UVOD V NAČRTOVANJE PODATKOVNIH BAZ

Iztok Savnik

Literatura

Predstavljena snov temelji na knjigah:

1.Toby Teorey, Sam Lightstone, Tom Nadeau, Database Modeling and Design: Logical Design, Elsevier, 2006.

2.Grady Booch, James Rumbaugh, Ivar Jacobson, The Unified Modeling Language User Guide, 2000, Addison Wesley.

3.Roger S.Pressman, Software Engineering: A Practicioner's Approach, McGraw Hill, 2010.

4. Raghu Ramakrishnan, Johannes Gehrke, Database Management Systems, McGraw Hill, 2007.

Razporeditev

- Mini projekt (60%)
- Ustni izpit (40%)
- Delni oceni morata biti pozitivni

Mini-projekt

- Delujoč informacijski sistem
- Sistem za delo s podatkovnimi bazami MySQL
  - Lahko uporabite tudi drug sistem
  - Okolje za MySQL, podpora, ... asistent
- Razvoj informacijskega sistema
  - Opis metodologij, tehnologij, principov, ...
  - Predavanja

Mini-projekt

- Načrtovanje in implementacija sistema
  - Analiza potreb
  - Postopkovno in podatkovno načrtovanje
  - Uporaba specifkacij
- Tehnologije
  - MySQL, PostgresSQL, Virtuoso, ...
  - PhP, JSP, ...

Mini-projekt

- Primeri:
  - Spletna banka
  - Spletna trgovina
  - Športni portal
  - Socialni sistem za izmenjavo glazbe
  - Podjetje Taksi
  - Bolha
  - IP-TV operater
  - Portral filmov
  - ...

Uvodni koncepti

Podatki

Podatovni element

Skupina podatkov

Zapis

Datoteka

Podatkovna baza

Sistem za upravljanje s podatkovnimi bazami (SUPB)

Administrator podatkovnih baz (DBA)

Podatki

- Dejstvo
- Nekaj na osnovi česa se lahko sklepa
- Informacija ali znanje ima vrednost
- Podatki imajo ceno

Podatkovni element

- Najmanjši imenovana enota podatka, ki ima pomen v realnem svetu

- Primeri: priimek, naslov, emso, politična stranka

Skupina podatkov

- Agregacija podatkov
- Kolekcija povezanih podatkov, ki tvori koncept
- Enostavna skupina je fiksna kolekcija podatkov
- Primeri: datum, množica alias imen
- Skupina podatkov ima variabilno dolžino

Zapis

- Skupina povezanih podatkov, ki se obravnava kot enota iz strani aplikacijskega programa.
- Primeri: predsedniki, študenti, fakultete.

Datoteka

- Zbirka zapisov danega tipa.
- Primeri: predsednik, student, fakulteta.

Podatkovna baza (zbirka)

- Kolekcija povezanih podatkov, ki jo uporablja več uporabnikov v eni ali več organizacijah.
- Kolekcija tabel v relacijskem modelu.

SUPB

- Sistem za upravljanje podatkovnih baz
- Programski sistem za shranjevanje in spreminjanje podatkovnih baz.
- Vsebuje:
- Logični pogled (shema, pod-shema).
- Fizični pogled (metode dostopa, skupki).
- Jezik za definicijo podatkov.
- Povpraševalni jezik.
- Varnost, obnavljanje, integriteta, itd.

Administrator podatkovnih baz

- Oseba ali skupina oseb odgovorna za učinkovito uporabo tehnologij SUPB v organizaciji.

Smisel upravljanja podakovnih baz

- Dostopnost podatkov.
- Integriteta podatkov.
- Privatnost (cilj) in varnost (orodje).
- Kontrola upravljanja.
- Podatkovna neodvisnost (relativen izraz).
- Fizična podatkovna neodvisnost.
- Logična podatkovna neodvisnost.

Dostopnost podatkov

- Omogoči dostop do zbirke podatkov širokem naboru raznolikih uporabnikom.
- Razumna cena dostopa: performanse popravkov podatkov, redundanca, ...
- Smiselen format podatkov: DDL, DD, ...
- Enostaven dostop: 4GL, SQL, forms, okna, meniji, ...

