassandra:

- BASE (konačna konzistentnost; *eventual consistency*) umjesto ACID
 - › pojedinačni insert/update/delete može se postaviti kao ACID
- bez provjere primarnog ključa (čitanje!) prije pisanja: prebrisivanje umjesto da se odbije izvršiti operacija; ne postoje strani ključevi
- nema SPAJANJA (JOIN) prilikom čitanja (*on-read*)
 - › spajanje moguće prilikom pisanja (tj. tablice su denormalizirane)
 - › spajanje se može izvesti naknadno, na klijentu (izbjegavati kod većeg opterećenja na čitanje i na širinu mreže)
- ograničeni uvjeti pretraživanja kod čitanja (tj. WHERE u SELECT): tablica je fizički spremljena kao indeks; upiti koji ne slijede indeksnu organizaciju su sintaksno zabranjeni
- nema GROUP BY; ograničena uporaba agreg. funkcija od v2.2

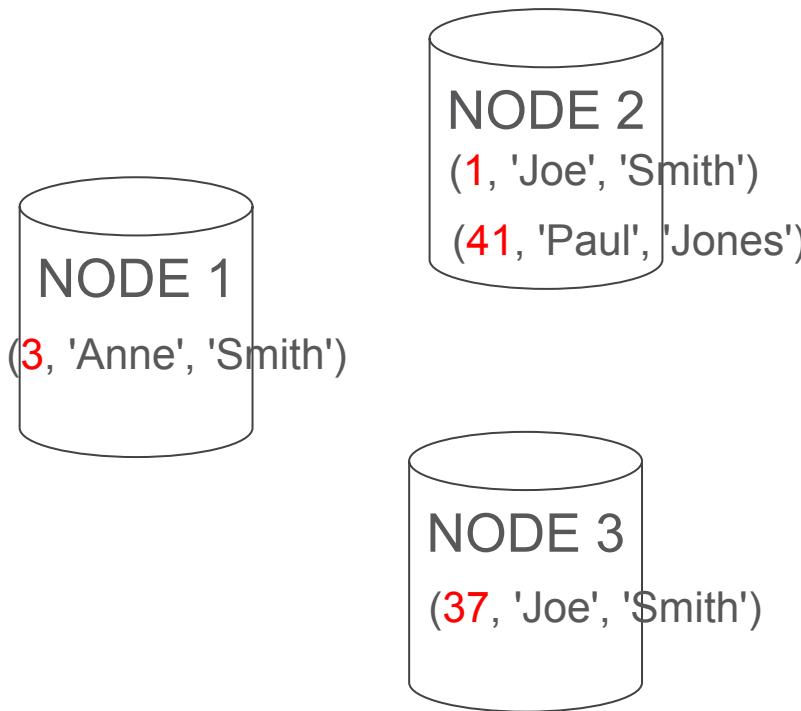
BAZA POTADAKA PREMA MODELU *PEER-TO-PEER*



- › svaki čvor koji radi može preuzeti i obaviti bilo koji klijentski zahtjev: čvor koordinator
- › osnovni odsječak podataka (*particija*) se nalazi na jednom ili više čvorova „vlasnika”
- › koordinator šalje interni upit na vlasnike particije; dobiva odgovore; šalje odgovor klijentu baze



MEHANIZAM PARTICIONIRANJA U CASSANDRI



```
CREATE TABLE PERSON (
    id int PRIMARY KEY,
    first_name varchar,
    last_name varchar
);
```

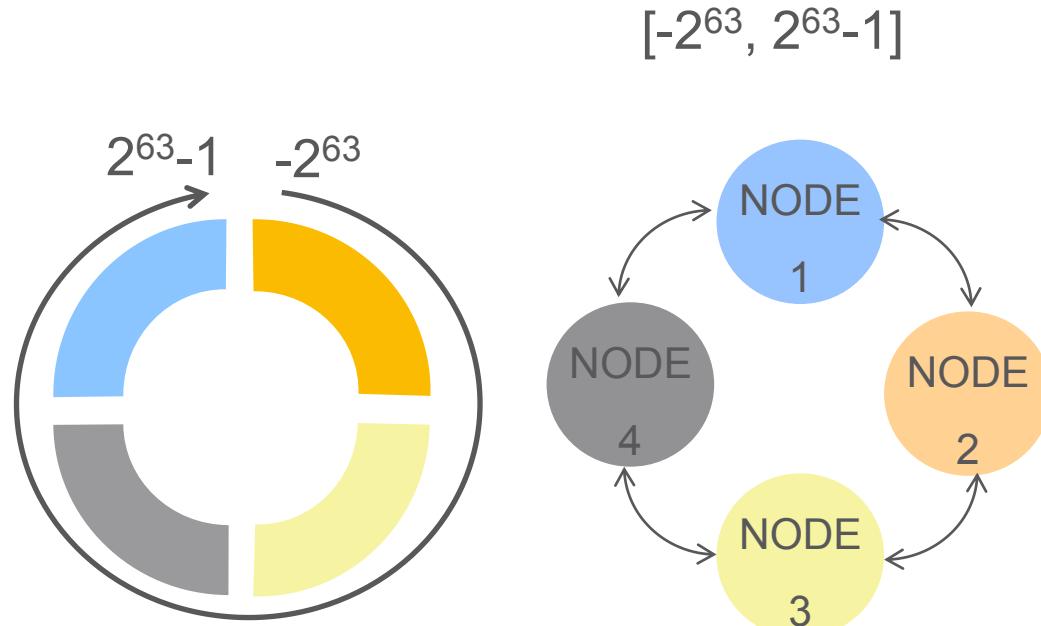
(CQL naredba CREATE TABLE –
CQL tip podataka varchar)

- › osnovna ideja: particioniranje zasnovano na vrijednosti primarnog ključa
- › interno, svi tipovi pohranjeni su kao nizovi bajtova (java.nio.ByteBuffer)
- › problem: funkcija mora nizu bajtova pridružiti diskretni skup čvorova {1, 2, ..., N}

CASSANDRA TOKEN RING



- › mehanizam particoniranja zasnovan na *hash* funkciji
- › niz bajtova kao reprezentacija bilo kojeg „primarnog ključa“
→ **hash** → identifikator (*token*)
- › skup identifikatora [Long.MIN_VALUE, Long.MAX_VALUE]
- › svaki čvor dobiva
podjednak dio
ukupnog skupa
identifikatora



CASSANDRA TOKEN RING

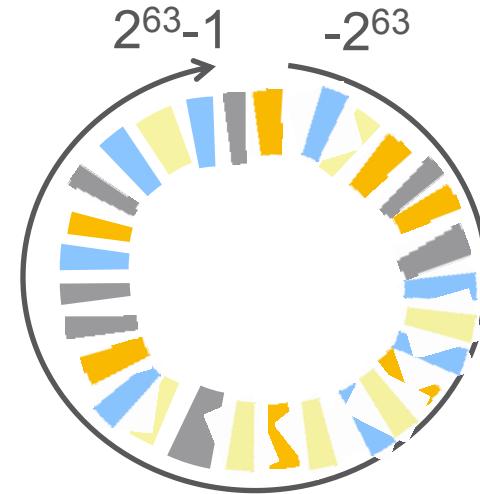


- › kad se dodaju novi čvorovi ili se miču postojeći, identifikatori se preraspodijele s obzirom na vlasnike
 - ručno
 - automatski, koristeći *virtualne čvorove*
- › podaci čiji su identifikatori promijenili vlasnika se šalju mrežom novim čvorovima vlasnicima (veliki promet!!!)

VIRTUALNI ČVOROVI



- › svaki od N čvorova vlasnik cca $1/N$ skupa identifikatora
- › kod dodavanja novog čvora svaki postojeći zadrži cca $N/(N+1)$ svog podskupa, dok cca $1/(N+1)$ daje novom podskup dodijeljen svakom pojedinom sloju podijeli se u 256 dijelova koji se zovi VIRTUALNI ČVOROVI
- › dodavanje 5. čvora u sustav od 4 čvora
 - 51/52 VN prelazi novom čvoru
 - 205/204 VN ostaje
 - svaki novi podskup sada se opet podijeli na 256 novih (manjih) VN



REPLIKACIJSKI FAKTOR I VLASNIŠTVO NAD IDENTIFIKATORIMA



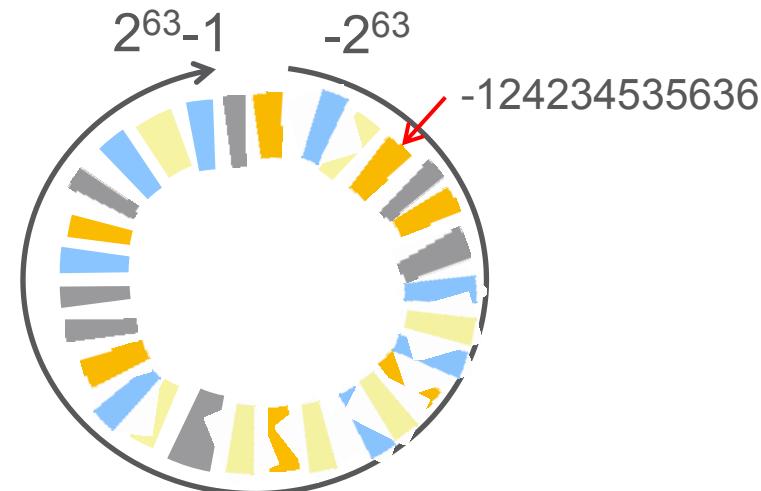
- › replikacijski faktor RF ($1 \leq RF \leq N_{NODES_TOTAL}$)
- › ne postoji „vlasnik” i „kopija”; sve kopije zovu se *replike*

RF=3

```
INSERT INTO PERSON  
(id, first_name, last_name) VALUES  
(501, 'John', 'Travolta');
```

501 HASH → -124234535636

-124234535636 u intervalu vlasnika 



REPLIKACIJSKI FAKTOR I VLASNIŠTVO NAD IDENTIFIKATORIMA



RF=3

```
INSERT INTO PERSON  
(id, first_name, last_name) VALUES  
(501, 'John', 'Travolta');
```

501 $\xrightarrow{\text{HASH}}$ -124234535636

-124234535636 u intervalu vlasnika

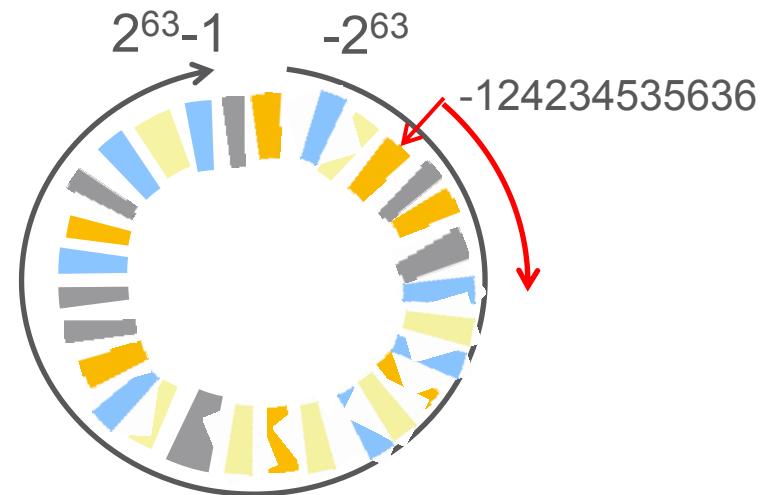
prva replika (za $RF \leq 1$) vlasnik

tražimo ASC idući VN koji ne pripada čvoru

druga replika (za $RF \leq 2$) vlasnik

tražimo ASC idući VN koji ne pripada

treća replika (za $RF \leq 3$) vlasnik





(DEMO 1)

- › konfiguracijske datoteke: cassandra.yaml; cassandra-env.sh
- › podizanje sustava s 3 ili 4 čvora
- › provjera statusa čvora (nodetool status)
- › CREATE TABLE PERSON s RF 3 + INSERT
- › izvršavanje selekcijskog upita (SELECT)
- › provjera koji su vlasnici replika za pojedine particije (nodetool getendpoints <keyspace_name> person <partition_value>)

HINTED HANDOFF („ODGOĐENA PRIMOPREDAJA”)



- › dok se izvršavaju operacije pisanja neki čvorovi mogu biti izvan funkcije
- › čvor koordinator lokalno sprema „nalog“ (*hint*) za pisanje koji će se izvršiti kad se uoči da se spomenuti čvor vratio u funkciju
- › ako je čvor izvan funkcije bez prestanka dulje od *max_hint_window_in_ms* (cassandra.yaml; default 10 800 000 ms tj. 3 sata) prestaju se spremati „nalozi“
 - **nastaju trajne nekonzistentnosti**
 - **potrebni su dodatni sinkronizacijski procesi: nodetool repair**

SINKRONIZACIJA NEKONZISTENTNIH PODATAKA

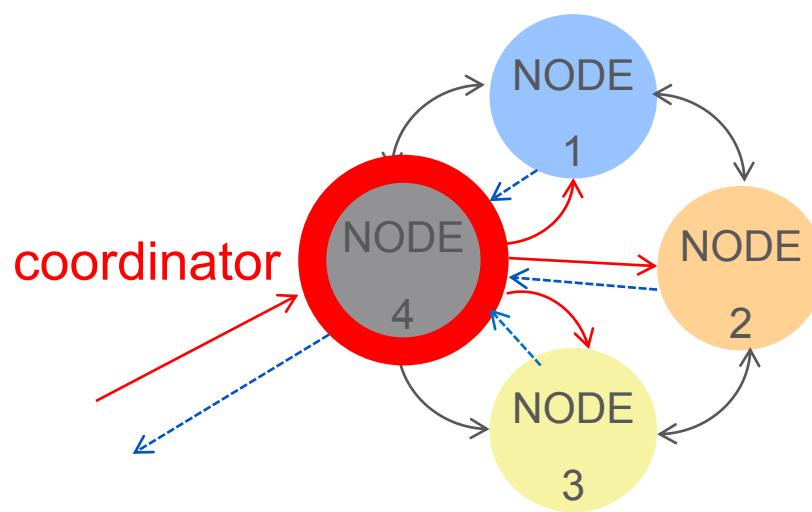


- › sve vrijednosti imaju vremenski žig (*timestamp*)
- › ako se na različitim čvorovima pročitaju međusobno različite vrijednosti, usporede se njihovi vremenski žigovi
 - vrati se najnovija vrijednost
 - na svim se replikama vrijednosti postave na najnoviju
 - tzv. *blocking read repair* (dodatno kašnjenje upita zbog procesa sinkronizacije)
 - sinkronizacija satova na čvorovima ima esencijalno značenje!

RAZINE KONSISTENTNOSTI



› OPERACIJA PISANJA s replikacijskim faktorom RF=3



→ node gossiping
→ request
→ response

- › zahtjevi sa pisanje idu na sve tri replike
- › može li koordinator potvrditi (ACK) pisanje klijentu prije nego SVE replike potvrde pisanje njemu?
- › što ako su neke replike pale?