Integriteta podatkov

- Zagotovi korektnost in veljavnost podatkov.
- Kontrolna točka / Ponovitev / Obnavljanje
- Kontrola vzporednosti.
- Administracija, sledenje poteka (finančno, legalno)

Privatnost in varnost

- Kontrola dostopa
- Dovoljenja, vloge
- Shema / pod-shema gesla.

Kontrola upravljanja

- DBA
- Kontrola življenskega cikla
- Usposabljanje
- Vzdrževanje

Podatkovna neodvisnost

- Izogibamo se ponovnem programiranju aplikacij.
- Omogoča enostavno pretvarjanje in reorganizacijo.
- Logična PN: program se ne spremni ob spremembi sheme.
- Fizična PN: program se ne spremeni ob spremembi predstavitve podatkov in ob spremembi metod dostopa.

Primer SSN:

- Primer številke socialnega zavarovanja:
- Spremenili so zapis čekov iz $999,99 v $9999.99.
- Morali so spremeniti 600 aplikacijskih programov.
- 20,000 delovnih ur (10 č/l).

Primer Y2K:

- Primer „Year 2000“ :
- 00 pomeni 1900.
- Veliko dodatnega dela za spremembo programskih sistemov.

Zakaj izdelujemo modele?

- Uspešna softverska firma konstantno zagotavlja kvalitetno programsko opremo po potrebah uporabnika.
- Organizacija, ki lahko razvija takšno programsko opremo z učinkovito uporabo virov (materialnih in človeških) ima stabilen posel.

Zakaj izdelujemo modele?

- Osnovni produkt razvojnega teama je programska oprema!
  - Ne lepa dokumentacija, svetovno-znane delavnice, odlični slogani, itd.
  - Seveda kvalitetna dokumentacija je potrebna za stabilen razvoj.
- Kvalitetna programska oprema, ki zadovoljuje uporabnike in poslovno okolje!
- Stabilna in kvalitetna programska oprema zahteva solidno definicijo arhitekture sistema.
  - Prilagodljivost zasnove sistema je pomembna

Zakaj izdelujemo modele?

- Hiter in učikovit razvoj kvalitetne programske opreme zahteva:
  - prave ljudi,
  - prava orodja in
  - pravo usmeritev.
- Da bi lahko izvajali vse to konsistentno in predvidljivo je potrebno imeti uglašen razvojni proces, ki se lahko prilagodi spremenljivim potrebam posla.

Zakaj izdelujemo modele?

- Modeliranje je osnovna aktivnost, ki vodi do izdelave kvalitetne programske opreme.
- Modele gradimo za:
  - Komunikacijo glede željene strukture in obnašanja sistema z uporabnikom
  - Vizualizacija in kontrola nad načrtovanim sistemom
  - Boljše razumevanje sistema, ki ga gradimo kar običajno vodi do poenostavitev in ponovne uporabnosti
  - Modele gradimo za upravljanje s tveganjem!

Pomembnost modelov

- Gradbeni projekt - izgradnja hiše
  - Pasja hišica
    - Deske, late in žeblji,
  - Običajna hiša
    - Skice, načrt elektrike, vode in ogrevanja, načrt hiše
    - Lastno delo, najemanje delavcev, skupin
  - Nebotičnik
    - Deske in žeblji :) ?
    - Investitor verjetno je organizacija in potrebuje načrt
    - Vodja samo vodi teame, ki načrtajo, naredijo in potem izvedejo kompletno delo
    - Potreben je natančen načrt kompletne konstrukcije, statike, oblikovanja, napeljav, izvedbe, itd.

Pomembnost modelov

- Precej programskih sistemov se začne kot pasja hišica.
- Pravi ljudje, pravi razvoj, urejenost planetov, ...
- Hiša ali nebotičnik v programski opremi zahteva več kot samo pisanje velike količine kode
  - Pisanje prave kode
  - Kako napisati manj kode ?
- Kdaj pasja hiša zrasla zaradi uspeha kolapsira zaradi lastne teže?
- Neuspešni projekti padejo na zelo različne načine – uspešni projekti imajo precej skupnega.

Pomembnost modelov

- Pomemben skupen element je uporaba modeliranja
  - Modeliranje ni samo del softverske industrije
- Modeliranje je uveljavljena in široko sprejeta inženirska tehnika
  - Avtomobili, letala, ...
    - Modeli delovanja, vetrni tuneli, prototipi
  - Ekonomija, sociologija, filmska industrija, poslovni sistemi
    - Finančni modeli, matematični modeli, vizualizacija, projekcije, ...
- Gradimo modele hiš zato, da naročniki lahko vizualizirajo končen izdelek

Pomembnost modelov

- Model je poenostavitev realnosti
- Kaj je lahko model?
  - Podroben shematski načrt sistema
  - Abstrakten načrt ali pogled od zgoraj
  - Model nivoja abstrakcije: elementi relevantni za nivo abstrakcije
  - Imamo različne vrste modelov: model strukture sistema, obnašanja sistema, itd.

Pomembnost modelov

- Modele gradimo zato, da bolje razumemo sistem, ki ga razvijamo.
- Dosežemo štiri cilje:
  - Model nam pomaga vizualizirati sistem kot naj bi bil oz. želimo, da bi bil
  - Model nam omogoča predstaviti strukturo in obnašanje sistema
  - Model nam da vzorec na osnovi katerega zgradimo sistem
  - Model dokumentira odločitve, ki smo jih sprejeli

Pomembnost modelov

- Modeliranje ni samo za velike sisteme
  - Tudi pasja hišica lahko izgleda lepše
- Velike sisteme modeliramo zato ker ne moremo v umu zaobjeti vseh aspektov in podrobnosti
  - Človek običajno ima meje pri dojemanju kompleksnih sistemov
  - Preko modela usmerimo pozornost na aspekt sistema
  - Osnovni princip: deli in vladaj
  - Preko modeliranja vzpodbujamo človeški intelekt
  - Pravilno izbran model omogoča delo na višjih nivojih abstrakcije

Principi modeliranja

- Štiri osnovni principi modeliranja:

1. Izbira modelov ima zelo velik vpliv na način reševanja problema in na obliko rešitve
   - Pravi modeli osvetlijo težek problem in dajejo vpogled v problem, ki siscer nebi bil mogoč
   - Nepravilni modeli lahko zapeljejo in povzročijo fokus na napačne aspekte

- Kakšno rešitev bo izdelal:
  - Načrtovalec podatkovnih baz?
  - Strukturni analitik?
  - Načrtovalec objektnih sistemov?

Principi modeliranja

2. Vsak model lahko izrazimo na različnih nivojih podrobnosti.
   - Včasih je potreben pogled iz višine 5km, zato da se vidi vpetost v okolico
   - Drugič je potreben pogled na podrobnosti npr. vodovodno napeljavo
   - Podobno je pri programskih sistemih
   - Grob prototip vmesnika, natančen študij omrežne programske opreme pred implementacijo, itd.

Principi modeliranja

3. Najboljši modeli so povezani z realnostjo.

- Primeri:
  - Fizični model stavbe se odziva enako kot realen model in ima omejeno vrednost
  - Matematični model letala, ki predpostavlja idealne pogoje in izdelavo lahko zakrije kritične napake
  - Najbolje bi bilo uporabljati modele, ki imajo jasno povezavo z realnostjo
  - Zavedati se je potrebno kaj model zakrije
  - Zaradi povezav z realnostjo je velikokrat razkol med modelom analize in načrtanim dejanskim modelom sistema

Principi modeliranja

4. En model ne zadošča. Vsak netrivialen sistem je najbolje predstaviti z majhno množico neodvisnih modelov.
   - Operativni plani so večinoma „skoraj neodvisni“
   - Na primer, električna shema in fizični načrt nadstropja sta neodvisna vendar vseeno povezana
   - Potrebno je imeti modele na katerih lahko delamo neodvisno vendar so vseeno med sabo povezani
   - V objektno-usmerjenem svetu imamo več ločenih pogledov: diagrami uporabe, načrtovalski pogled, procesni pogled, implementacijski pogled, in inštalacijski pogled

Proces programskega inženirstva (PI)

- Analiza potreb
- Načrtovanje
  - Arhitekturno načrtovanje
  - Načrtovanje komponent
  - Načrtovanje na osnovi vzorcev
- Implementacija
- Zagotavljanje kakovosti
  - Testiranje
- Vzdrževanje

Analiza potreb

- Faze analize potreb
  - Zametek, elicitacija, primeri uporabe, model potreb, dogovarjanje, validiranje.
- Modeli analize potreb
  - Use-case, scenariji, diagrami razredov, diagrami obnašanja, tok podatkov, vzorci.