RAZINE KONSISTENTNOSTI I „TRAČANJE”



- › čvorovi „tračaju” (*gossip*)
 - svake sekunde svaki čvor kontaktira 1-3 čvora u vezi njihovog statusa i pogleda sustava (tj. preostalih čvorova)
- › čvorovi koji ne šalju signale „tračanja” se od ostalih čvorova smatraju palima
- › i do 10 s je potrebno da se otkrije da je pojedini čvor pao; dakle čvor može pasti dok su ostali čvorovi neko vrijeme toga nesvjesni (te od njih dobiva zahtjeve kao replika podataka)

RAZINE KONSISTENTNOSTI



- › konzistentnost se može podešiti!

| WRITE | READ |
|--------|--------|
| ALL | ALL |
| ONE | ONE |
| TWO | TWO |
| QUORUM | QUORUM |
| ANY | - |

- sve replike moraju potvrditi operaciju
- jedna replika mora potvrditi
- dvije replike moraju potvrditi
- $\lfloor(RF/2)\rfloor+1$ tj. većina replika mora potvrditi
- potvrđuje se čak i ako su sve replike nedostupne; „nalozi“ (*hints*) za odgođenu pohranu spreme se na koordinatoru za sve replike

RAZINE KONSISTENTNOSTI



- › ALL – čim je jedna replika nedostupna zathjev se odbija
 - ALL čini Cassandru CA bazom (umjesto AP)
- › QUORUM: većina
 - za RF=3: ALL=3; QUORUM=2
 - ako je treća replika pala, a dvije potvrde, zahtjev je prošao, a koordinator za palu repliku spremi „odgođeni nalog“ (*hint*)

RAZINE KONSISTENTNOSTI : TRENUTNA KONSISTENTNOST



- › stvarna konzistentnost (jaka, trenutna; *true, strong, immediate*) ne treba i čitanje i pisanje na razini ALL
- › trenutna konzistentnost za #R+#W>RF
 - W:1; R:ALL
 - › sa svih replika se čita, uključujući i onu koja je potvrdila prethodno pisanje; njen vremenski žig je najnoviji (u slučaju postojanja diskrepancija)
 - W:ALL; R:1
 - W:QUORUM; R:QUORUM
 - › pisanje na više od pola replika; čitanje s više od pola replika; postoji bar jedna replika na kojoj su izvršene obje operacije, trenutna konzistentnost postignuta

RAZINE KONSISTENTNOSTI : TRENUTNA KONSISTENTNOST



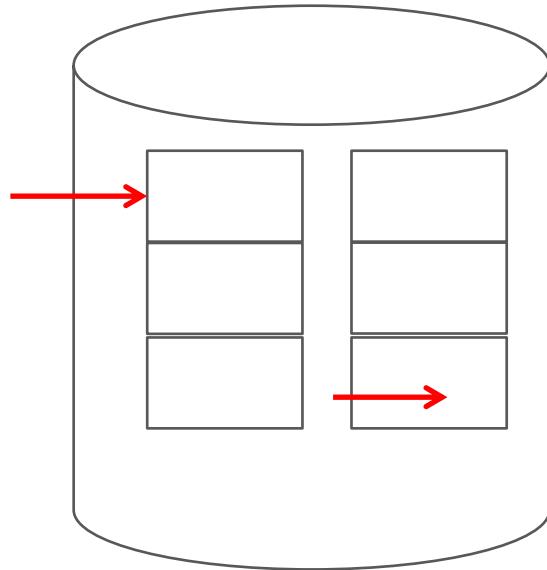
- › W:QUORUM; R:QUORUM je za $RF \geq 3$ bolja kombinacija od W:1; R:ALL or W:ALL; R:1
 - › dozvoljava da bar jedna replika bude privremeno nedostupna, a istodobno osigurava trenutnu konzistentnost (za $RF=3$, QUORUM=2 tj. jedna replika smije biti nedostupna)
- › trenutna konzistentnost ne osigurava ACID
 - › izoliranost nije postignuta
 - › trenutna konzistentnost znači da NAKON što je pisanje potvrđeno više nije moguće u nadolazećim zahtjevima čitati podatke koji su zastarjeli bez obzira što su neki čvorovi možda nedostupni ili su dostupni, ali se pisanje na njima još nije izvelo

RAZINE KONSISTENTNOSTI



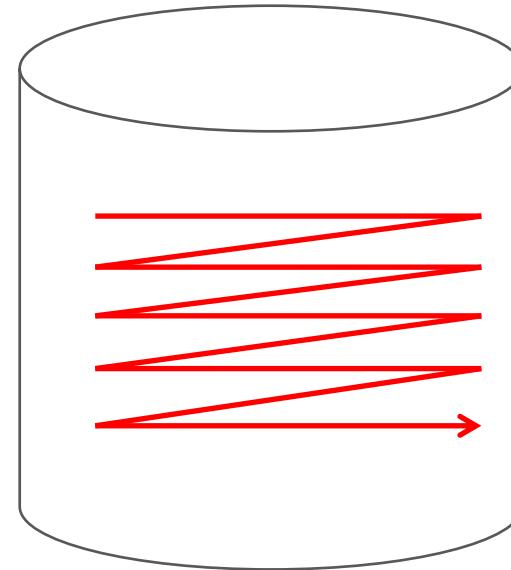
- › za brojne slučajeve uporabe koristi se opcija s najslabijom razinom konzistentnosti, ali i najkraćim vremenom obrade: W:ONE; R:ONE
- › s W:ONE; R:ONE nije moguće čitanjem slučajno pronaći diskrepancije između replika (neće se dogoditi *blocking read repair*)
 - *non-blocking read repair*: nakon svakog zahtjeva za čitanjem, s danom vjerojatnošću (default 10%) izvodi se asinkroni *read repair* zahtjev (tj. zahtjev za sinkronizacijom)
 - *non-blocking read repair*: u slučaju diskrepancije podataka, moguće da je prethodno čitanje vratio zastarjele podatke, ali buduća neće

PISANJE NA CASSANDRI



RDBMS:

Pretražuje i zapisuje vrijednosti na unaprijed definirane lokacije



Cassandra

Podaci se dodaju sekvencijalno na kraj datoteke

PISANJE NA CASSANDRI: OSNOVNI POJMOVI



- › **Memtables** – tablice u memoriji koje struktrom odgovaraju CQL tablicama
- › **CommitLog** – log-datoteka u koju se zapisuje samo na kraj, izvrsti se u slučaju kad je potrebno obnoviti *Memtablica* palog čvora
- › **SSTables** – snimke (*snapshots*) *Memtablica* koje se povremeno prebacuju (*flush*), na disk, pritom čisteći memoriju i commitlog
- › **Kompakcija** – periodički proces stapanja *SSTablica*

PISANJE NA CASSANDRI: OSNOVNI POJMOVI



› kad bilo koji čvor zaprimi zahtjev za pisanje

1. zapis se dodaje u *CommitLog*, and
2. zapis se dodaje u *Memtablicu* za odgovarajuću CQL tablicu
3. periodički, *Memtablice* se prebacuju na disk kao *SSTablice*, čisteći memoriju (JVM Heap) i *CommitLog*
4. periodički, izvodi se *Kompakcija* kako bi se stopile *SSTablice*

PISANJE NA CASSANDRI



```
INSERT INTO PERSON VALUES  
(32, 'Jane', 'Doe');
```

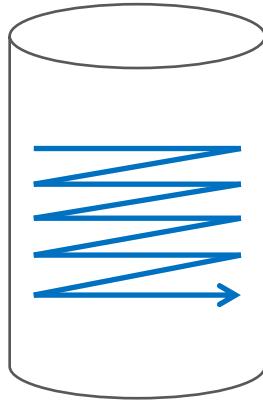


MemTable

| | | |
|------------------|-------|-------|
| Partition key 1 | Joe | Smith |
| Partition key 73 | Steve | Jones |

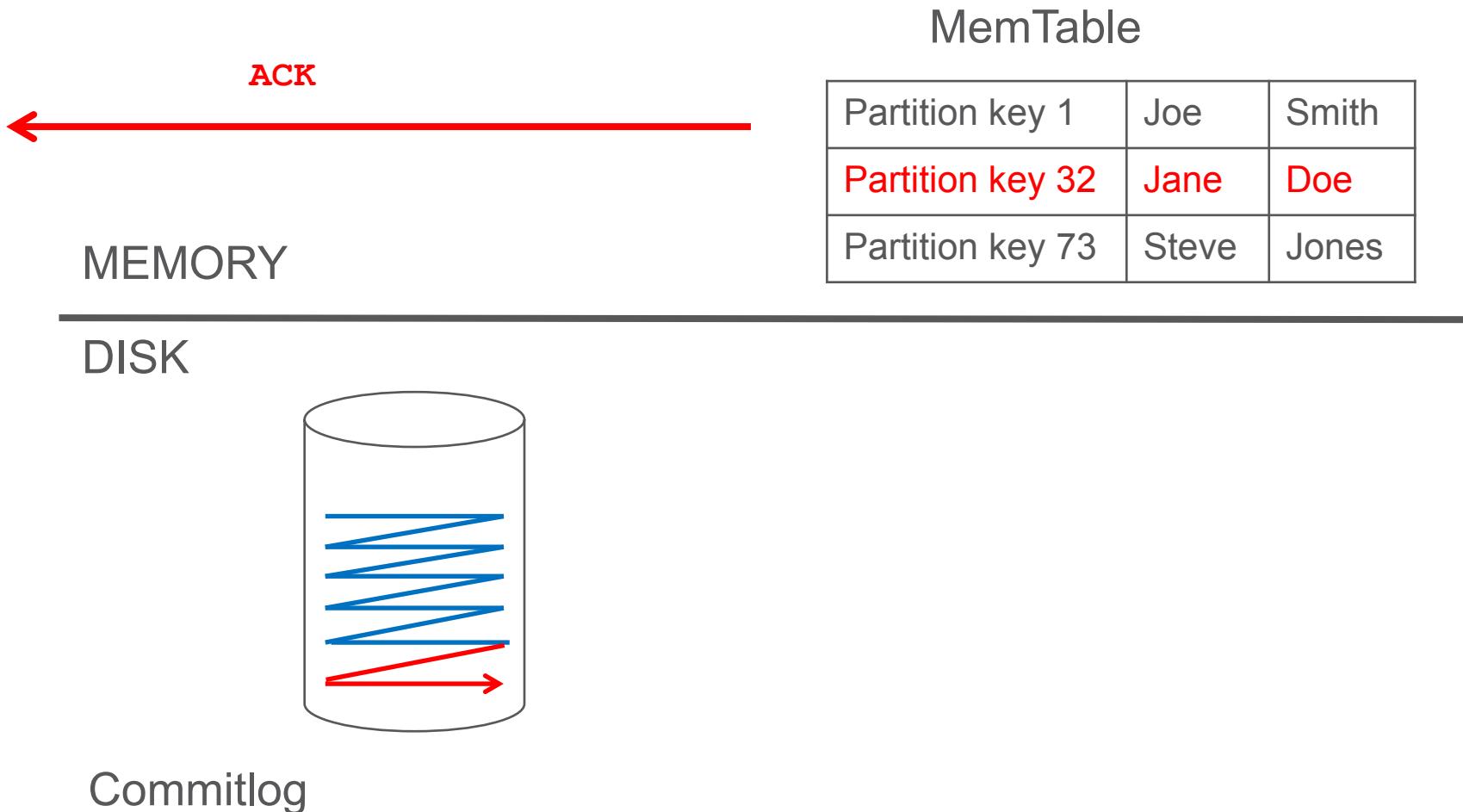
MEMORY

DISK



Commitlog

PISANJE NA CASSANDRI



PISANJE NA CASSANDRI: FLUSH

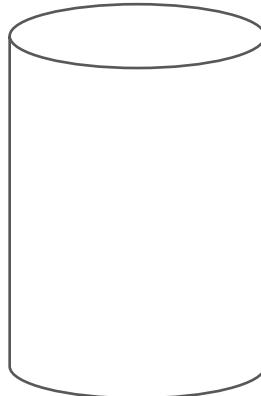


MEMORY

MemTable

| | | |
|--|--|--|
| | | |
| | | |
| | | |

DISK



SSTable

| | | |
|------------------|-------|-------|
| Partition key 1 | Joe | Smith |
| Partition key 32 | Jane | Doe |
| Partition key 73 | Steve | Jones |

Commitlog

- › *flush* čisti MemTablicu i Commitlog; stvara SSTablicu
- › SSTablica = Sorted String Table; nepromjenjiva

PISANJE NA CASSANDRI: FLUSH



- › SSTable
 - snimka (*snapshot*) of MemTablice u trenutku kad se izvede *flush*
 - nepromjenjiva (*immutable*)
- › u slučaju kasnije izmjene dotičnog retka i novog *flusha*
 - podaci koji odgovaraju pojedinačnom zapisu (tj. primarnom ključu) bit će u dvije ili više SSTablica

delete

- zastavica (nazvana *tombstone*, tj. *nadgrobni kamen* ili *stećak*) se postavi u SSTablici koja nastaje u sljedećem *flushu* kako bi se označilo da se svi ranije uneseni podaci koji se odnose na dani primarni ključ ne trebaju vraćati u čitanju; zastarjeli reci se drže u SSTablicama (nepromjenjive!) do kompakcije

PISANJE NA CASSANDRI: FLUSH



INSERT INTO PERSON VALUES
(32, 'Joseph', 'Smith'); **(FLUSH)**

| | | |
|------------------|--------|--------|
| Partition key 32 | Joseph | Smith |
| ts=123 | ts=123 | ts=123 |

UPDATE PERSON SET
first_name='Joe' WHERE id=32;
(FLUSH)

| | | |
|------------------|--------|--|
| Partition key 32 | Joe | |
| | ts=234 | |

DELETE FROM PERSON WHERE id=32;
(FLUSH)

| | | |
|------------------|----------|----------|
| Partition key 32 | Joe | Smith |
| ts=345 | 🚫 ts=345 | 🚫 ts=345 |

INSERT INTO PERSON VALUES
(32, 'Jane', 'Doe'); **(FLUSH)**

| | | |
|------------------|--------|--------|
| Partition key 32 | Jane | Doe |
| ts=456 | ts=456 | ts=456 |

UPDATE PERSON SET
first_name='Janice' WHERE id=32;
(FLUSH)

| | | |
|------------------|--------|--|
| Partition key 32 | Janice | |
| | ts=567 | |

UPDATE PERSON SET last_name='Smith'
WHERE id=32; **(FLUSH)**

| | | |
|------------------|--|--------|
| Partition key 32 | | Smith |
| | | ts=678 |

PISANJE NA CASSANDRI: KOMPACIJA



| | | |
|----------------------------|------------------|-----------------|
| Partition key 32 ts=123 | Joseph ts=123 | Smith ts=123 |
|----------------------------|------------------|-----------------|

| | | |
|------------------|---------------|--|
| Partition key 32 | Joe ts=234 | |
|------------------|---------------|--|

| | | |
|----------------------------|---------------|-----------------|
| Partition key 32 ts=345 | Joe ts=345 | Smith ts=345 |
|----------------------------|---------------|-----------------|

| | | |
|----------------------------|----------------|---------------|
| Partition key 32 ts=456 | Jane ts=456 | Doe ts=456 |
|----------------------------|----------------|---------------|

| | | |
|------------------|------------------|--|
| Partition key 32 | Janice ts=567 | |
|------------------|------------------|--|

| | | |
|------------------|--|-----------------|
| Partition key 32 | | Smith ts=678 |
|------------------|--|-----------------|

› primjer lijevo:

- 6 SSTablica
- 5 zapisa za first_name; samo jedan valjan
- 4 zapisa za last_name; samo jedan valjan

› pisanje na Cassandri je brzo, no stvara redundanciju:

- sporije čitanje
- veći utrošak diskovnog prostora

KOMPAKCIJA



- › pozadinski proces na pojedinom čvoru
 - nezavisno od kompakcija na drugim čvorovima
- › stapa sadržaj više SSTablica u jednu
 - poboljšanje performansi čitanja
 - oslobođanje diskovnog prostora
 - stare tablice se brišu; nove se stvaraju
- › najzahtjevniji proces za CPU koji postoji na Cassandri
 - okida se kad nema (mnogo) zahtjeva
 - jednodretvena ili višedretvena; u slučaju pristizanja velikog broja zahtjeva, prioritet kompakcije je najniži



KOMPACIJA

Partition key 32

| | | |
|--------|--------|--------|
| ts=123 | Joseph | Smith |
| ts=123 | ts=123 | ts=123 |

Partition key 32

| | | |
|--------|-----|--|
| ts=234 | Joe | |
| ts=234 | | |

Partition key 32

| | | |
|--------|--------|--------|
| ts=345 | Joe | Smith |
| ts=345 | ts=345 | ts=345 |

Partition key 32

| | | |
|--------|--------|--------|
| ts=456 | Jane | Doe |
| ts=456 | ts=456 | ts=456 |

Partition key 32

| | | |
|--|--------|--|
| | Janice | |
| | ts=567 | |

Partition key 32

| | | |
|--|--|--------|
| | | Smith |
| | | ts=678 |

Partition key 32

| | | |
|--------|--------|--------|
| ts=456 | Janice | Smith |
| ts=456 | ts=567 | ts=678 |

(DEMO 2)



- › izvesti *flush*
- › ubaciti (naredba INSERT) redak + izvesti *flush* (nekoliko puta)
- › istražiti datotečne strukture na disku
- › ubaciti još nekoliko novih redaka, za svaki redak napraviti *flush*, sve dok ne započne kompakcija

ČITANJE NA CASSANDRI

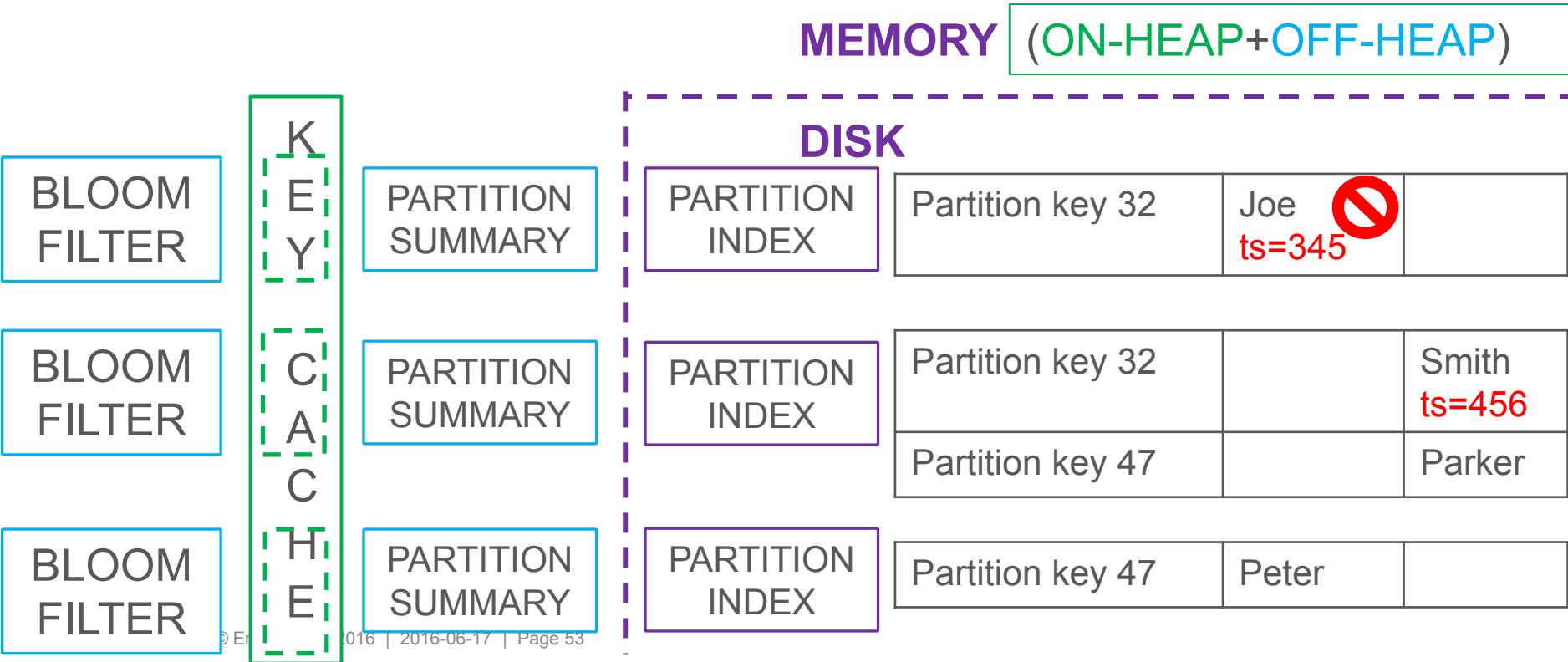


- › najnoviji podaci mogu biti u MemTablicama
 - MemTablice uvijek u memoriji
 - › u potpunosti na Java Heapu → brže, zauzima više mesta na Heapu
 - › dijelom na Heapu; dijelom izvan Heapa; sporije
- › ako vrijednost makar jedne kolone nije u MemTablici, mora se čitati iz SSTablica (s diska!)
- › postoji skup dodatnih *cache* i filterskih struktura da se smanji količina čitanja s diska
 - bloom filter (in-memory; off-heap)
 - key cache (in-memory; on-heap)
 - partition summary (in-memory; off-heap)
 - partition index (on-disk)

ČITANJE NA CASSANDRI



| | | |
|------------------|------|-----------------|
| Partition key 32 | | Baker ts=901 |
| Partition key 65 | Tony | Cooper |



ČITANJE NA CASSANDRI: PARTICIJSKI INDEKS



› SSTablice

- nepromjenjive binarne datoteke
- sortirano po partijskom ključu (i grupnom odn. *clustering* ključu unutar istog part. ključa: kasnije)

› partijski indeks: mapa između partijskog ključa i njegove točne pozicije u SSTablici

- štedi vrijeme koje bi se utrošilo na random binary access

| | | |
|-------|------------------|---------------|
| Row 1 | Partition key 1 | Pos 0 |
| Row 2 | Partition key 11 | Pos 123456789 |
| Row 3 | Partition key 32 | Pos 345678901 |
| Row 4 | Partition key 65 | Pos 567890123 |

ČITANJE NA CASSANDRI : PARTICIJSKI SAŽETAK (SUMMARY)



- › sažetak u memoriji (*off-heap*) partijskog indeksa (npr. svaki 32. ili 64. partijski ključ; konfigurabilna gustoća)
- › sprečava sekvensijalno čitanje (s diska!) datoteke partijskog indeksa

PARTITION INDEX

PARTITION SUMMARY

| | |
|-------------------|--------|
| Partition key 1 | Row 1 |
| Partition key 234 | Row 33 |
| Partition key 678 | Row 65 |
| ... | ... |

Part. key 321?

321>1; 321>234; 321<678;

go to value 234 i.e. row 33

| Row 1 | Partition key 1 | Pos 0 |
|--------|-------------------|---------------|
| Row 2 | Partition key 11 | Pos 123456789 |
| ... | ... | ... |
| Row 33 | Partition key 234 | Pos 678901023 |
| ... | ... | ... |
| Row 65 | Partition key 678 | Pos 890123456 |
| ... | ... | ... |

ČITANJE NA CASSANDRI : KEY CACHE



- › zasebno na razini svake SSTablice; *on-heap*
- › *key cache* sadrži nedavno dohvaćene partijske ključeve s cjelokupnim sadržajem
- › u slučaju pogotka na *cache* (*cache hit*) tj. *cache* sadrži traženi ključ
 - sadržaj se dohvaća iz *cachea*
 - nema potrebe za čitanjem s diska! Preskaču se partijski sažetak, partijski indeks i SSTablica

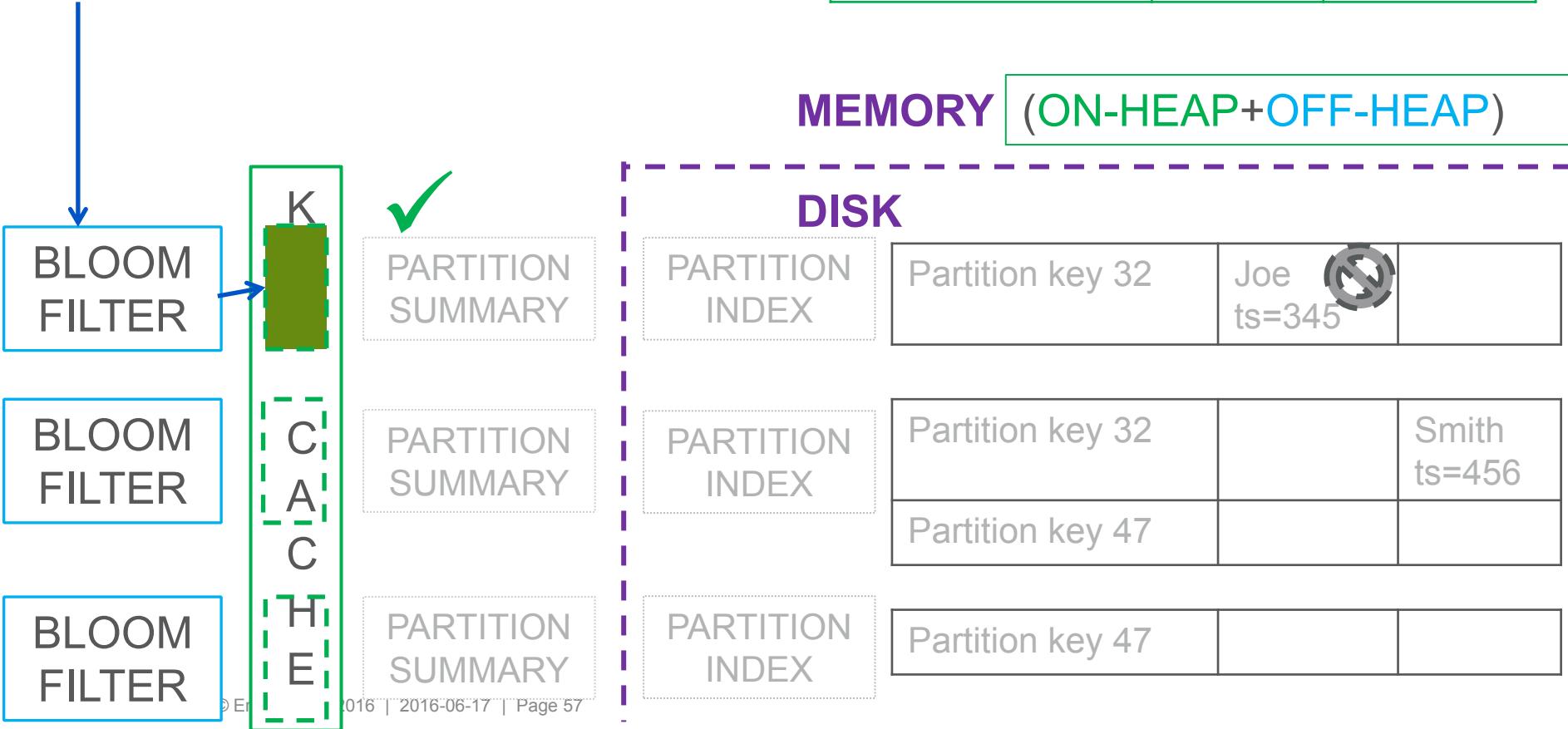
ČITANJE NA CASSANDRI : POGODAK NA KEY CACHEU



SELECT * FROM
PERSON WHERE ID
= 32



| | | |
|------------------|------|-----------------|
| Partition key 32 | ??? | Baker ts=901 |
| Partition key 65 | Tony | Cooper |



ČITANJE NA CASSANDRI : *BLOOM FILTER*



- › vjerojatnosna struktura; zasebno za svaku SSTablicu
- › za dani partijski ključ određuje vjerojatnost da se zapis(i) s tim ključem nalaze spremjeni u toj SSTablici
- › bez lažno negativnih (*false negatives*); mogu postojati lažno pozitivni (*false positives*); tj. postojeći ključ u SSTablici se nikad neće propustiti pročitati, no beskorisno pretraživanje za nepostojeće ključeve koje troši vrijeme može se ponekad dogoditi)
 - tipičan udio lažno pozitivnih između 0.01 i 0.1 (0.01 bolje sprečava beskorisno čitanje, ali troši više memorijskih resursa)

```
PROCEDURE READ (INPUT: partition_key)
read from MemTable (partition_key);
set_latest_timestamp(column_set);
FOR each SSTable {
    read from BloomFilter;
    IF (key_may_exist(partition_key)
        && timestamp_in_SSTable_smaller_than_current_latest(column_set)){
        read from KeyCache();
        IF(key_does_exist(partition_key)){
            set_latest_timestamp(column_set);
            return content(partition_key);
        }
        read from partition summary;
        read from partition index;
        IF(key_does_exist(partition_key)){
            read from disk;
            set_latest_timestamp(column_set);
            return content(partition_key);
        }
    }
}
```

(DEMO 3)



- › uključiti *tracing*
- › ubaciti novi redak
- › dohvatiti dotični redak (iz MemTable); pratiti vrijeme dohvata
- › flush
- › dohvatiti isti redak (ovaj put iz SSTable); pratiti vrijeme dohvata
- › dohvatiti isti redak (ovaj put iz SSTablice); pratiti vrijeme dohvata



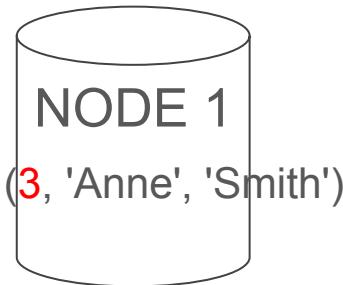
MODELIRANJE PODATAKA NA CASSANDRI

PARTICIJSKI KLJUČ I PRIMARNI KLJUČ

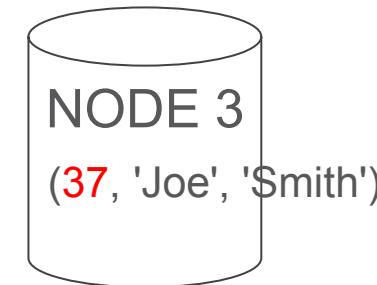
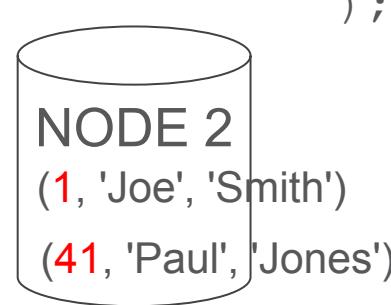


- › partitioniranje po primarnom ključu (jednostupčanom)

```
CREATE TABLE PERSON (
    id int PRIMARY KEY,
    first_name varchar,
    last_name varchar
);
```



```
CREATE TABLE PERSON (
    id int,
    first_name varchar,
    last_name varchar,
    PRIMARY KEY (id)
);
```