Načrtovanje

- Proces načrtovanja
  - Spiralni, po korakih, iterativen
- Principi načrtovanja
  - Abstrakcija, arhitektura, vzorci, modularnost, skrivanje informacij, funkcijska neodvisnost, postopna izboljšava, reorganizacija, ...
- Načrtovalski model
  - Model podatkov, model arhitekture, model vmesnika, ...

Arhitekturno načrtovanje

- Načrtovanje vodeno z arhitekturo
- Stili, vrste, vzorci,
- Osnovni stili
  - Funkcijski, podatkovno voden, podatkovni tokovi, objektno-usmerjen, nivojski, voden s programskim jezikom...
- Pregled alternativnih načrtov
- Preslikava arhitekture v nižje modele

Implementacija

- Izbor programskega sistema
  - Orodja, knjižnice, programska okolja, sistemi, podatkovne baze, načrtovalski sistemi, programski jeziki, ...
- Načrtovanje za kakovost
  - Testiranje
  - Dokumentacija
- Avtomatsko prevajanje

Zagotavljanje kakovosti

- Kaj je zagotavljanje kakovosti?
- Testiranje sistema
  - Načrtovanje testibilnosti, testiranje komponent, testiranje končnega sistema, sistemi za delo z napakami
- Doseganje kakovosti
  - Metode PI, upravljanje projektov, kontrola kvalitete, zagotavljanje kvalitete

Vzdrževanje

- Načrtovanje programskega sistema za vzdrževanje
- Najdaljaša faza PI

Življenjski cikel relacijskih podatkovnih baz

- Analiza potreb.
- Logično načrtovanje podatkovnih baz.
- Fizično načrtovanje podatkovnih baz.
- Implementacija, monitoring in vzdrževanje.

Življenski cikel načrtovanja PB

- Korak 1: Analiza zahtev.
- Korak 2: Logično načrtovanje.
- Korak 2(a): Konceptualno načrtovanje podatkov.
- Korak 2(b): Združevanje pogledov.
- Korak 2(c): Transformacija konceptualne sheme v SQL tabele.
- Korak 2(d): Normalizacija SQL tabel.
- Korak 3: Fizično načrtovanje.
- Indeksiranje, skupine, particije, okna, denormalizacija.
- Korak 4: Implementacija, monitoring in vzdrževanje.

Analiza zahtev

- Naravna razmerja med podatki (neodvisna od procesa).
- Funkcionalne zahteve (odvisne od procesov).
- Strojna / programska platforma (OS,DBMS).
- Performanse in integritetne omejitve.
- Rezultat:
- Specifikacija zahtev in podatkovni slovar.

Logično načrtovanje podatkovnih baz

- Konceptualno načrtovanje podatkovnih baz.
- Integracija shem in oken.
- Transformacija konceptualnega modela v SQL.
- Normalizacija SQL tabel.
- Rezultat:
- Globalna shema PB
- Predstavitev s tabelami.

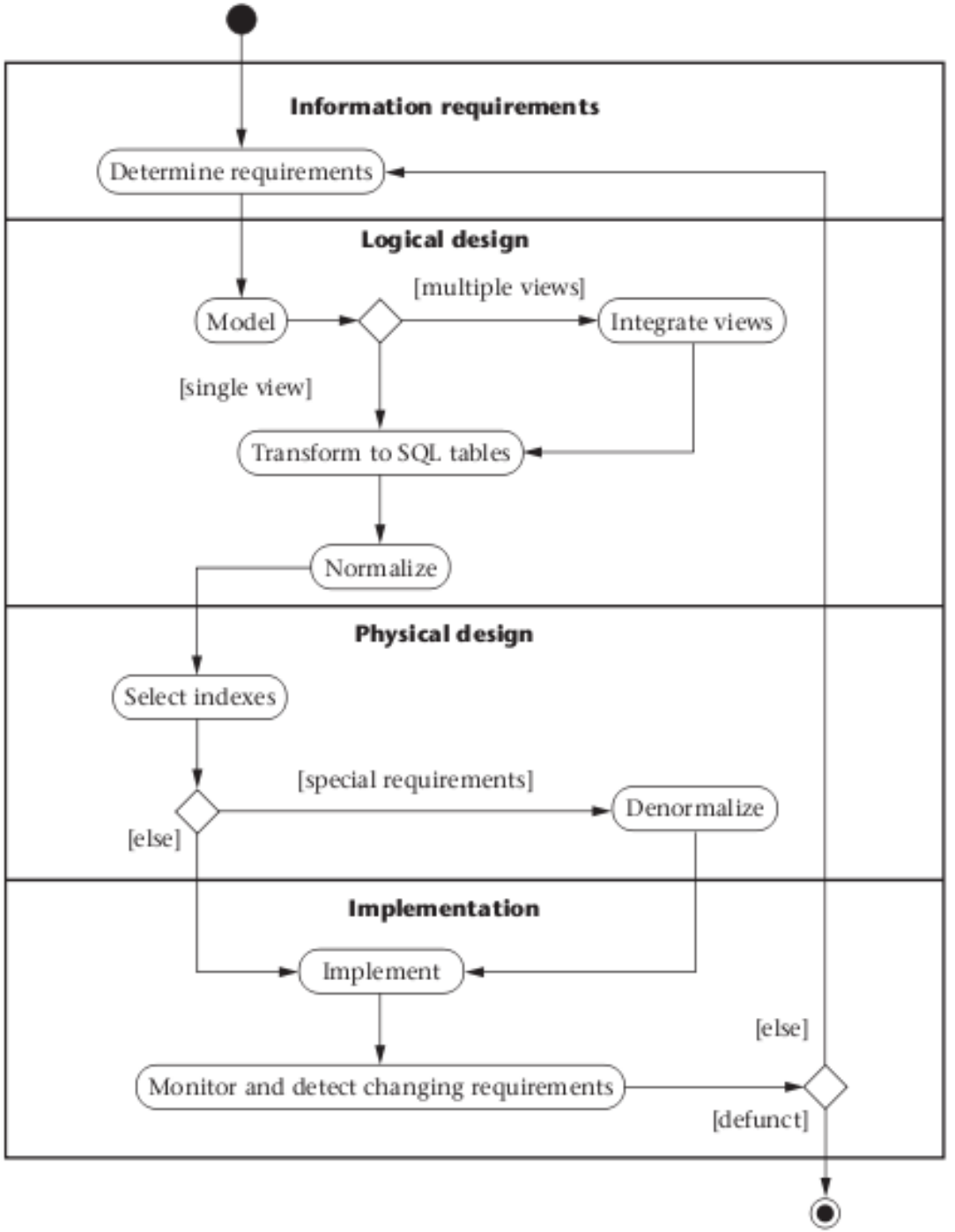
Fizično načrtovanje podatkovnih baz

- Izbor indeksov
- Materializirana okna, skupki, particioniranje, denormalizacija
- Porazdelitev podatkov po mreži

Implementacija, monitoring in spreminjanje PB

- Implementacija formalne sheme
- Uporaba orodij: DDL, DML
- Avtomatsko prevajanje
- Spremljanje izvajanja aplikacije
- Testiranje, monitoring,
- Vzdrževanje, popravki, ...
- Odstopanja od zasnove
- Nova spoznanja
- Konstantno vzdrževanje, popravki, ...