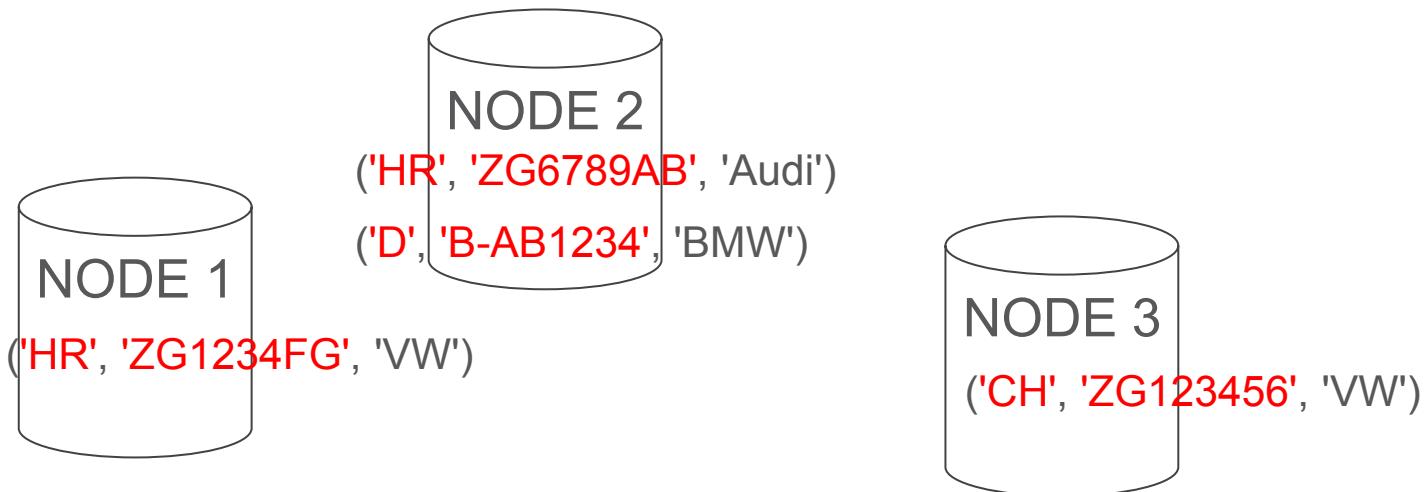


PARTICIJSKI KLJUČ I PRIMARNI KLJUČ



- › partitioniranje po primarnom ključu (višestupčanom)

```
CREATE TABLE VEHICLE (
    plate varchar,
    state varchar,
    manufacturer varchar,
    PRIMARY KEY (plate, state)
);
```

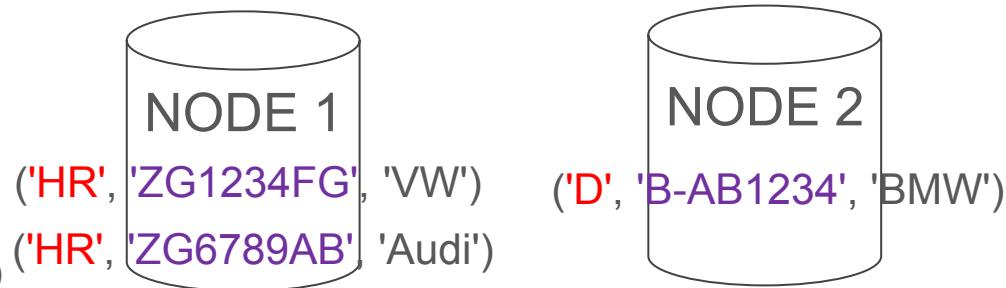


PARTICIJSKI KLJUČ I PRIMARNI KLJUČ



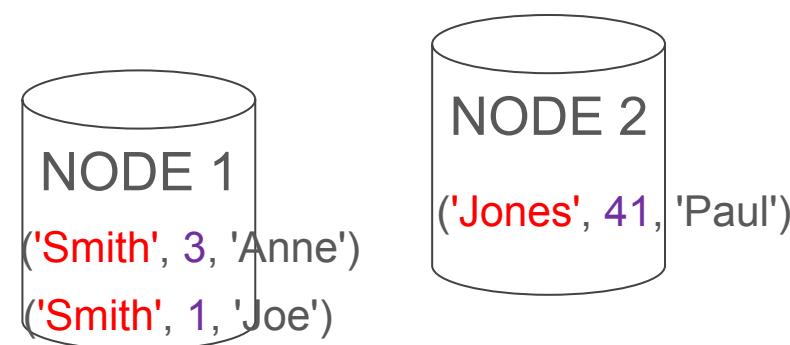
- › particija samo po dijelu višestupčanog primarnog ključa?

```
CREATE TABLE VEHICLE (
    plate varchar,
    state varchar,
    manufacturer varchar,
    PRIMARY KEY ((state), ?)
);
```



- › particija po stupcu koji nije dio primarnog ključa

```
CREATE TABLE PERSON (
    id int,
    first_name varchar,
    last_name varchar,
    PRIMARY KEY ((last_name), ?)
);
```

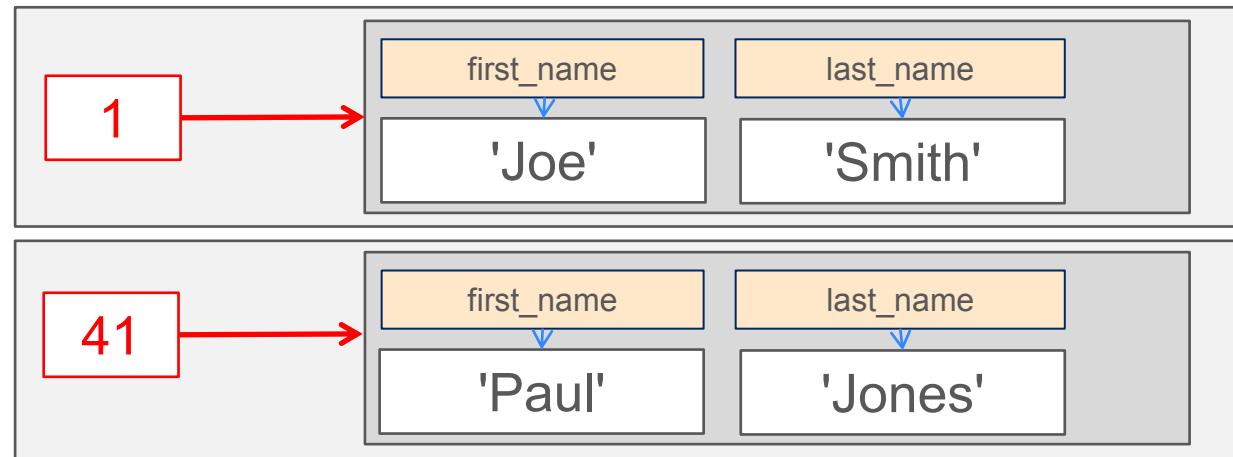


PARTICIJSKI KLJUČ I SKUP STUPACA



```
CREATE TABLE PERSON (
    id int PRIMARY KEY,
    first_name varchar,
    last_name varchar
);
```

- › partijski ključ = primarni ključ → jedan redak po partijski

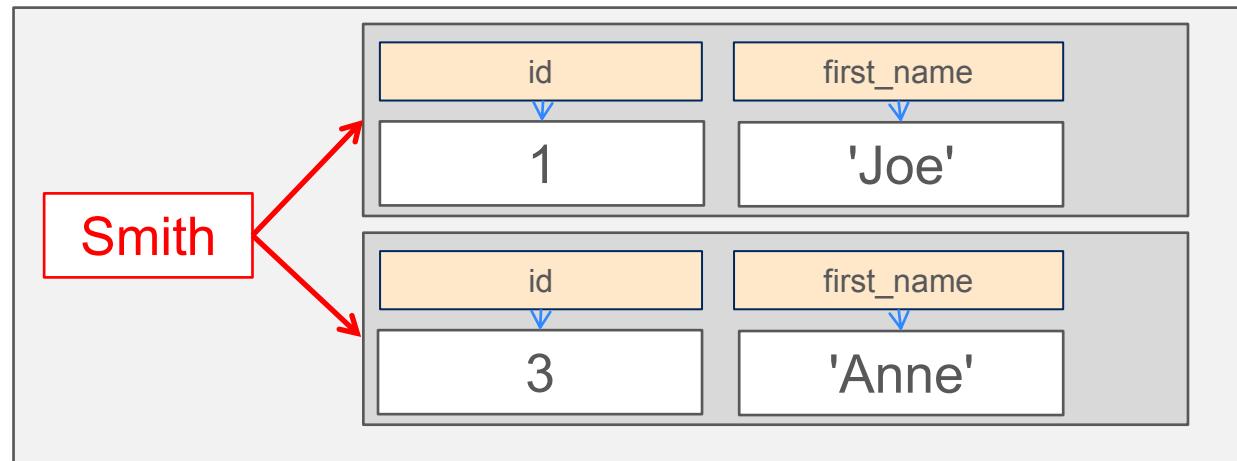


PARTICIJSKI KLJUČ I SKUP STUPACA



```
CREATE TABLE PERSON (
    id int,
    first_name varchar,
    last_name varchar,
    PRIMARY KEY ( (last_name) , ?)
);
```

particijski ključ ≠ primarni
ključ → moguće više redaka
po jednoj particiji



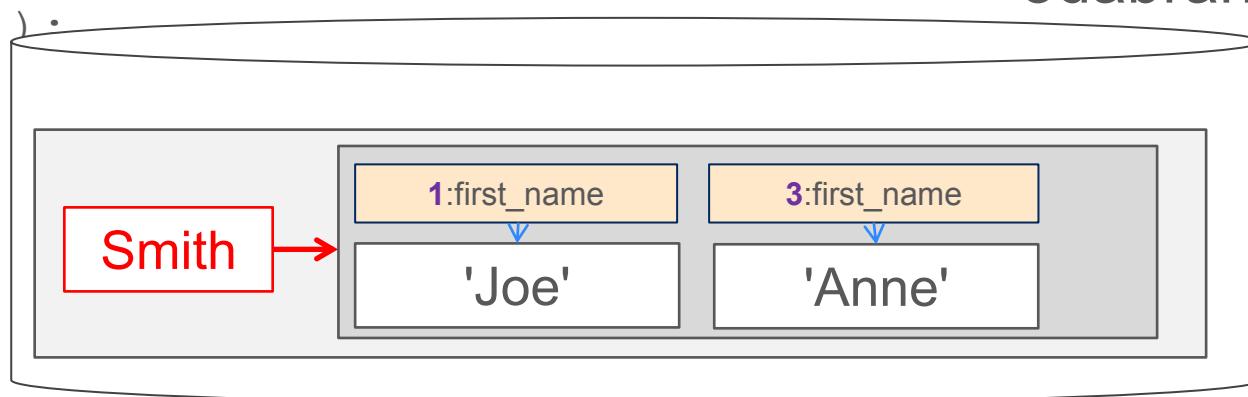
› problemi:

- kako razlikovati retke (istoimeni stupci u različitim recima)
- kako spremiti više redaka da se mogu efikasno čitati

GRUPNI KLJUČ



```
CREATE TABLE PERSON (
    id int,
    first_name varchar,
    last_name varchar,
    PRIMARY KEY ((last_name), ?)
).
```



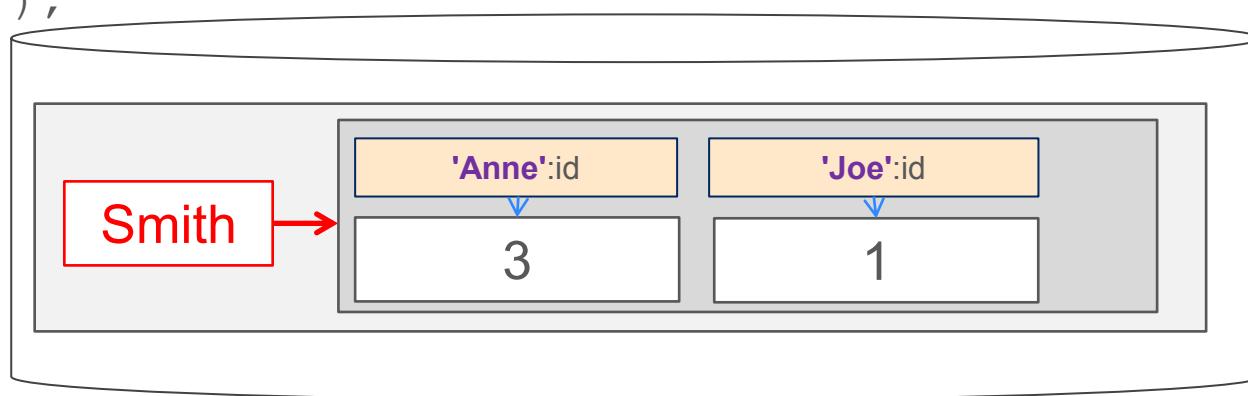
ODGOVOR: organizirati višestruke redove unutar particije kao INDEKS (po odabranim stupcima)

- › reci se indeksiraju po **GRUPNOM KLJUČU (CLUSTERING KEY)** (ovdje: **id**)
- › SSTablica fizički spremi retke poredane po grupnom ključu
- › ono što percipiramo kao vrijednosti stupaca u grupnom ključu (*id*) postaje dio imena preostalih stupaca tj. metapodatak

GRUPNI KLJUČ



```
CREATE TABLE PERSON (
    id int,
    first_name varchar,
    last_name varchar
    PRIMARY KEY (last_name, ?)
);
```



ALTERNATIVNO: bismo li mogli grupirati po stupcu first_name?

- › nije osigurana jedinstvenost: `INSERT INTO PERSON VALUES (5, 'Joe', 'Smith')` uzrokuje dvoznačnost
- › rješenje: grupni ključ mora osigurati jedinstvenost
 - GK (id) OK;
 - GK (id, first_name) OK, ali redundantno

GK (first_name) NIJE OK;

GRUPNI KLJUČ



```
CREATE TABLE PERSON (
    id int,
    first_name varchar,
    last_name varchar
    PRIMARY KEY (last_name), ?
) ;
```

- › rješenje: grupni ključ mora osigurati jedinstvenost
 - PART. KEY (last_name); CLUST KEY (id) ✓
 - › za zadani *last_name*, zapisi poredani po *id*
 - PART. KEY (last_name); CLUST KEY (first_name) ✗
 - › nema jedinstvenosti
 - PART. KEY (last_name); CLUST KEY (id, first_name) ?
 - › *id* osigurava jedinstvenost; *first_name* je redundantan
 - PART. KEY (last_name); CLUST KEY (first_name, id) OK ✓
 - › ispravno rješenje za zadatak se prethodne folije
 - › za zadani *last_name*, zapisi se spremaju poredani po *first_name*, a za istu vrijednost *first_name*, poredani po *id*

PRIMARNI KLJUČ



- › PRIMARY KEY = PARTICIJSKI KLJUČ + GRUPNI KLJUČ
 - mora osigurati jedinstvenost
- › partijski ključ
 - određuje čvorove sustava na kojima se spremaju replike (primjena *hash* funkcije nad part. ključem predstavljenim kao niz bajtova)
 - može osigurati jedinstvenost, ali i ne mora
- › grupni ključ
 - postoji u slučaju kad partijski ključ ne osigurava jedinstvenost
 - mora osigurati jedinstvenost
 - omogućuje ORDER BY prilikom pisanja tj. bez sortiranja prilikom čitanja

PRIMARNI KLJUČ: SINTAKSA NAREDBE *CREATE TABLE*



› PRIMARY KEY

((part_k_1, ..., part_k_M), clust_k_1, ..., clust_k_N)

- › unutarnje zagrade razdvajaju partijski ključ od grupnog ključa
 - mogu se izostaviti kad je partijski ključ jednostupčan

PRIMARNI KLJUČ: SINTAKSA NAREDBE *CREATE TABLE*



- › PRIMARY KEY (partition_key_column)
 - jednostupčani particijski ključ, bez grupnog ključa
- › PRIMARY KEY ((partition_key_col1, ..., partition_key_colN))
 - višestupčani particijski ključ, bez grupnog ključa
- › PRIMARY KEY (partition_key_column, clustering_column1, ..., clustering_columnM)
 - jednostupčani particijski ključ i grupni ključ (jedan par zagrada!)
- › PRIMARY KEY ((partition_key_col1, ..., partition_key_colN), clustering_column1, ..., clustering_columnM)
 - višestupčani particijski ključ i grupni ključ

UPsert



› insert

– RDBMS:

- › provjeri postojeće primarne ključeve (koristeći indeks tj. s diska; po definiciji, primarni ključevi na RSUBP jesu indeksirani)
- › zapiši ako ne postoji redak sa zadanim ključem

– Cassandra

- › operacija pisanja zapisuje u *commitlog* i *MemTable*; izostavlja se bilo kakvo čitanje diska: mnogo brže čitanje
- › zapisi se upisuju bez provjere postojanja zapisa s istom vrijednošću primarnog ključa
- › izvršit će se prebrisivanje novih vrijednosti preko postojećih zapisa: **INSERT može postati UPDATE**



UPsert

Cassandra

```
CREATE TABLE person (
    id int,
    first_name varchar,
    last_name varchar,
    PRIMARY KEY (id)
);

INSERT INTO person (id,
first_name, last_name)
VALUES (23, 'LeBron',
'James'); ✓

INSERT INTO person (id,
first_name, last_name)
VALUES (23, 'Michael',
'Jordan'); ✓
```

RDBMS

```
CREATE TABLE person (
    id integer,
    first_name varchar(64),
    last_name varchar(64),
    PRIMARY KEY (id)
);

INSERT INTO person (id,
first_name, last_name)
VALUES (23, 'LeBron',
'James'); ✓

INSERT INTO person (id,
first_name, last_name)
VALUES (23, 'Michael',
'Jordan'); ✗
```

| | | |
|----|---------|--------|
| 23 | Michael | Jordan |
|----|---------|--------|

| | | |
|----|--------|-------|
| 23 | LeBron | James |
|----|--------|-------|



UPsert

Cassandra

```
CREATE TABLE person (
    id int,
    first_name varchar,
    last_name varchar,
    PRIMARY KEY (id)
);
-- EMPTY TABLE
```

```
UPDATE person
SET
first_name='Michael',
last_name='Jordan'
WHERE id=23; ✓
```

| | | |
|----|---------|--------|
| 23 | Michael | Jordan |
|----|---------|--------|

- › naredba UPDATE na Cassandri MORA sadržavati cijeli primarni ključ u svojem WHERE dijelu
 - odnosit će se na jedan redak
 - naredbom UPDATE ubacit će se i nepostojeći redak
 - **UPDATE može postati INSERT**
- › **UPDATE = INSERT → „UPsert”**
- › uvijek idempotentno (osim kod brojača, *counters*, koji su izvan dosega gradiva)

SELECT



- › efikasni su upiti koji se odnose na samo JEDNU PARTICIJU (odnosno najviše nekoliko particija)
- › SELECT * FROM table_name je sintaksno ispravna naredba, ali prolazi kroz sve particije tj. razne čvorove → dugotrajno izvođenje
- › ograničenja definirana pomoću WHERE su sintaksno korektna samo za efikasne upite



SELECT - WHERE

```
CREATE TABLE person (
    id int,
    first_name varchar,
    last_name varchar,
    PRIMARY KEY ((last_name), first_name, id)
);
```

- › particija po last_name; grupiranje (order by) po first_name, potom po id

```
SELECT * person WHERE last_name = 'Smith'; ✓
```

- › jedna particija, više redaka

```
SELECT * person WHERE last_name IN ('Smith', 'Jones'); ✓
```

- › više particija (više redaka svaka)



SELECT - WHERE

```
CREATE TABLE person (
    id int,
    first_name varchar,
    last_name varchar,
    PRIMARY KEY ((last_name), first_name, id)
);
```

```
SELECT * person WHERE last_name = 'Smith' AND
first_name = 'Tom'; ✓
```

```
SELECT * person WHERE last_name = 'Smith' AND
first_name > 'Tom'; ✓
```

› poređak grupiranja = fizički redoslijed u SSTablicama

```
SELECT * person WHERE last_name = 'Smith' AND
first_name IN ('Joe', 'Tom'); ✓
```



SELECT - WHERE

```
CREATE TABLE person (
    id int,
    first_name varchar,
    last_name varchar,
    PRIMARY KEY ((last_name), first_name, id)
);
```

```
SELECT * person WHERE first_name = 'Tom'; ✗
```

› bez specificirane particije

```
SELECT * person WHERE last_name = 'Smith' AND
first_name = 'Tom' AND id=4; ✓
```

```
SELECT * person WHERE last_name = 'Smith' AND id=4; ✗
```

› „desnije” kolone grupnog ključa ne mogu se koristiti ako nedostaju „ljevije”

SELECT – SEKUNDARNI INDEKS



```
CREATE TABLE athlete (
    id int,
    first_name varchar,
    last_name varchar,
    sport varchar,
    PRIMARY KEY ((last_name), first_name, id)
);
```

SELECT * person WHERE sport = 'basketball'; ✗

```
CREATE INDEX idx_name ON athlete(sport)
```

SELECT * person WHERE sport = 'basketball'; ✓

- › zasebna indeksna struktura gradi se na svakom čvoru
- › indeksi čine pisanje manje efikasnim (reorganizacija)
- › treba ih izbjegavati koliko god je moguće

SELECT – ORDER BY



```
CREATE TABLE person (
    id int,
    first_name varchar,
    last_name varchar,
    PRIMARY KEY ((last_name), first_name, id)
);
```

- › grupni ključ uvijek implicitno nameće poredak
- › prepostavljeni (*default*) poredak (i, implicitno, njemu obrnuti poredak) može se zadati eksplicitno

```
SELECT * person WHERE last_name = 'Smith' ORDER BY
first_name; ✓ (prepostavljeni poredak)
```

```
SELECT * person WHERE last_name = 'Smith' ORDER BY
first_name, id; ✓ (prepostavljeni poredak)
```

SELECT – ORDER BY



```
CREATE TABLE person (
    id int,
    first_name varchar,
    last_name varchar,
    PRIMARY KEY ((last_name), first_name, id)
);
```

SELECT * person WHERE last_name = 'Smith' ORDER BY first_name DESC; ✓ (obrnuti poredak)

SELECT * person WHERE last_name = 'Smith' ORDER BY first_name DESC, id DESC; ✓ (obrnuti poredak)

SELECT * person WHERE last_name = 'Smith' ORDER BY first_name DESC, id; ✗ (trebalo bi dodatno sortiranje)

SELECT * person WHERE last_name = 'Smith' ORDER BY first_name, id DESC; ✗ (trebalo bi dodatno sortiranje)

TIPOVI PODATAKA



| C* type | | C* type | |
|---------|---|-----------|--|
| bigint | 64-bitni Long s predznakom | map | niz riječi u stilu JSON-a : { riječ : riječ, riječ : riječ ... } |
| blob | niz bajtova (bez validacije) | set | skup od jednog ili više elemenata bez poretku |
| boolean | true / false | text | string enkodiran kao UTF-8 |
| counter | raspodijeljeni brojač (64-bitni Long) | timestamp | date plus time, encoded as 8 bytes since epoch |
| decimal | decim. broj varijabilne preciz. | uuid | UUID standardnog formata |
| double | 64-bit IEEE-754 floating point | timeuuid | UUID isključivo tipa 1 |
| float | 32-bit IEEE-754 floating point | varchar | (=text) |
| inet | niz znakova koji opisuje IP adresu, u IPv4 ili IPv6 formatu | varint | cijeli broj s preciznošću po volji |
| list | skup od jednog ili više elemenata s poretkom | | |

OBLIKOVANJE BAZE POD. U SUSTAVU CASSANDRA



```
CREATE TABLE person (
    id int,
    first_name varchar,
    last_name varchar,
    PRIMARY KEY ((last_name), first_name, id));
```

```
SELECT * person WHERE last_name = 'Smith'; ✓
```

```
SELECT * person WHERE first_name = 'Joe'; ✗
```

- › dva gornja upita ne mogu se izvesti nad samo jednom tablicom u Cassandri (osim ako se ne napravi sek. indeks)
- › oba bi upita radila nad jednom tablicom u RSUBP s kolonama `last_name, first_name` bez obzira na primarni ključ

OBЛИKOVANJE BAZE POD. U SUSTAVU CASSANDRA



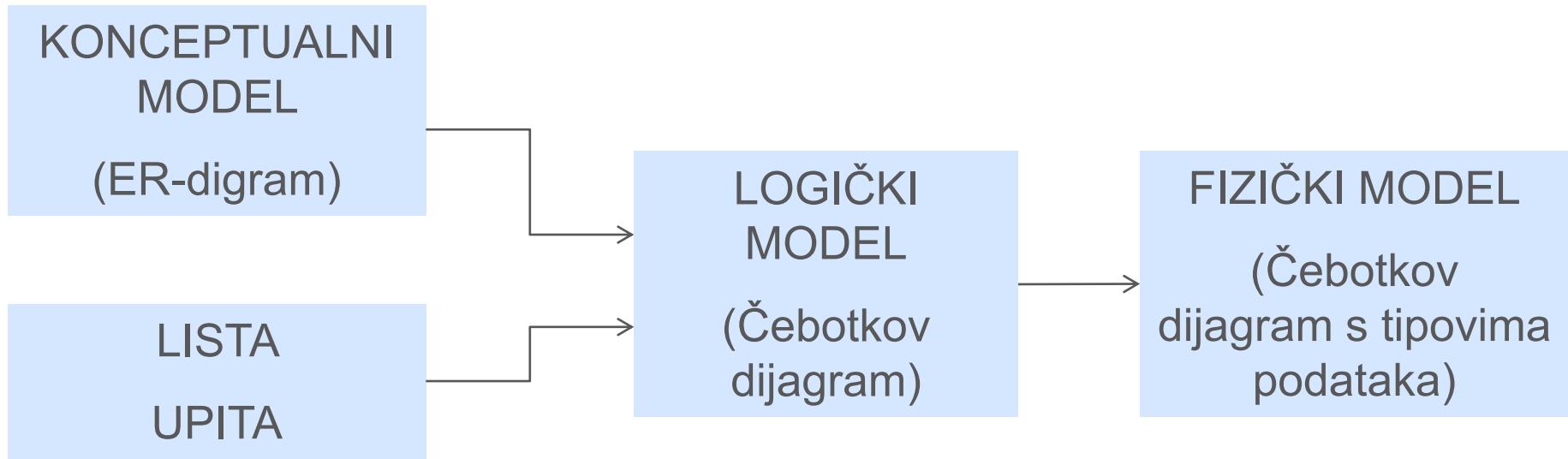
› SPAJANJA zabranjena

- no denormalizirana tablica u Cassandri može u sebi sadržavati više tablica iz relacijskog modela
- čitanje nekoliko redaka iz jedne denormalizirane tablice bolje nego čitanje više cjelokupnih tablica, pa potom spajanje na klijentskoj strani (*client-side join*)

› svaka tablica u Cassandri služi JEDNOM UPITU

- dovodi do udvostručavanja (*duplication*) podataka
- uz pažljiv odabir partijskog i grupnog ključa, tablica može služiti i za više različitih upita sa sličnim WHERE dijelom

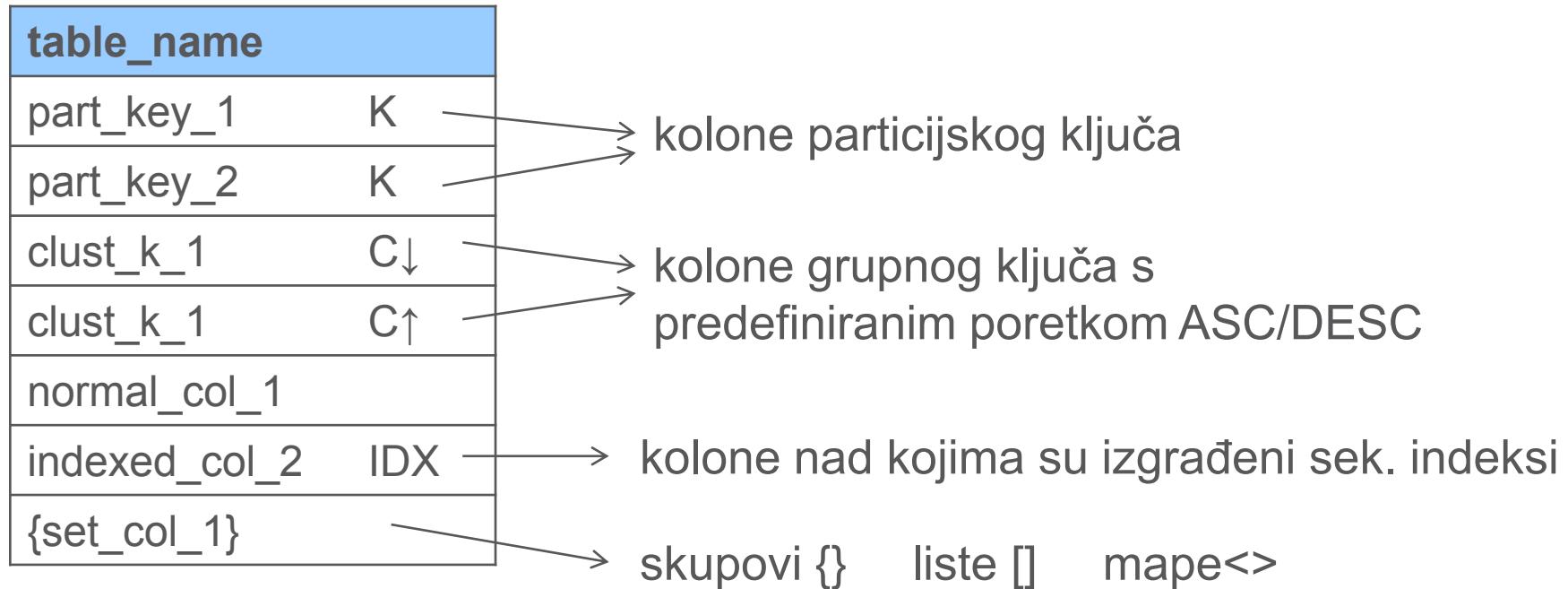
OBЛИKOVANJE BAZE POD. U SUSTAVU CASSANDRA