Življenski cikel načrtovanja podatkovne baze



Enostaven primer

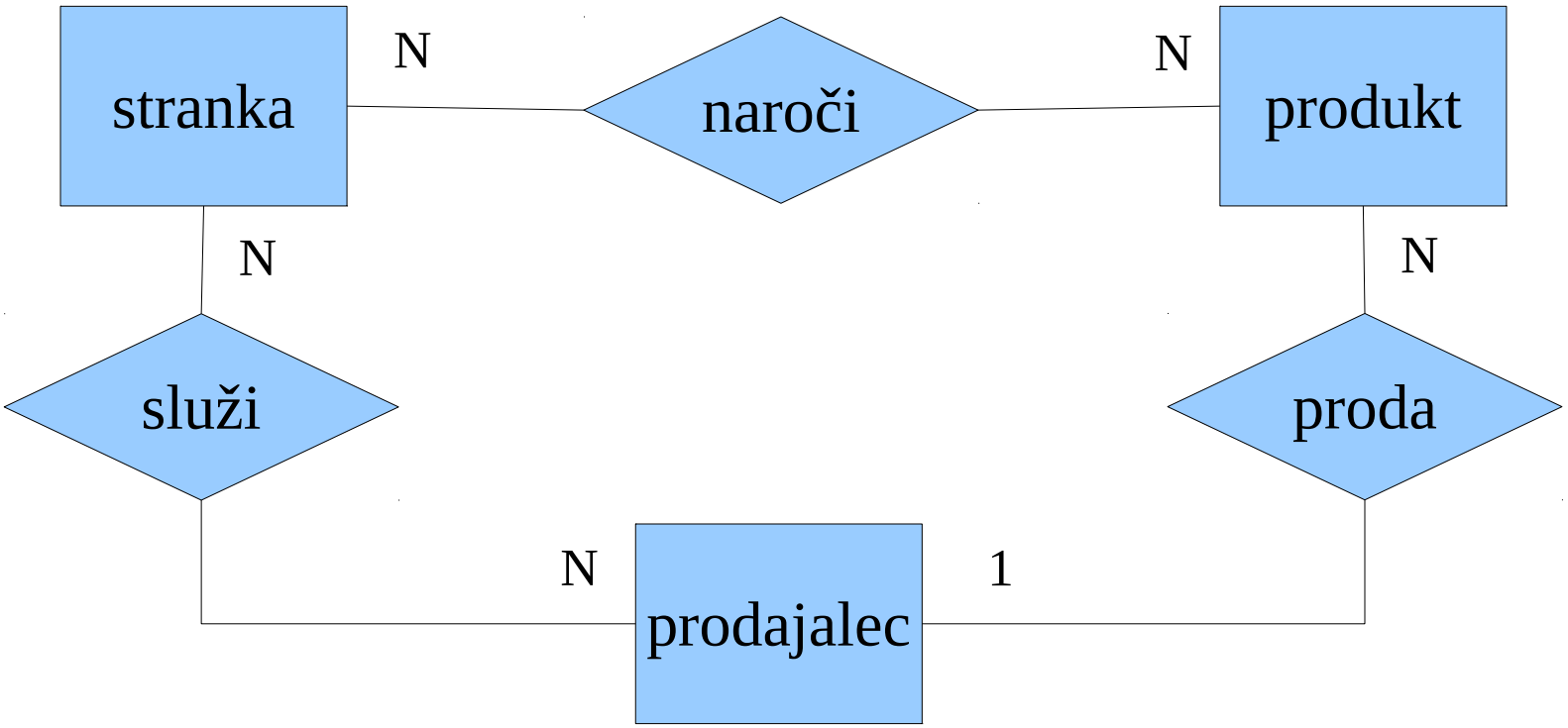
- Informacijski sistem dobavitelja
- Naročila, produkti, stranke, ...

Korak I: Analiza zahtev

Korak II: Logično načrtovanje

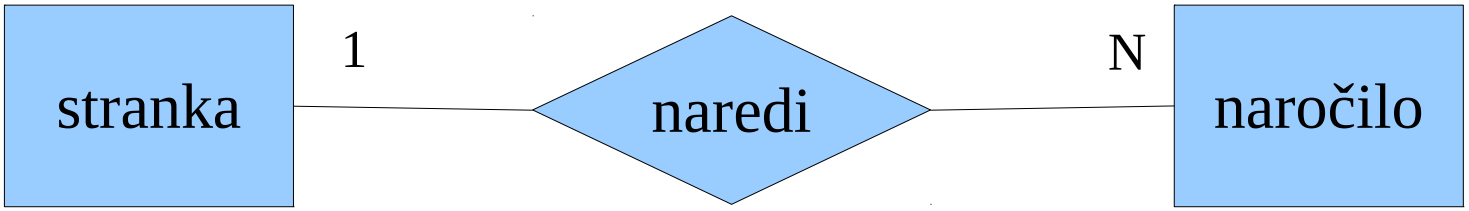
II(a) - konceptualno načrtovanje

Pogled prodajalca:

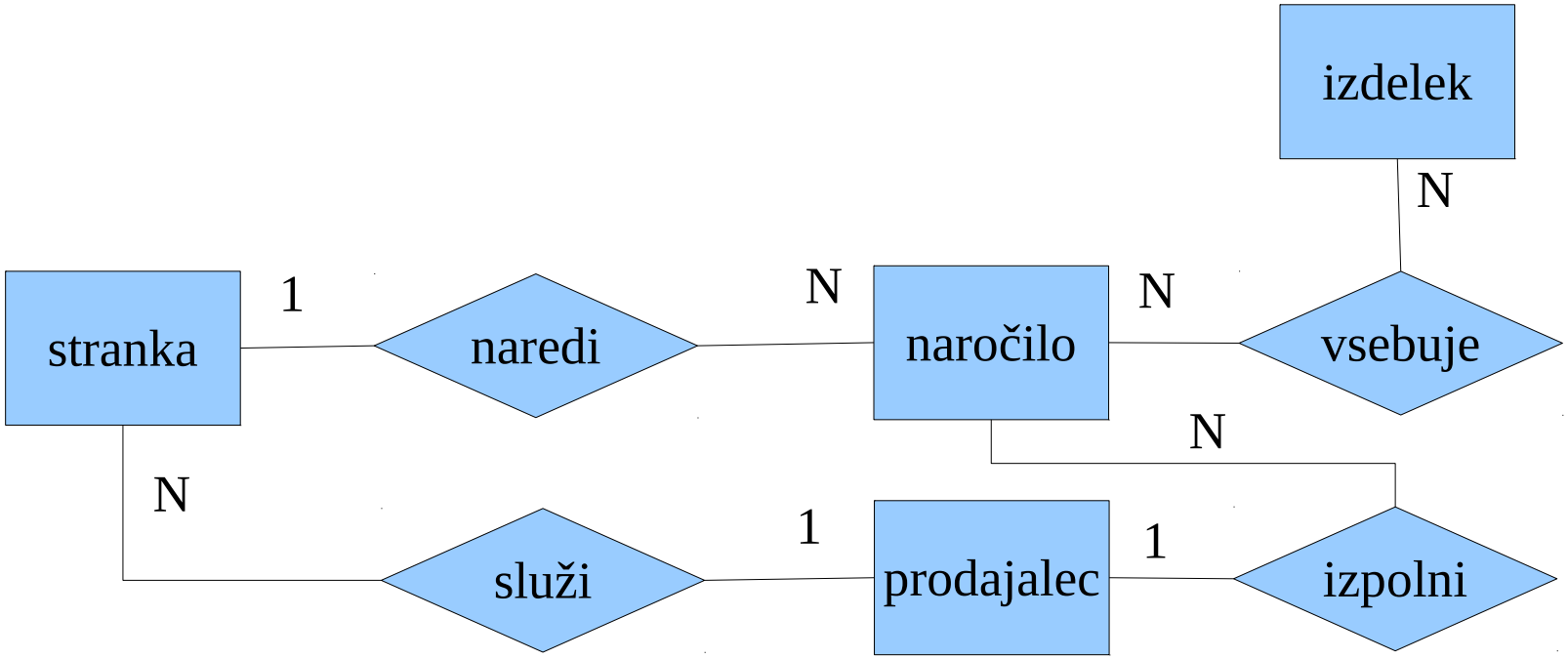


Korak II(b) - Integracija pogledov

Pogled stranke:



Korak II(b) - Integracija pogledov



Korak II(c): Prevod v tabele

Stranka

stran-id	stran-ime	...

Izdelek

izde-id	izde-ime	...

```
create table stranka (
    stran_id integer,
    stran_ime char(15),
    stran_naslov varchar(30),
    prod_ime char(15),
    izde_id integer,
    primery key (stran_id),
    foreign key (prod_ime)
        references prodajalec,
    foreign key (izde_id)
        references izdelki);
```

Prodajalec

prod-ime	prod-nasl	prod-odd	nivo-posl	dni-dopu

Naročilo

naro-id	prod-ime	stran-id

Naro-izde

naro-id	izde-id

Korak II(d): Normalizacija

Dekompozicija tabel in anomalije pri popravljanju.