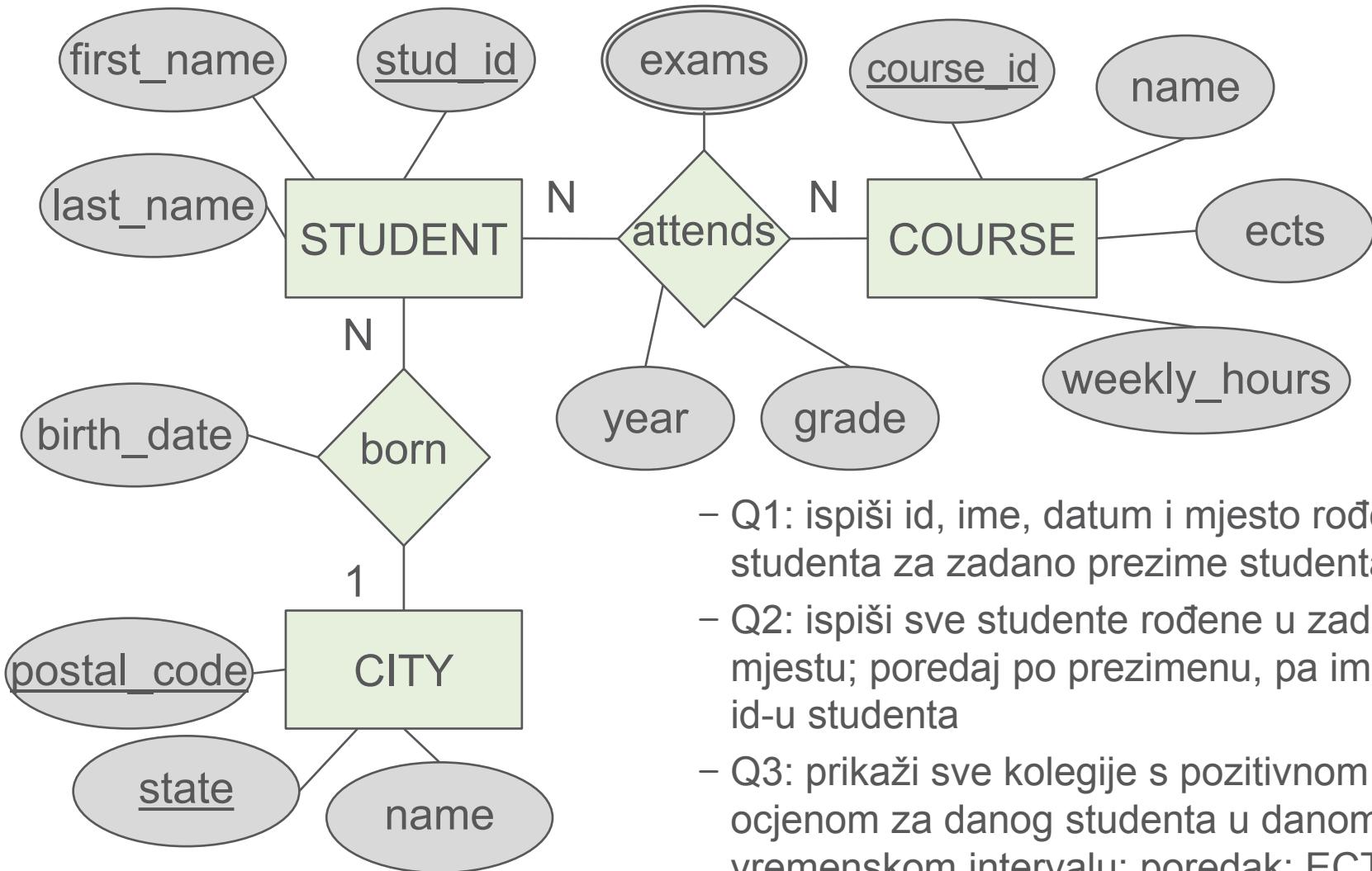
ČEBOTKOV DIJAGRAM



logički model



OBLIKOVANJE BP U CASSANDRI - PRIMJER



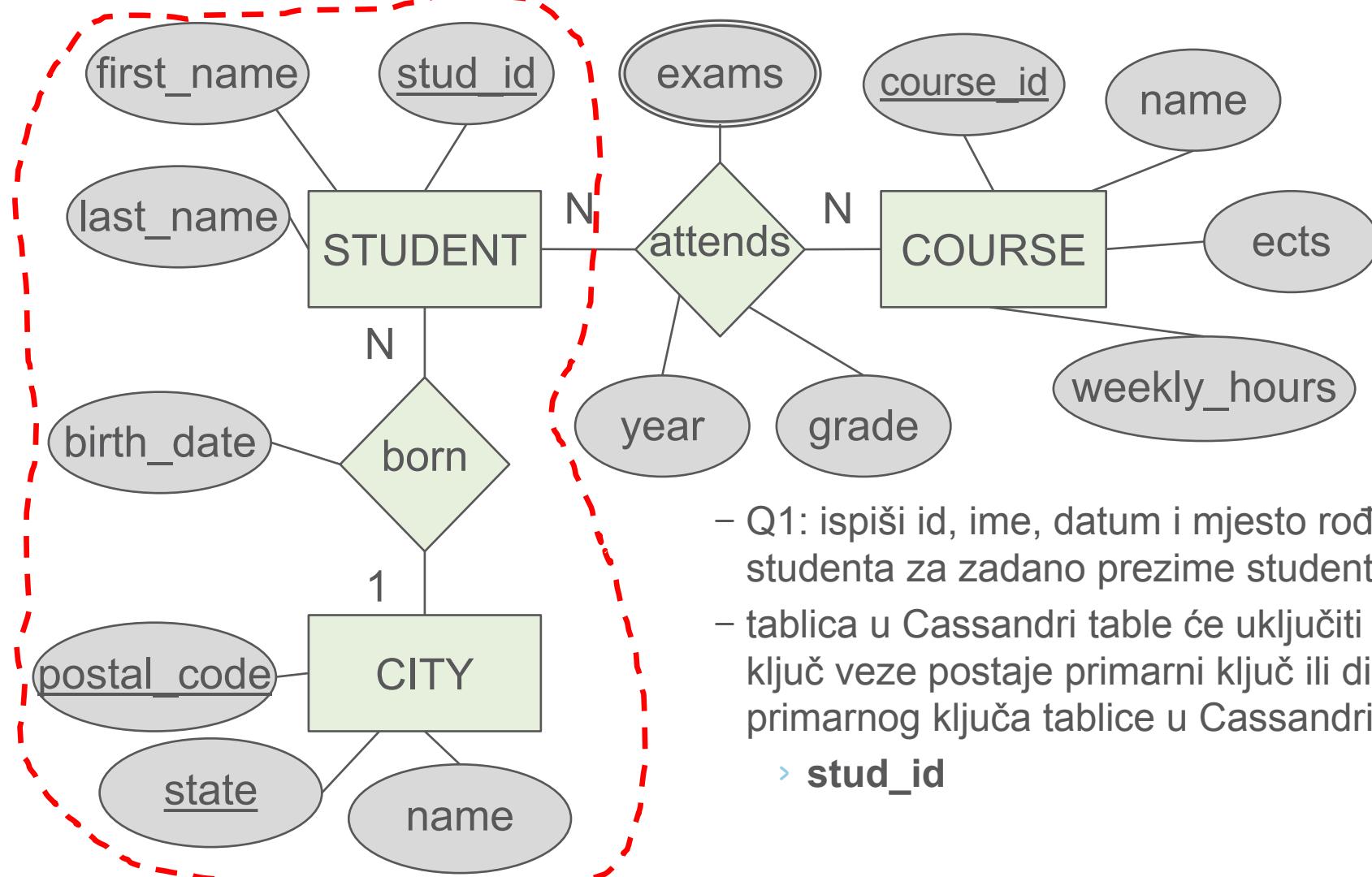
- Q1: ispiši id, ime, datum i mjesto rođenja studenta za zadano prezime studenta
- Q2: ispiši sve studente rođene u zadanom mjestu; poredaj po prezimenu, pa imenu, pa id-u studenta
- Q3: prikaži sve kolegije s pozitivnom ocjenom za danog studenta u danom vremenskom intervalu; poredak: ECTS DESC, potom ime kolegija ASC

OSNOVNA NAČELA OBЛИKOVANJA

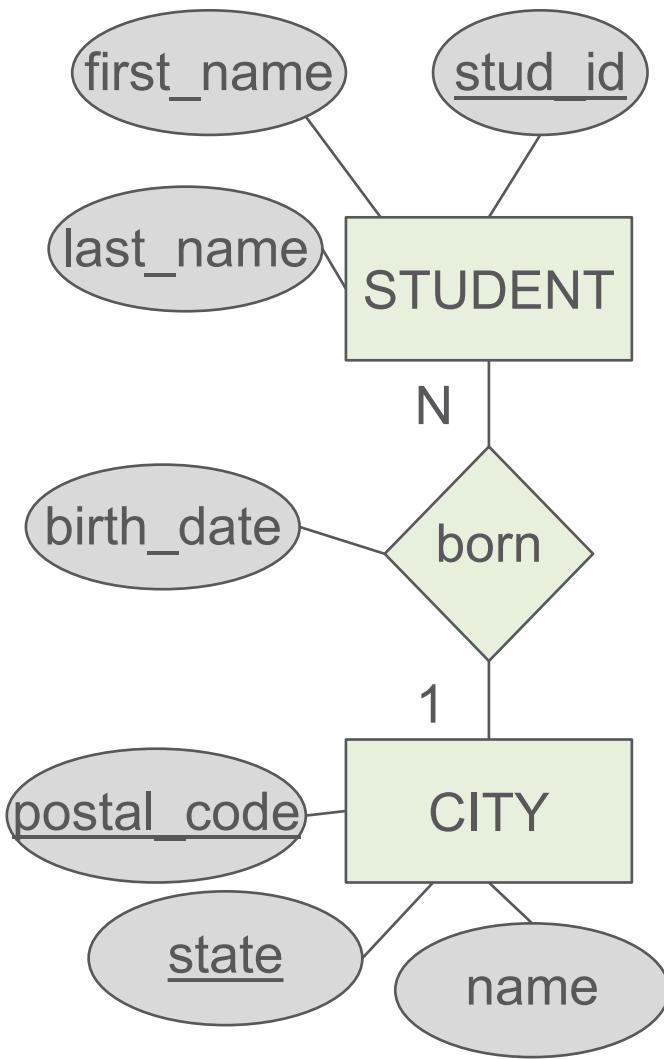


- › SELECT... WHERE column_name_1=?
 - column_name_1 → partijski ključ
- › ...WHERE column_name_1=? AND column_name_2=?
 - › column_name_1, column_name_2 → komponente part. k.
 - › jedna kolona → partijski ključ; druga kolona → grupni ključ
- › ...WHERE column_name_1=? AND column_name_2>?
 - column_name_1 → partijski ključ; column_name_2 → grupni ključ
- › ne zaboraviti osigurati jedinstvenost
 - dodati još uvijek nedostajuće komponente primarnog ključa u konceptualnom modelu kao „najdesnije“ komponente grupnog ključa tablice u Cassandri

OBLIKOVANJE BP U CASSANDRI - PRIMJER



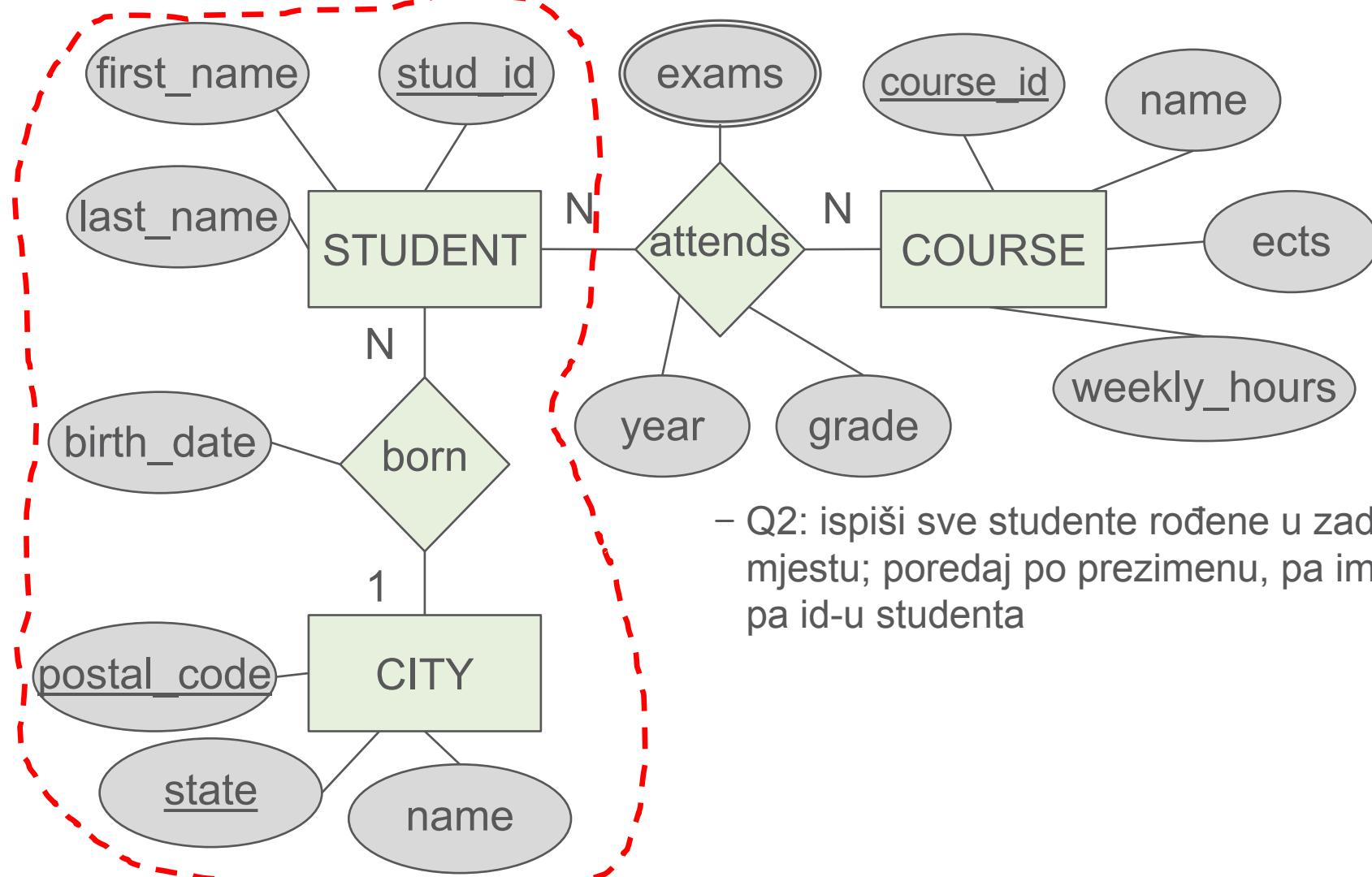
OBLIKOVANJE BP U CASSANDRI - PRIMJER



- Q1: ispiši id, ime, datum i mjesto rođenja studenta za zadano prezime studenta
- prim. klj. konc. mod.: **stud_id**
- ... *za zadano prezime studenta* → K
- bez uvjeta pretraživanja s nejednakošću; bez poretnika
- zadnje: dodati `stud_id` kao grupni ključ

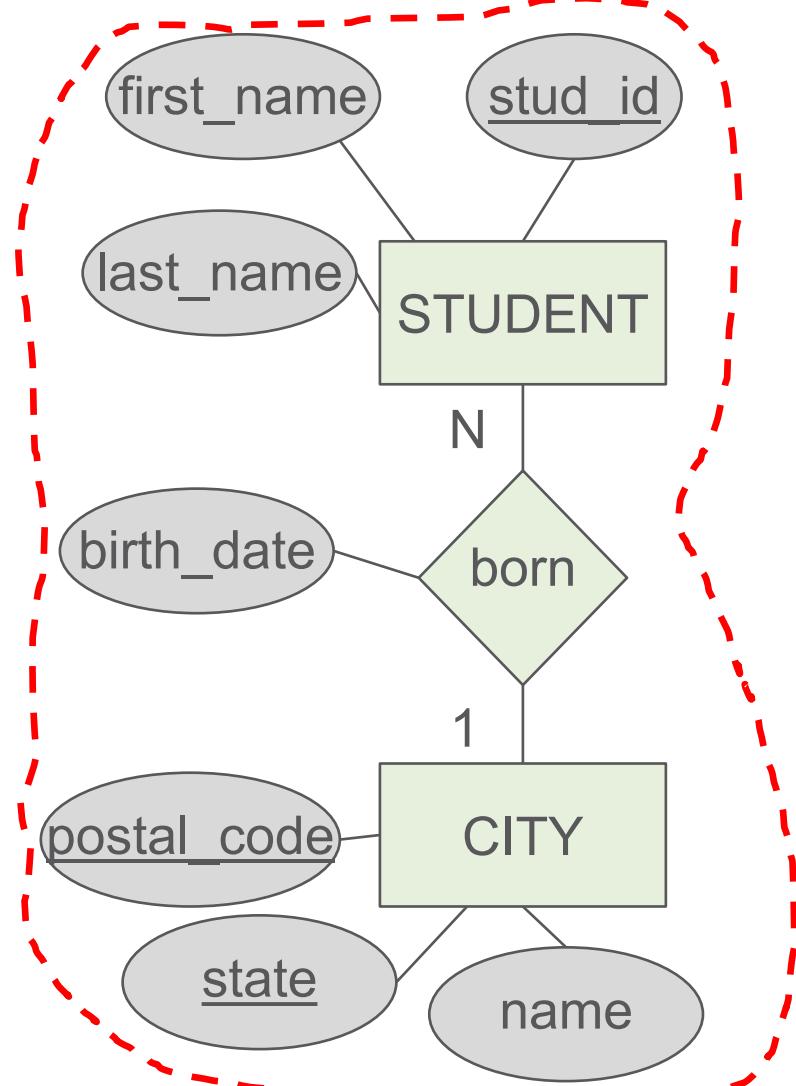
| students_by_last_name | |
|-----------------------|----|
| last_name | K |
| stud_id | C↑ |
| first_name | |
| birth_date | |
| postal_code | |
| state | |
| city_name | |

OBLIKOVANJE BP U CASSANDRI - PRIMJER



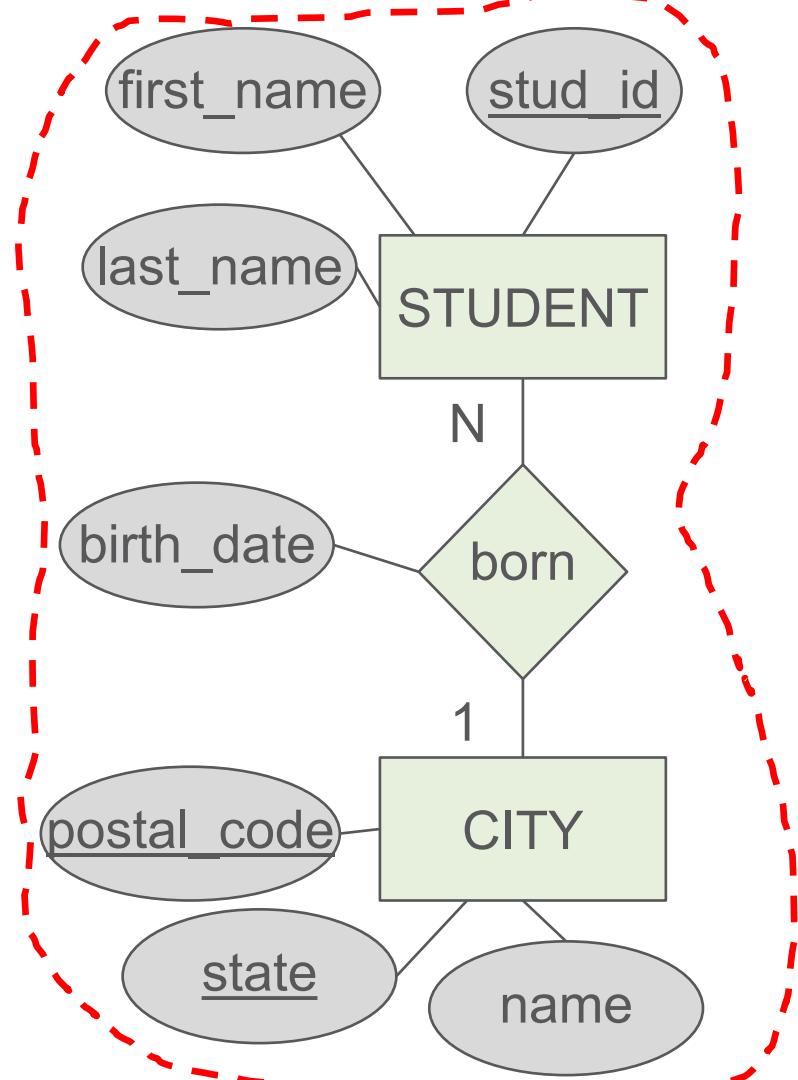
– Q2: ispiši sve studente rođene u zadanim mjestima; poredaj po prezimenu, pa imenu, pa id-u studenta

OBЛИKOVANJE BP U CASSANDRI - PRIMJER



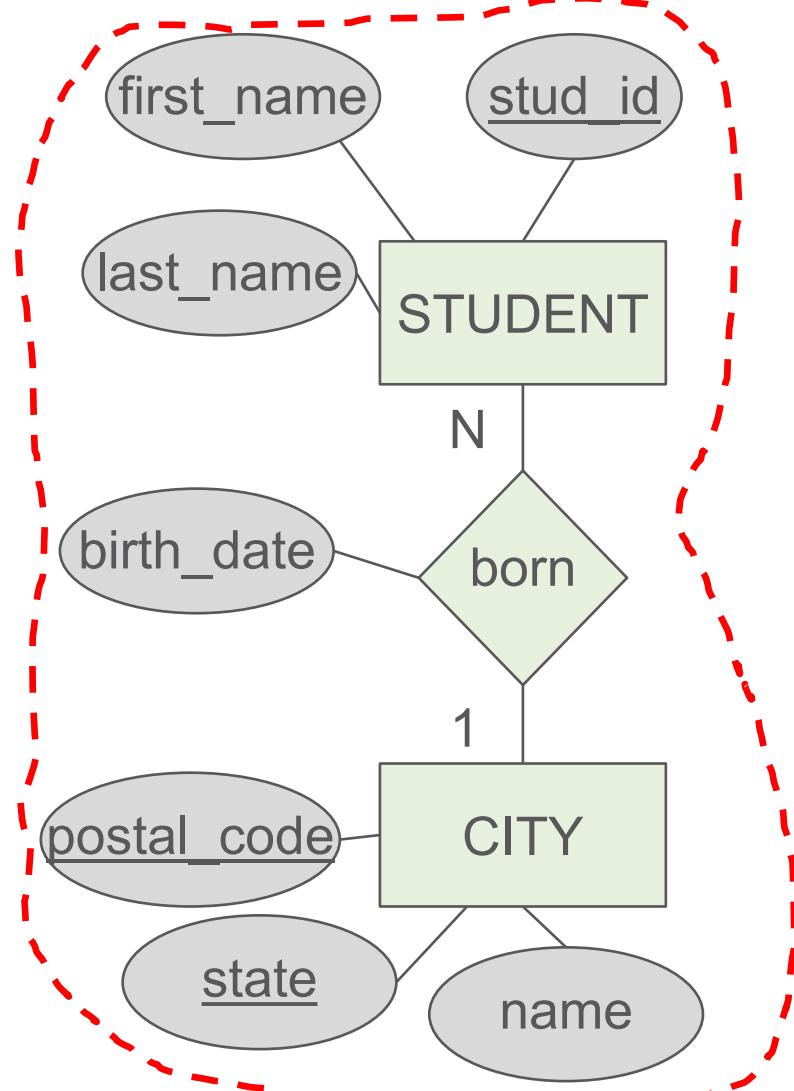
- Q2: ispiši sve studente rođene u zadanom mjestu; poredaj po prezimenu, pa imenu, pa id-u studenta
- ...**rođene u zadanom mjestu** → mjesto treba biti K, no koja točno kolona?
 - > prim. klj. za entitet CITY je u redu
 - > možda bi se pretraživalo po imenu mesta
 - > moramo pitati korisnika na što je točno mislio!
 - > bez tog odgovora dvije su mogućnosti
 - name K, postal_code K/C, state K/C
 - postal_code K, state K/C (bez mogućnosti pretraživanja po *name*)

OBLIKOVANJE BP U CASSANDRI - PRIMJER



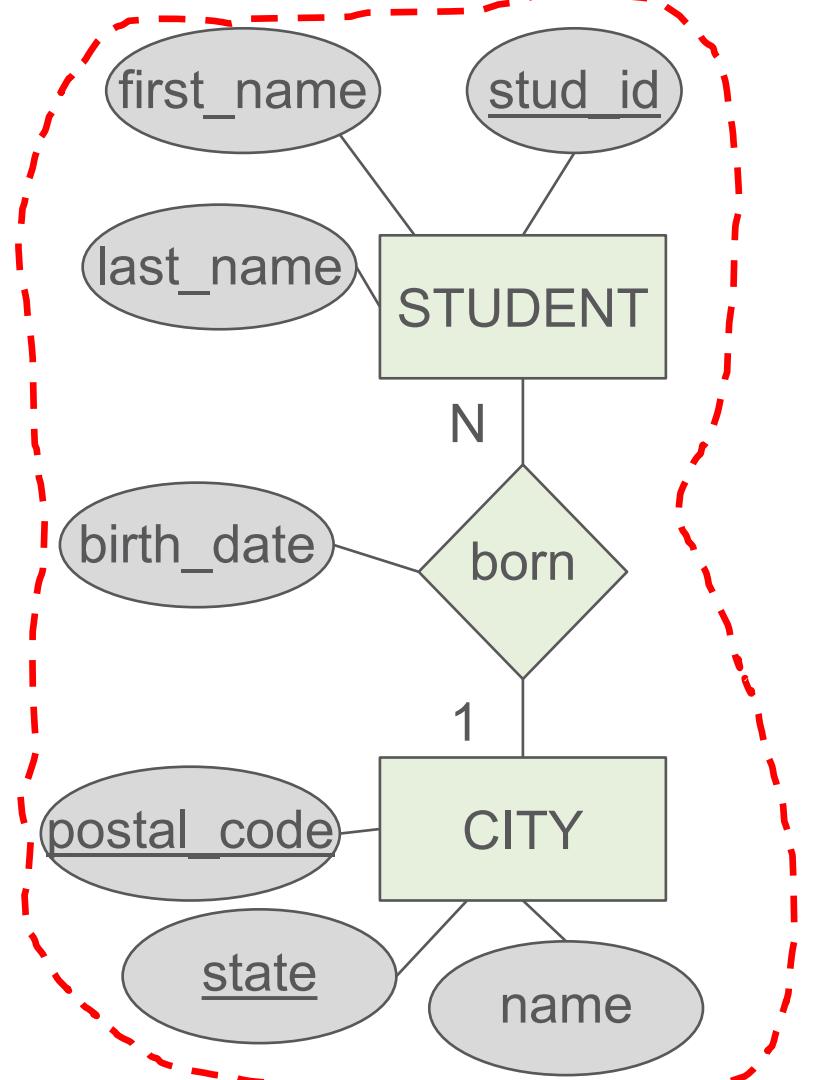
- Q2: ispiši sve studente rođene u zadanom mjestu; poredaj po prezimenu, pa imenu, pa id-u studenta
- ... **rođene u zadanom mjestu** → uz pretpostavku da je **name K, postal_code K/C, state K/C** ispravno, bi li *postal_code*, *state* bili K ili C?
 - > name K, postal_code C, state C
 - WHERE name=? ✓
 - WHERE name=? AND postal_code=? ✓
 - WHERE name=? AND postal_code=?
AND state=? ✓
 - > name K, postal_code K, state K
 - WHERE name=? ✗
 - WHERE name=? AND postal_code=? ✗
 - WHERE name=? AND postal_code=?
AND state=? ✓
- NO particije su manje!

OBLIKOVANJE BP U CASSANDRI - PRIMJER



- Q2: ispiši sve studente rođene u zadanom mjestu; poredaj po prezimenu, pa imenu, pa id-u studenta
- ... poredaj po prezimenu, pa imenu, pa id-u studenta
 - > `last_name` C↑, `first_name` C↑, `student_id` C↑
 - > ako je `postal_code` C, `state` C trebaju li doći „desnije“ u grupnom ključu?
 - > Ovisi o tome koji upit želimo omogućiti:
 - WHERE `name=? AND last_name=?`
 - WHERE `name=? AND postal_code=? AND state=? AND last_name=?`
- primarni ključ. veze `stud_id` već je C

OBLIKOVANJE BP U CASSANDRI - PRIMJER

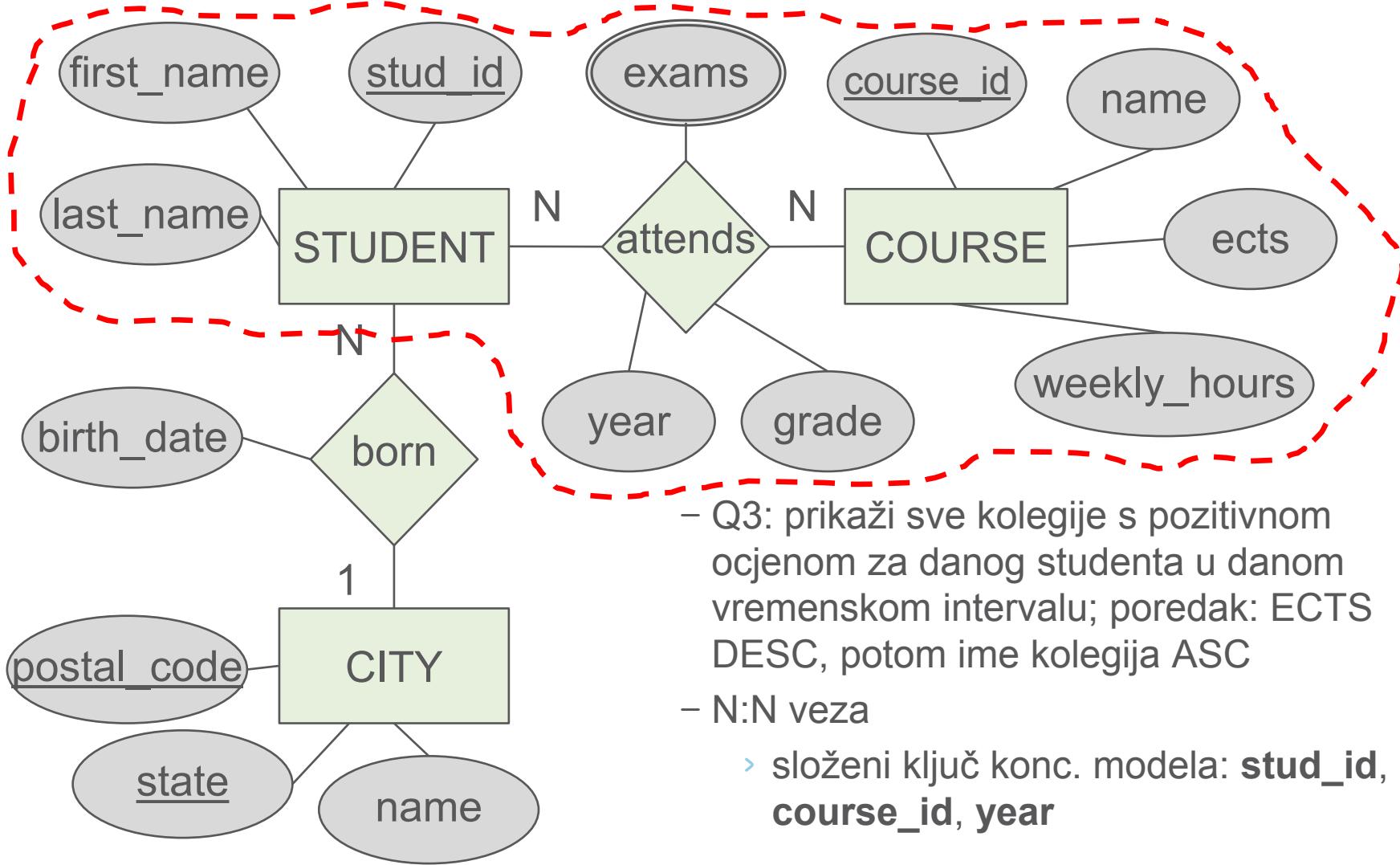


- Q2: ispiši sve studente rođene u zadanom gradu; poredaj po prezimenu, pa imenu, pa id-u studenta
- dva moguća rješenja (lijevo bolje)