Prodajalec

prod-id	prod-nasl	prod-odd	nivo-posl

Prod-dopu

nivo-posl	dni-dopu

Objektno-usmerjeno modeliranje

- V gradbeništvu uporabljajo zelo veliko modelov
  - Najbolj pogosti so strukturalni modeli
  - Omogočajo vizualizacijo vseh delov sistema
  - Dinamični modeli
- Najbolj pogosti pristopi k modeliranju programske opreme:
  - Algoritmični vidik
  - Objektno-usmerjen vidik

Objektno-usmerjeno modeliranje

- Algoritmično modeliranje
  - Osnovni gradnik modeliranja je funkcija oz. procedura
  - Pozornost je usmerjena v kontrolo in dekompozicijo večjih algoritmov v manjše
  - Algoritmični pristop se izkaže kot težko prilagodljiv ob večjih spremembah in rasti sistema

Objektno-usmerjeno modeliranje

- Moderni razvoj informacijskih sistemov temelji na objektno-usmerjenem modeliranju
  - Osnovni gradniki uporabljeni za modeliranje so objekti in razredi
  - Vsak objekt ima identiteto, stanje in obnašanje
  - Naravna predstavitev modeliranega okolja

Objektno-usmerjeno modeliranje

- Primer objektnega sistema: blagajna
  - Trije nivoji sistema: uporabniški vmesnik, aplikacija in podatkovna baza
  - Uporabniški vmesnik sestavljajo razni objekti kot so npr. meniji, gumbi, dialogi, itd.
  - Podatkovno bazo sestavljajo tabele, ki opisujejo domeno: stranke, produkti, naročila, itd.
  - Aplikacijo sestavljajo transakcije, poslovna pravila, operacije nad objekti kot so stranke, produkti, itd.

Objektno-usmerjeno modeliranje

- Objektni sistemi so v zadnjih desetletjih glaven tok razvoja
  - Praktična uporabnost in praktični rezultati
  - Primerni so za modeliranje sistemov vseh velikosti in kompleksnosti
- Večino orodij je objektno-usmerjenih
  - Programski jeziki, podatkovne baze, operacijski sistemi, načrtovalska orodja, itd.
  - Pomagajo videti svet „objektno“

Objektno-usmerjeno modeliranje

- Posledice izbora objektno-usmerjenega modela
  - Kakšna je struktura dobrega objektno-usmerjenega sistema?
  - Kako pristopiti k objektno-usmerjenem modeliranju?
  - Kakšne izdelke naj kreira projekt?
  - Kdo naj kreira izdelke?
  - Kako meriti izdelke?
- Vprašanja s katerimi se bomo ukvarjali ...
  - Objektno-usmerjen model
  - UML

Značilnosti dobrega procesa načrtovanja podatkovnih baz

- Iterativna analiza potreb.
- Izboljšava po korakih.
- Iterativno ponovno načrtovanje.
- Dobro definiran team in postopek za pregled načrta.
- Kdaj narediti pregled načrta.
- Pravila za pregled načrta.

Iterativna analiza potreb

- Naredi intervjuje od zgoraj-navzdol.
- Uporabi enostavne modele za podatkovne tokove in razmerja med podatki.
- Preveri model.

Izboljšava po korakih.

- Iterativno ponovno načrtovanje (re-design).

Dobro definiran team in postopek za pregled načrta

- Načrtovalci podatkovne baze.
- SUPB programska skupina.
- Končni uporabniki iz pod-področij aplikacije.
- Kdaj pregledati načrt?

Pravila za pregled načrta.

- Kratka dokumentacija vnaprej.
- Formalna predstavitev.
- Kritika produkta, ne oseb.
- Cilj je lokalizacija problemov, naredi rešitve offline.

Kdaj narediti pregled načrta

- Po analizi zahtev
- Po konceptualnem načrtovanju
- Po fizičnem načrtovanju
- Po implementaciji (uglaševanju)