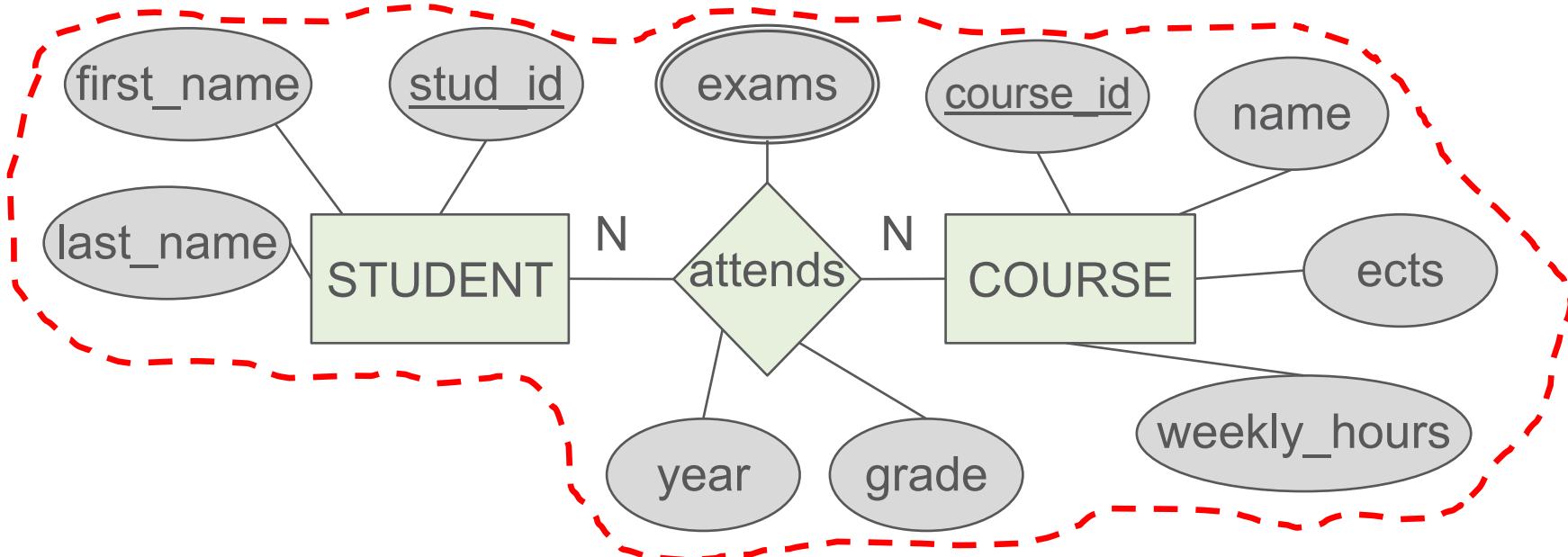
| students_by_city | |
|------------------|----|
| city_name | K |
| last_name | C↑ |
| first_name | C↑ |
| student_id | C↑ |
| postal_code | C↑ |
| state | C↑ |
| birth_date | |

| students_by_city | |
|------------------|----|
| city_name | K |
| postal_code | K |
| state | K |
| last_name | C↑ |
| first_name | C↑ |
| student_id | C↑ |
| birth_date | |

OBLIKOVANJE BP U CASSANDRI - PRIMJER

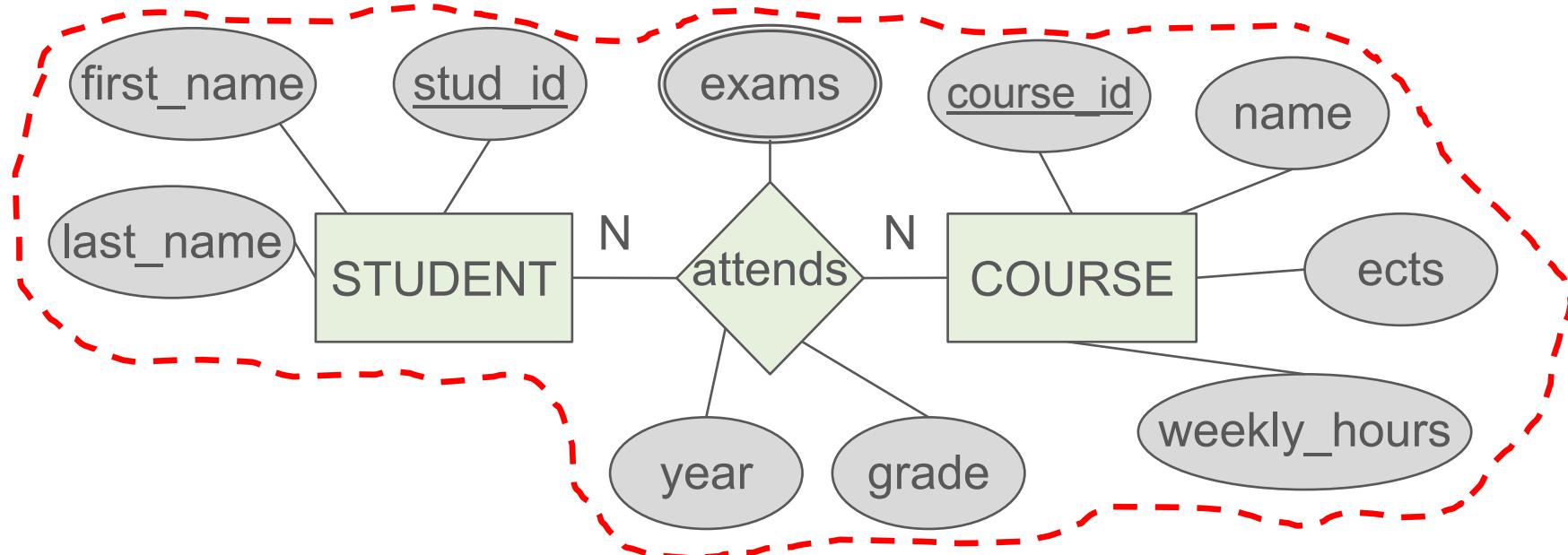


OBLIKOVANJE BP U CASSANDRI - PRIMJER



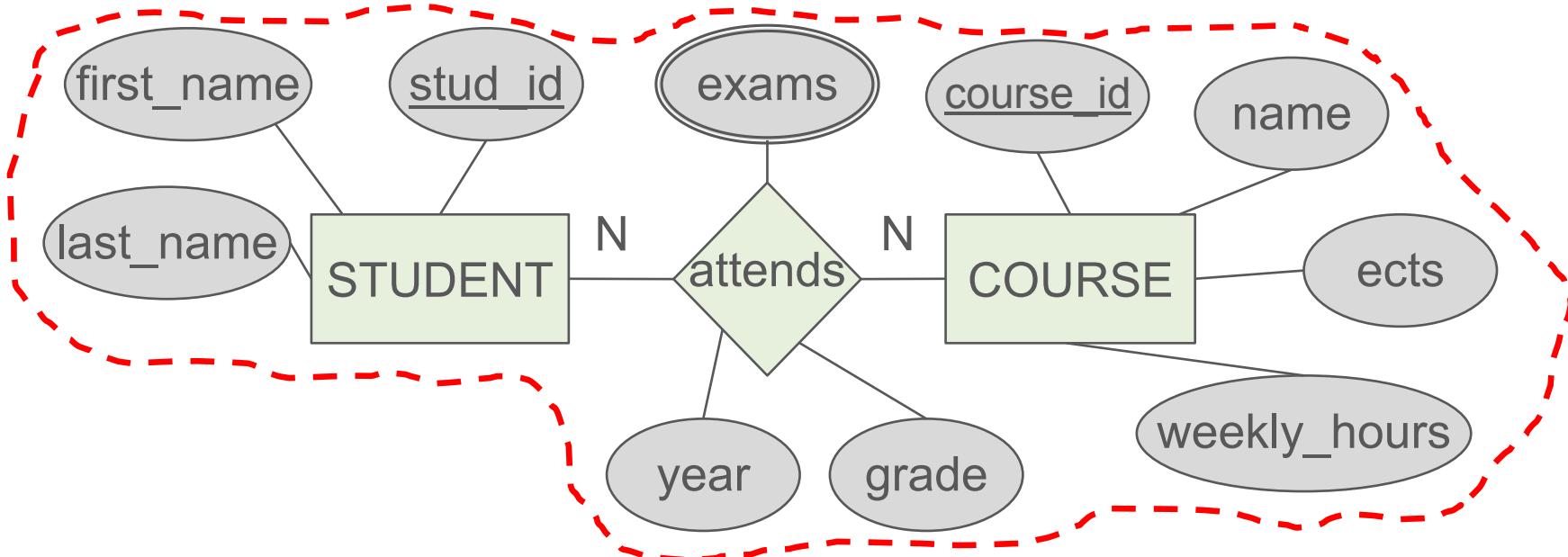
- Q3: prikaži sve kolegije s pozitivnom ocjenom za danog studenta u danom vremenskom intervalu; poredak: ECTS DESC, potom ime kolegija ASC
- složeni ključ konc. modela : **stud_id, course_id, year**
- ... za danog studenta
 - > stud_id=? (K)
- ... za zadani vremenski interval
 - > year>? AND year <?
 - > pretraživanje po uvjetu nejednakosti → year C (asc/desc)

OBЛИКОВАЊЕ BP U CASSANDRI - PRIMJER



- Q3: prikaži sve kolegije s pozitivnom ocjenom za danog studenta u danom vremenskom intervalu; poredak: ECTS DESC, potom ime kolegija ASC
- ... pozitivna ocjena: $\text{grade} > 1$
 - > nije dio upita ($\text{grade}=?$; $\text{grade}>?$; $\text{grade}<=?$; etc)
 - > zapisi s negativnim ocjenama uopće ne smiju doći u tablicu!

OBLIKOVANJE BP U CASSANDRI - PRIMJER



- Q3: prikaži sve kolegije s pozitivnom ocjenom za danog studenta u danom vremenskom intervalu; poredak: ECTS DESC, potom ime kolegija ASC
- ... poredak: ECTS DESC, potom ime kolegija ASC
 - > ECTS: C↓, name: C↑
- ... složeni ključ konc. modela: **stud_id, course_id, year**
 - > stud_id i year su već K/C
 - > course_id: C (asc ili desc)

OBLIKOVANJE BP U CASSANDRI - PRIMJER



- › Q3: prikaži sve kolegije s pozitivnom ocjenom za danog studenta u danom vremenskom intervalu; poredak: ECTS DESC, potom ime kolegija ASC
 - › stud_id=? , (K)
 - › year: C (asc ili desc); ECTS: C↓, name: C↑; course_id: C (asc ili desc)
- › postoje i uvjet pretraživanja s nejednakošću (year) i kolone po kojima se sortira (ECTS, name)
- › uvjeti pretraživanja s nejednakošću (year) imaju VIŠI PRIORITET u GRUPNOM KLJUČU od kolona za sortiranje (ECTS, name)

| year↓ | ECTS↓ | name↑ |
|-------|-------|-------|
| 2015 | 5 | Prog |
| 2015 | 6 | OE |
| 2014 | 5 | Prog |
| 2013 | 6 | Mat1 |

moguće pretraživanje po intervalu godina

| ECTS↓ | name↑ | year↓ |
|-------|-------|-------|
| 6 | Mat1 | 2015 |
| 6 | OE | 2013 |
| 5 | Prog | 2015 |
| 5 | Prog | 2013 |

nije moguće pretraživanje po intervalu godina!

OBLIKOVANJE BP U CASSANDRI - PRIMJER



| courses_passed_by_student | |
|---------------------------|----|
| stud_id | K |
| year | C↓ |
| ECTS | C↓ |
| course_name | C↑ |
| course_id | C↑ |
| first_name | S |
| last_name | S |
| weekly_hours | |
| grade | |
| <exams> | |



BATCHES

- › Cassandra batch: skup naredbi koje se izvršavaju zajedno, kao jedna atomna jedinica
- › zbog udvostručavanja podataka kao temeljnog načela oblikovanja, nemoguće izbjegći simultano pisanje (insert/update) u više tablica

```
BEGIN BATCH
```

```
    INSERT INTO students_by_town (town_name, town_post_code,  
        stud_id, last_name, first_name) VALUES (...);  
  
    INSERT INTO students_by_course (course_name, course_id,  
        stud_id, last_name, first_name) VALUES (...);
```

```
APPLY BATCH;
```

- › Cassandra batch ≠ RDBMS batch

LAKE TRANSAKCIJE (LIGHTWEIGHT TRANSACTIONS)

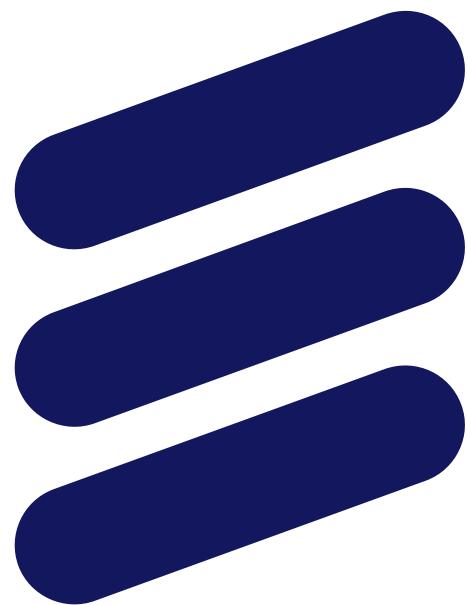
- › ACID transakcije nisu (osnovno) obilježje Cassandre
- › pojedinačni INSERT/UPDATE je po prirodi atoman i trajan (*atomic and durable*)
- › *batches* su atomni i trajni
- › trenutna konzistentnost može se ostvariti odabirom odgovarajuće konzistentnosti pisanja i čitanja
- › izolacija?
 - batches ne osiguravaju izolaciju
- › poseban slučaj izolacije s razinom *serializable* postoji za operacije tipa „čitaj_provjeri_piši“

LAKE TRANSAKCIJE (LIGHTWEIGHT TRANSACTIONS)

```
INSERT INTO customer_account (customerID,  
customer_email) VALUES ('johns', 'jsmith@gmail.com')  
IF NOT EXISTS;
```

```
UPDATE customer_account SET  
customer_email='john.smith@gmail.com' IF customerID=  
'johns';
```

- › **INSERT:** najprije se provjerava primarni ključ
- › **UPDATE:** provjerava se zadani uvjet
- › algoritam Paxos
 - cca. 6 puta sporije od standardne operacije pisanja



ERICSSON